Ф. Ж. Таннинг
F. J. Tangning

ОСНОВЫ ПРОГРАММИРОВАНИЯ В PYTHON: ТРИ В ОДНОМ

```
1  import pandas as pa
2
3  d_se = [67, 70, 156.5, 105, 170, 120, 105, 144, 135.5, 99]
4  ind = ["Личи", "Банан", "Авокадо", "Кумкват", "Ананас",
5                  "Апельсин", "Помело", "Лимон",
6                  "Мандарин", "Грейпфрут"]
7  data_se = pa.Series(data=d_se, index=ind)
8  new_data = data_se.sort_index(axis=0, level=None,
9                  ascending=True, kind='stable')
10 print("Названия фруктов, отсортированных в порядке
11                 возрастания: \n%s"%new_data)
```

```
Названия фруктов, отсортированных в порядке возрастания:
Авокадо     156.5
Ананас      170.0
```

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Владимирский государственный университет
имени Александра Григорьевича и Николая Григорьевича Столетовых»

Ф. Ж. ТАННИНГ
F. `J. TANGNING

ОСНОВЫ ПРОГРАММИРОВАНИЯ В PYTHON: ТРИ В ОДНОМ

THE BASICS OF PYTHON PROGRAMMING: THREE IN ONE

Учебное пособие (The course book)

Электронное издание (Electronic publication)

В четырех томах (In four volumes)

Том 2 (*Volume II)*

Владимир 2022

УДК 004.432
ББК 32.973.2

Рецензенты:
Кандидат технических наук, доцент
доцент кафедры информационных систем
Тюменского государственного университета
Ю. Е. Карякин

Доктор технических наук, профессор
зав. кафедрой информационных систем и программной инженерии
Владимирского государственного университета
имени Александра Григорьевича и Николая Григорьевича Столетовых
И. Е. Жигалов

Таннинг Ж., Ф.

Основы программирования в Python: три в одном [Электронный ресурс] : учеб. пособие. В 4 т. Т. 2 / Ф. Ж. Таннинг ; Владим. гос. ун-т им. А. Г. и Н. Г. Столетовых. – Владимир : Изд-во ВлГУ, 2022. – 472 с. – ISBN 978-5-9984-1361-2 (т. 2). – ISBN 978-5-9984-1227-1. – Электрон. дан. (8,92 Мб). – 1 электрон. опт. диск (CD-ROM). – Систем. требования: Intel от 1,3 ГГц ; Windows XP/7/8/10/11, Linux, Mac OS ; Adobe Reader ; дисковод CD-ROM. – Загл. с титул. экрана.

В книге описывается процесс программирования на языке Python с использованием практических примеров и упражнений. Цель пособия состоит в том, чтобы научиться программировать индивидуально в определенных научных областях: обработка файлов, анализ данных с помощью библиотеки Pandas и др. Обучающий материал пособия также дает возможность иностранным студентам освоить дисциплину, преодолевая языковой барьер. Студенты, владеющие русским языком, могут совершенствовать свои знания на французском и английском благодаря синхронизации материала, представленного на этих языках.

Издание рекомендовано для студентов различных технических направлений, преподавателям вузов и всем, кто начинает делать свои первые шаги в программировании, используя язык Python в качестве инструмента программирования.

Рекомендовано для формирования общепрофессиональных компетенций в соответствии с ФГОС ВО.

Табл. 93. Ил. 39. Библиогр.: 19 назв.

УДК 004.432
ББК 32.973.3

ISBN 978-5-9984-1361-2 (т. 2)
ISBN 978-5-9984-1227-1

UDC 004.432
BBK 32.973.2

Reviewers:
Candidate of Technical Sciences, Associate Professor
Associate Professor of the Department of Information Systems
of Tyumen State University
Yu. E. Karyakin

Doctor of Technical Sciences, Professor
Head of the Department of Information Systems and Software Engineering
of Vladimir State University named after Alexander and Nikolai Stoletovs
I. E. Zhigalov

Tangning J., F.

The basics of python programming: three in one [Electronic resource] : the course book. In 4 vol. Vol. 2 / F. J. Tangning ; Vladimir State University named after Alexander and Nikolai Stoletovs. – Vladimir : Publishing house VlSU, 2022. – 472 p. – ISBN 978-5-9984-1361-2 (vol. 2). – ISBN 978-5-9984-1227-1. – Electronшc data (8.92 MB). – 1 optical disk (CD-ROM). – Systems Requirements: CPU Intel from 1.3 GHz; Windows XP/7/8/10/11, Linux, Mac OS ; Adobe Reader; CD-ROM drive. – Name from the title screen.

The course book describes the process of programming in Python language using practical examples and exercises. The goal of the manual is to learn to program individually in certain scientific fields: file processing, data analysis using the Pandas library, etc. The teaching material in the book enables foreign (non-Russian) students to master the discipline by overcoming the language barrier. Students who already speak Russian can improve their knowledge of French and English through the synchronization of the material presented in these languages.

The publication is recommended for students of various technical fields, teachers and anyone who is starting to take their first steps in programming, using the Python language as a programming tool.

It is also recommended for the acquisition of general professional skills in accordance with the higher education standards of the Russian Federation.

Tabl. 93. Il. 39. Bibliogr.: 19 titl.

UDC 004.432
BBK 32.973.2

ISBN 978-5-9984-1361-2 (vol. 2)
ISBN 978-5-9984-1227-1

RU СПИСОК ОСНОВНЫХ СОКРАЩЕНИЙ
EN MAIN ABBREVIATIONS
FR LES PRINCIPALES ABRÉVIATIONS

RU – означает, что за этой частью главы на русском языке следует перевод на английский и французский языки
EN – означает, что за этой частью главы на английском языке следует перевод на русский и французский языки
FR – означает, что за этой частью главы на французском языке следует перевод на русский и английский языки

RU – *means that this part of the chapter in Russian is followed by a translation into English and French*
EN – *means that this part of the chapter in English is followed by a translation into Russian and French*
FR – *means that this part of the chapter in French is followed by a translation into Russian and English*

RU – *signifie que cette partie du chapitre en russe est suivie d'une traduction en anglais et en français*
EN – *signifie que cette partie du chapitre en anglais est suivie d'une traduction en russe et en français*
FR – *signifie que cette partie du chapitre en français est suivie d'une traduction en russe et en anglais*

ОГЛАВЛЕНИЕ

TABLE OF CONTENTS

TABLE DES MATIÈRES

RU ПРЕДИСЛОВИЕ

Начнем с того, что эта книга является продолжением серии томов, посвященных изучению основ программирования на языке Python посредством упражнений.

В первом томе обоснован выбор этого языка для объяснения основных принципов программирования. Во втром томе мы не будем возвращаться к этому вопросу, но можем просто напомнить, что Python – один из самых простых языков программирования; он лёгок для понимания и с его помощью можно выполнить практически все виды задач в различных областях повседневной и научной деятельности:

- решение формальных ежедневных задач;
- решение задач со свойствами искусственного интеллекта;
- решение задач по обработке больших объемов данных;
- решение задач в веб-сфере и т. д.

Будем продолжать в том же духе объяснять новые темы, используя тот же подход к обучению (который использовался в первом томе), основанный на реальных примерах, позволяющий усвоить программирование на языке Python.

Как установить виртуальную среду и запустить ее в операционную систему Windows?

1) необходимо создать каталог (например, каталог “Virtuald” на диске “C”), из которого будут установлены все файлы;

2) потом, выполнить установку модуля «***virtualenv***», который позволяет создать виртуальную среду:

c:\work\virtuald>***pip install virtualenv***

3) из ранее созданного каталога (C:\Work\virtuald>), потребуется создать виртуальную среду, запустив следующую команду: ***python -m venv namvirtual*** (параметр “*-m*” позволяет использовать только

модули, подходящие для этой версии Python; и "*namvirtual*" – любое виртуальное имя).

4) Если все действия проходят нормально, то создается каталог с именем «namvirtual» со следующим содержимым:

- в каталогах: «Include», «Lib», «site-packages» и «Scripts»;
- из файлов: «python.exe», «pip.exe», «pip3.exe», «Activate.bat», «deactivate.bat», «pyvenv.cfg», ...

5) Чтобы запустить эту виртуальную среду, просто выполните следующую команду: «activate», которая находится в каталоге: «C:\Work\virtuald\namvirtual\Scripts». В таком состоянии можно установить в эту виртуальную среду необходимые библиотеки. Кроме того, с помощью команды «pip list» можно будет просмотреть список всех библиотек, установленных в этом виртуальном компьютере.

6) Чтобы **завершить** эту виртуальную среду, необходимо в том же предыдущем каталоге выполнить следующую команду: «deactivate».

Библиотеки и версии, установленные и используемые для решения задачи с использованием Python в операционной системе Windows

cffi==1.15.0
chardet==4.0.0
cryptography==36.0.0
distlib==0.3.4
filelock==3.4.2
keyboard==0.13.5
pdfminer.six==20200517
pdfplumber==0.5.28
Pillow==8.4.0
platformdirs==2.4.1
pycparser==2.21
pycryptodome==3.12.0
pynput==1.7.5
six==1.16.0
sortedcontainers==2.4.0
virtualenv==20.13.0

Wand==0.6.7
anaconda-navigator==2.1.1
conda==4.11.0
keras==2.7.0
pandas==1.4.1
pytest==6.2.4

Порядок чтения

Чтобы лучше усвоить и понять примеры, показанные в этом новом томе, и применить их на практике, необходимо будет иметь минимальные знания по программированию на языке Python в следующих разделах:

- понятие переменных;
- понятие о выводах данных;
- как вводить данные;
- применение различных арифметических действий;
- использование исключений для обработки ошибок при выполнении кода;
- использование собственных функций, модулей и создание новых функций.

1. Глава I посвящена файлам, то есть учимся выполнять различные манипуляции с файлами: создавать файлы, читать и вводить данные в файл, обновлять определенную информацию в файле и многое другое.

2. Глава II сосредоточена на взаимодействии данных с PDF-файлами. Здесь мы узнаем, как читать и записывать информацию в такие файлы с помощью библиотек.

3. Глава III основана на обработке строк в Python. Мы не только учимся манипулировать текстом с учетом его кодирования, но и получаем управление над клавиатурой и мышью с помощью программирования.

4. Глава IV посвящена существующим сортировкам алгоритмов, схематическому описанию и наличию некоторых собственных функций сортировки в Python.

5. Глава V – самая объемная работа в книге, которая зарезервирована для библиотеки Pandas, которая является инструментом анализа данных. Предлагаются различные задачи и решения, чтобы понять важность Pandas в науке о «Big DATA» – (Больших Данных).

В том же порядке, что и в первом томе, эта работа также посвящена изучению основ программирования на Python; она направлена на преодоление языковых барьеров для учащихся из франкоязычных и англоязычных стран, обучающихся в вузах Российской Федерации.

Благодарности

Автор считает своим долгом выразить искреннюю благодарность директору института Информационных технологий и радиоэлектроники ВлГУ, профессору А. А. Галкину; заведующему кафедрой информатики и защиты информации, д.т.н., профессору М. Ю. Монахову за поддержку и помощь в издании пособия.

Автор будет благодарен обучающимся и заинтересованным читателям за замечания и предложения по содержанию книги, которые просит направлять по электронному адресу: tajifirmin2@yahoo.com

Продолжение на странице 22

EN PREFACE

Let’s start with the fact that this book is a continuation of a series of volumes devoted to the study of the basics of programming in Python with a set of exercises.

The first volume justifies the choice of this language to explain the basic principles of programming. In the second volume we will not return to this issue, but we can simply remind you that Python is one of the easiest languages to understand - with its help you can perform almost all types of problems in various areas of everyday and scientific activity:

- solving formal daily problems;
- solving problems with artificial intelligence properties;
- solving problems for processing large amounts of data;
- solving problems in the web sphere, etc.

We will continue in the same spirit to explain new topics using the same approach to learning that was used in the first volume, based on real examples, allowing you to learn programming in Python.

How do I install a virtual environment and run it on the Windows operating system?

1) you need to create a directory (for example, the “Virtuald” directory on the “C” drive) from which all files will be installed;

2) then, install the «***virtualenv***» module, which allows you to create a virtual environment:

C:\work\virtuald>***pip install virtualenv***

3) from a previously created directory (C:\Work\virtuald >), you will need to create a virtual environment by running the following command: ***python -m venv namvirtual*** (the “-m” parameter allows you to use only modules suitable for this version of Python; and “*namvirtual*" – any virtual name).

4) If all actions are completed successfully, then a directory named “namvirtual” is created with the following contents:

- in the directories: «Include», «Lib», «site-packages» and «Scripts»;
- from files: «python.exe», «pip.exe», «pip3.exe», «Activate.bat», «deactivate.bat», «pyvenv.cfg», ...

5) To start this virtual environment, just run the following command: «*activate*», which is located in the directory: «C:\Work\virtuald\namvirtual\Scripts». In this state, you can install the necessary libraries into this virtual environment. In addition, using the «pip list» command, you can view a list of all the libraries installed on this virtual computer.

6) To terminate this virtual environment, it is necessary to execute the following command in the same previous directory: «*deactivate*».

Libraries and different versions installed and used to solve the problem using Python in the Windows operating system

cffi==1.15.0
chardet==4.0.0
cryptography==36.0.0
distlib==0.3.4
filelock==3.4.2
keyboard==0.13.5
pdfminer.six==20200517
pdfplumber==0.5.28
Pillow==8.4.0
platformdirs==2.4.1
pycparser==2.21
pycryptodome==3.12.0
pynput==1.7.5
six==1.16.0
sortedcontainers==2.4.0
virtualenv==20.13.0
Wand==0.6.7
anaconda-navigator==2.1.1
conda==4.11.0
keras==2.7.0
pandas==1.4.1

pytest==6.2.4

Reading order

To better assimilate and understand the examples shown in this new volume and apply them in practice, it will be necessary to have minimal knowledge of programming in Python in the following sections:

- the concept of variables;
- the concept of data outputs;
- how to enter data;
- application of various arithmetic operations;
- using exceptions to handle errors when executing code;
- using your own functions, modules and creating new functions.

1. Chapter I is devoted to files, that is, we learn how to perform various manipulations with files: create files, read and enter data into a file, update certain information in a file, and much more.

2. Chapter II focuses on the interaction of data with PDF files. Here we will learn how to read and write information to such files using libraries.

3. Chapter III is based on Python string processing. We not only learn how to manipulate text, taking into account its encoding, but also gain control over the keyboard and mouse using programming.

4. Chapter IV is devoted to the existing sorting algorithms, a schematic description and the presence of some proprietary sorting functions in Python.

5. Chapter V – The most voluminous work in the book is reserved for the Pandas library, which is a data analysis tool. Various tasks and solutions are proposed to understand the importance of Pandas in the science of Big DATA.

In the same order as in the first volume, this work is also devoted to the study of the basics of programming in Python; it is aimed at overcoming language barriers for students from French-speaking and English-speaking countries studying at universities in the Russian Federation.

Acknowledgement

The author considers it his duty to express his sincere thanks to the Principle of the Institute of Information Technology and Radio-Electronics – Professor Galkin A. A.; to the Head of the Computer Science Department, Doctor of Technical Sciences, Professor Monakhov M. Yu. for their support and assistance in the publication of the manual.

The author would like to thank the users of the manual for their comments and suggestions on the content of the book, which he requests to send to the following e-mail address: tajifirmin2@yahoo.com

Continued on page 24

FR LA PRÉFACE

Commençons par dire que ce livre est la continuité d'une série de tomes liée aux principes fondamentaux de la programmation en python sous forme d'exercices. Dans le premier tome, nous avons justifier pourquoi avons choisi ce langage pour expliquer ces principes fondamentaux de la programmation. Dans ce deuxième volume, nous ne reviendrons pas là dessus, mais nous pouvons juste rappeler que Python est un des plus simples langages facile à comprendre et capable de résoudre presque toutes sortes d'exercices dans les différents domaines de la vie active et virtuelle:

- exercices quotidiens formels;
- exercices avec des propriétés d'intelligence artificielle;
- exercice sur le traitement des mégadonnées;
- exercices dans le domaine web, etc.

Nous continuons dans la même lancée l'explication des nouveaux thèmes en utilisant la même approche d'apprentissage qui a été utilisée au premier volume, celle qui consiste à partir des exemples concrets, de permettre l'assimilation de la programmation du langage Python.

Comment installer l'environnement virtuel et le mettre en fonctionnement dans le système opérationnel Windows?

1) Créer un répertoire ou catalogue à partir duquel se fera l'installation de tous les fichiers ; par exemple, répertoire "Virtuald" dans le disque "C".

2) Procéder à l'installation du module «***virtualenv***» qui permet de créer un environnement virtuel:

c:\work\virtuald>***pip install virtualenv***

3) À partir du répertoire créé précédemment, (C:\Work\virtuald>), faudra créer un environnement virtuel en lançant la commande suivante: ***python -m venv namvirtual*** (le paramètre "-m" permet d'utiliser que les modules adaptés à cette version du Python; et "namvirtual" - le nom aléatoire du virtuel).

4) Si tout se passe bien, alors un répertoire est créé ayant pour nom «namvirtual» avec le contenu suivant:

- répertoires: «Include», «Lib», «site-packages» et «Scripts»;
- fichiers: «python.exe», «pip.exe», «pip3.exe», «Activate.bat», «deactivate.bat», «pyvenv.cfg», ...

5) Pour mettre cet environnement virtuel en activité, il suffit d'exécuter la commande suivante: «activate» qui se trouve dans le répertoire: «C:\Work\virtuald\namvirtual\Scripts». À cet état, il est possible d'installer dans ce virtuel les bibliothèques voulues. En plus, avec la commande «pip list», il sera possible de voir de lister toutes les bibliothèques installées dans ce virtuel.

6) Pour stopper cet environnement virtuel, il faudra dans le même répertoire précédent exécuter la commande suivante: «deactivate».

Bibliothèques et versions installées et utilisées pour résoudre les exercices en utilisant Python dans le système opérationnel Windows

cffi==1.15.0
chardet==4.0.0
cryptography==36.0.0
distlib==0.3.4
filelock==3.4.2
keyboard==0.13.5
pdfminer.six==20200517
pdfplumber==0.5.28
Pillow==8.4.0
platformdirs==2.4.1
pycparser==2.21
pycryptodome==3.12.0
pynput==1.7.5
six==1.16.0
sortedcontainers==2.4.0
virtualenv==20.13.0
Wand==0.6.7
anaconda-navigator==2.1.1

conda==4.11.0
keras==2.7.0
pandas==1.4.1
pytest==6.2.4

Ordre de lecture

Pour mieux assimiler et comprendre les exemples illustrés dans ce nouveau tome, et afin de les mettre en pratique, il sera indispensable d'avoir un minimum de connaissances de sur la programmation du langage Python dans les parties suivantes:

- notion de variables;
- notion d'affichage des données;
- mode d'introduction des données sur l'écran;
- le rôle des différents opérateurs arithmétiques;
- utilisation des exceptions pour gérer les erreurs dans l'exécution d'un code;
- utilisation des fonctions natives, des modules et la création de nouvelles fonctions.

1. Le chapitre I nous met en relation avec les fichiers c'est-à-dire que nous apprenons à faire une variété de manipulation avec les fichiers: création des fichiers, lecture et introduction des données dans le fichier, mise à jour de certaines informations dans le fichier et bien d'autres.

2. Le chapitre II est accentué sur l'interaction des données avec les fichiers PDF. Ici, nous apprenons comment lire et extraire une information dans ce genre de fichier en utilisant des bibliothèques.

3. Le chapitre III est basé sur le traitement de la chaîne en Python. Nous apprenons non seulement à manipuler un texte en tenant compte de son codage, mais aussi à prendre le contrôle du clavier et de la souris à travers la programmation.

4. Le Chapitre IV est porté sur les tris d'algorithmes qui existent, sur la description schématique et la présence de certaines fonctions natives tris dans Python.

5. Le chapitre V qui est le plus volumineux est réservé à la bibliothèque Pandas qui est l'outil d'analyse des données. Différents exercices et résolutions sont proposés pour comprendre son importance dans la science du «Big DATA».

Dans la même lancée que le premier volume, cet ouvrage est également concentré sur l'étude des bases de la programmation en Python; il vise à surmonter les barrières linguistiques pour les apprenants des pays francophones et anglophones dans les universités de la Fédération de Russie.

Remerciements

L'auteur estime de son devoir d'exprimer ses sincères remerciements à son chef du département d'informatique, docteur en sciences techniques, professeur Monakhov M. Yu; au directeur de l'Institut des Technologies de l'Information et de la Radio-électronique – professeur Galkin A. A. pour le soutien et l'assistance dans la publication du manuel.

L'auteur sera reconnaissant aux utilisateurs du manuel pour leurs commentaires et suggestions sur le contenu du livre, qu'il demande d'envoyer à l'adresse électronique suivante: tajifirmin2@yahoo.com

Suite à la page 26

RU ВВЕДЕНИЕ

В настоящее время мир претерпевает изменения, и мы с очевидностью замечаем, что важно также расширять знания, полученные в виртуальной или компьютерной области. То есть компьютер становится очень важным инструментом, способным влиять на принятие того или иного жизненно важного решения практически во всех областях деятельности человека. Эта виртуальная жизнь, по существу, состоит из программного обеспечения, которое способно влиять или играть важную роль в экономике страны. Для этого необходимо большое количество специалистов, способных создавать привлекательные, полезные и нужные элементы (программное обеспечение) в области информационных технологий.

Таким образом, для достижения этой цели необходимо изучать языки программирования (простые для понимания, универсальные и популярные) с помощью более простых и эффективных методов.

Существует несколько концепций обучения программированию. Одни концепции предпочтительнее других из-за различий в навыках каждого человека в этой области. Выбор концепции частично зависит от цели, которая поставлена перед собой. Например, если программирование изучается с целью последующего обучения ему других, то будет полезно сначала получить больше теории, узнать почему именно так происходят какие-либо явления, а затем присоединить теорию к упражнениям.

Если вы хотите научиться писать коды для создания программного обеспечения, то вам необходимо будет изучить гораздо больше конкретных примеров, а также освоить правила и синтаксисы команд, соответствующих языку.

Независимо от используемой концепции, эффективное изучение программирования требует, прежде всего, знаний общего назначения, чтобы иметь возможность определиться с целями и задачами обучения; после этого важно изучить базовое понятие языка, знать методы решения проблемы, знать, как найти кратчайший путь и эффективный способ решения задачи, узнать, что уже было выполнено в этой области (например, существование стандартных функций), чтобы

просто использовать их, не пытаясь «изобрести велосипед»; и, наконец, важно иметь представление о последовательности обучения (то есть соблюдать порядок приобретения знаний о выбранном языке). Например, необходимо знать, что существует несколько типов переменных, и каждая переменная может иметь определенное количество места для хранения информации в зависимости от типа данных, которые ей адресованы. Это означает, что логически будет нормально сначала узнать о существовании этих типов данных, а затем поговорить о переменной. Если заглянем немного дальше, нам будет интересно еще больше: сначала узнать о значении данных и как они выглядят, прежде чем мы даже поговорим о различных типах данных.

В первом томе утверждалось, что обучающийся, который хочет изучить язык программирования, должен задать себе и ответить на ряд приведённых вопросов. Сегодня мы говорим ему, что какой бы язык ни был выбран, концепция обучения должна оставаться такой же, как мы описываем в этом учебнике. Мы хотим знать, могут ли задачи и примеры с ответами, которые мы вам предлагаем, помочь вам развить свои интеллектуальные способности в программировании.

Продолжение на странице 28

EN INTRODUCTION

Currently the world is undergoing changes, and we clearly notice that it is also important to expand the knowledge gained in the virtual or computer field. That is, the computer becomes a very important weapon capable of influencing the adoption of a vital decision practically in all areas of human activity. This virtual life essentially consists of software that is able to influence or play an important role in the country's economy. This requires a large number of specialists who are able to create attractive, useful and necessary elements (software) in the field of information technology.

Thus, to achieve this goal, it is necessary to learn programming languages (easy to understand, universal and popular) using simpler and more effective methods.

There are several concepts of learning programming. Some concepts are preferable to others because of the differences in each person's skills in this area. The choice of the concept partly depends on the goal that is set for itself. For example, if Programming is being studied for the purpose of later teaching it to others, it will be useful to first get more theory, find out exactly why some phenomena occur, and then attach the theory to the exercises.

If you want to learn how to write codes for creating software, then you will need to study a few more specific examples, as well as master the rules and syntaxes of commands corresponding to the language.

Regardless of the concept used, an effective learning of programming requires, first of all, a general-purpose knowledge in order to be able to determine the goals and objectives of learning; after that, it is important to study the basic concept of the language in order to know the methods of solving the problem, to know how to find the shortest path and an effective way to solve the problem, to find out what has already been done in this area (for example, the existence of standard functions) to simply use them without trying to «reinvent the wheel». Finally, it is important to have an idea of the sequence of learning (that is, follow the order of acquiring knowledge about the chosen language). For example, you need to know that

there are several types of variables, and each variable can have a certain amount of space to store information, depending on the type of data that is addressed to it. This means that, logically, it would be normal to first find out about the existence of these data types, and then talk about the variable. If we look a little further, we'll be interested in even more: first learn about the meaning of data and what it looks like before we even talk about different types of data.

In the first volume, it was argued that a student who wants to learn a programming language should ask himself and answer a number of questions that have been listed. Today we tell him that whatever language is chosen, the concept of learning should remain the same as we describe in this textbook. We want to know if the tasks and examples with answers that we offer can help you develop your intellectual abilities in programming.

Continued on page 48

FR INTRODUCTION

Le monde est actuellement entrain de subir un changement où nous constatons avec lapalissade qu'il est important d'accroire aussi les connaissances acquises dans le domaine virtuel ou informatique. Car l'ordinateur devient une arme très importante qui peut basculer n'importe quelle décision d'un coté à l'autre. Cette vie virtuelle est essentiellement constituée de logiciels qui sont capables d'influencer ou de jouer un rôle important dans l'économie d'une nation. À cet effet, il est nécessaire de créer des éléments (logiciel) attrayants, utiles et nécessaires dans le domaine des technologies de l'information.

Alors, pour y parvenir, il est indispensable, d'apprendre des langages de programmation (faciles à comprendre, universels et populaires), à travers des méthodes plus faciles et effectives.

Il existe plusieurs conceptions pour l'apprentissage de la programmation. Parmi ces conceptions, certaines sont plus effectives que d'autres à cause des compétences de tout un chacun dans ce domaine. Le choix de la conception dépend en partie du but que vous vous êtes posés. Par exemple, si vous apprenez la programmation dans le but de l'enseigner plus tard aux autres, alors il sera utile d'avoir beaucoup plus de théories dans un premier temps, de savoir le pourquoi du comment, et après joindre la théorie aux exercices.

Dans le cas où vous voulez apprendre à écrire des codes pour la création des logiciels, il sera nécessaire d'apprendre avec beaucoup plus des exemples bien précis tout en maîtrisant les règles et syntaxes des commandes appropriées au langage.

Quelque soit la conception utilisée, l'apprentissage effectif de la programmation a besoin d'abord de la connaissance d'un aperçu général afin de pouvoir se positionner; après il est important d'apprendre la notion de base du langage, de savoir les techniques de résolution d'un problème, de savoir comment retrouver le chemin le plus court et effectif dans la

résolution d'un exercice, de savoir ce qui a été déjà réalisé (par exemple les fonctions natives) afin juste de les utiliser sans essayer «d'inventer une nouvelle bicyclette»; et enfin connaître le chemin à parcourir dans l'apprentissage (c'est-à-dire respecter l'ordre de la priorité d'acquisition de la connaissance du langage choisi). Par exemple, il est nécessaire de savoir qu'il existe plusieurs types de variables et que chaque variable peut avoir une quantité de place déterminée à une information selon le type de données qui lui est adressée. Ce qui veut dire que: logiquement, il sera normal de savoir d'abord l'existence des types de données et après de parler alors de la variable. Si nous allons un peu plus loin, nous serons intéressés encore plus, de savoir d'abord ce que s'est qu'une donnée, et à quoi elle ressemble avant même de parler des différents types de données.

Au premier tome, nous avons affirmé en disant qu'un apprenant qui voulait maîtriser un langage de programmation devait se poser et répondre à un certain nombre de questions que nous avons énuméré. Aujourd'hui, nous lui disons que quelque soit le langage choisi, la conception de l'apprentissage devrait rester le même que nous relatons dans ce manuel. Nous voulons bien savoir si les exercices et exemples avec résolutions que nous vous offrons, peuvent vous permettre de développer vos capacités intellectuelles dans la programmation.

Suite à la page 67

RU Глава I. ЧТЕНИЕ, ЗАПИСЬ И ДРУГИЕ ДЕЙСТВИЯ В ФАЙЛЕ

RU Введение

В программировании часто возникает необходимость чтения содержимого файлов. Считанная информация затем может храниться в переменных (см. Том 1) и анализироваться с помощью условных структур и интерактивных циклов (см. Том 1). По окончании, результаты таких анализов обычно сохраняются, записанные в файл непосредственно через Python.

Мы можем определить файл как часть памяти, в которой есть какая-либо информация. Информация, находящаяся в такой части памяти, может быть представлена различными способами: текст, исполняемый файл, изображение, видео, звук и т. д. Важно помнить, что информация хранится как последовательность битов (0 или 1). Но при необходимости эта информация будет интерпретирована таким образом, чтобы она была понятной пользователю.

Грубо говоря, мы можем разделить эти файлы на две категории:

- некодированные текстовые файлы (обычный текстовый файл или простой текстовый файл, состоящий из набора символов или строк), доступные для прочтения человеком; пример: файл Python с расширением “. ру”, файл документа с расширением “. txt” и т. д.
- файлы, закодированные в двоичной форме, состоящие из последовательности битов, организованных в пакет из восьми единиц, называемых байтами. Здесь содержимое – это не просто набор печатаемых символов, оно также не является интерпретируемым файлом; пример: файл изображения, звуковой файл, видео и т. д.

Преимущество двоичных файлов по сравнению с текстовыми файлами заключается в том, что они более компактны с точки зрения занимаемого места, а также быстрее читаются и пишутся. Для их просмотра потребуется специальный редактор или шестнадцатеричный, например: HxD, 010 Editor и многие другие.

RU I.1. Создание объекта для управления файлом

Предположим, что в памяти компьютера хранится файл (текстовый формат TXT).

Создание объекта Python, необходимого для обращения к файлу

Прежде чем у нас появится возможность читать, записывать и стирать информацию в файле, нам нужно создать определенный объект Python, который будет служить связующим звеном с файлом, расположенным на вашем компьютере (диске). Для этого нам нужно использовать функцию «***open***», которая будет принимать в качестве аргумента (абсолютную или относительную) локализацию файла, а также режим (см. таблицу 1.1) открытия (который может различаться) Если режим открытия не указан, то по умолчанию функция будет считать его как « r ». Также функция «***open***» возвращает объект типа файл для работы с ним: как только объект файла будет создан, мы можем вызвать его для чтения, записи или удаления (обновления) информации в этом файле. Для завершения, необходимо использовать функцию «*close*», чтобы прекратить соединение с файлом

Таблица 1.1 – Различные режимы работы с файлом

Ключ	**Описание ключа**
r	Позволяет открывать файл только в режиме чтения. В этом случае указатель будет помещен в начало файла. Как мы уже говорили выше, он представляет собой значение по умолчанию
w	Позволяет открывать файл только в режиме записи. Перезаписывает содержимое файла, если он уже существует, или создает новый файл в противном случае. Здесь указатель также помещается в начало файла
a	Позволяет открывать файл только в режиме добавления (в конце). После этого указатель прикрепляется к концу файла. Если файл не существует, он создается
rb	Позволяет открывать файл только для чтения в двоичном режиме, то есть данные считываются в байтах (тип «bytes»). В этом случае указатель также будет помещен в начало файла
wb	Позволяет открывать файл только в режиме записи для сохранения данных в байтах (тип "байты"). Перезаписывает содержимое файла, если оно уже существует, или создает новый файл в противном случае. Здесь указатель также помещается в начало файла
ab	Позволяет открыть файл с целью добавления определенных данных в двоичном режиме, то есть данные будут добавлены в виде байтов (типа «bytes»)
x	Он работает с версиями языка Python 3 и выше. Этот режим позволяет открыть файл с резервированием изменений, если файл уже существует. Таким образом, он не обнулит то, что уже существовало. В этом случае доступ к файлу возможен только в режиме записи. Удобно использовать этот ключ, чтобы случайно не обрезать файл, который уже существует с ключом «*a*» или «*w*»
t	Позволяет открывать файл в текстовом режиме (этот режим всегда используется по умолчанию, если он не указан).
rt	Этот ключ будет означать, что чтение будет только в текстовом режиме. Он выполняет те же функции, что и ключ « r »
wt	Этот ключ будет означать, что запись будет только в текстовом режиме. Он выполняет те же функции, что и ключ « w »

r+	Это позволяет открывать файл как в режиме чтения, так и в режиме записи одновременно. Содержимое файла не исчезает, если мы хотим добавить дополнительные данные. В таком случае файл уже должен быть прежде создан в этом режиме.
w+	Это позволяет открывать файл как в режиме чтения, так и в режиме записи одновременно. Перезаписывает содержимое файла, если оно уже существует, или создает новый файл в противном случае.
a+	Это позволяет открывать файл как в режиме чтения, так и в режиме записи одновременно. Файл уже должен существовать в этом режиме. Этот ключ используется для добавления нужной информации в конец файла.
rb+	Позволяет открывать файл в режиме чтения и в режиме записи в двоичном формате. Файл уже должен существовать в этом режиме. Содержимое файла не исчезает, если мы хотим добавить дополнительные данные
wb+	Позволяет открывать файл в режиме записи и в режиме чтения в двоичном формате. Перезаписывает содержимое файла, если оно уже существует, или создает новый файл в противном случае
ab+	Позволяет открывать файл в режиме записи и в режиме чтения в двоичном формате. Файл уже должен существовать в этом режиме. В конце файла будет добавлена необходимая информация.
x+	Этот режим позволяет открыть файл, сохраняя при этом возможность внесения изменений. Таким образом, он не создает то, что уже существует. В этом случае файл доступен как в режиме чтения, так и в режиме записи
rt+	Он выполняет те же функции, что и ключ « r+ »
wt+	Он выполняет те же функции, что и ключ « w+ »

Синтаксис функции «open» выглядит следующим образом (см. рис. 1.1):

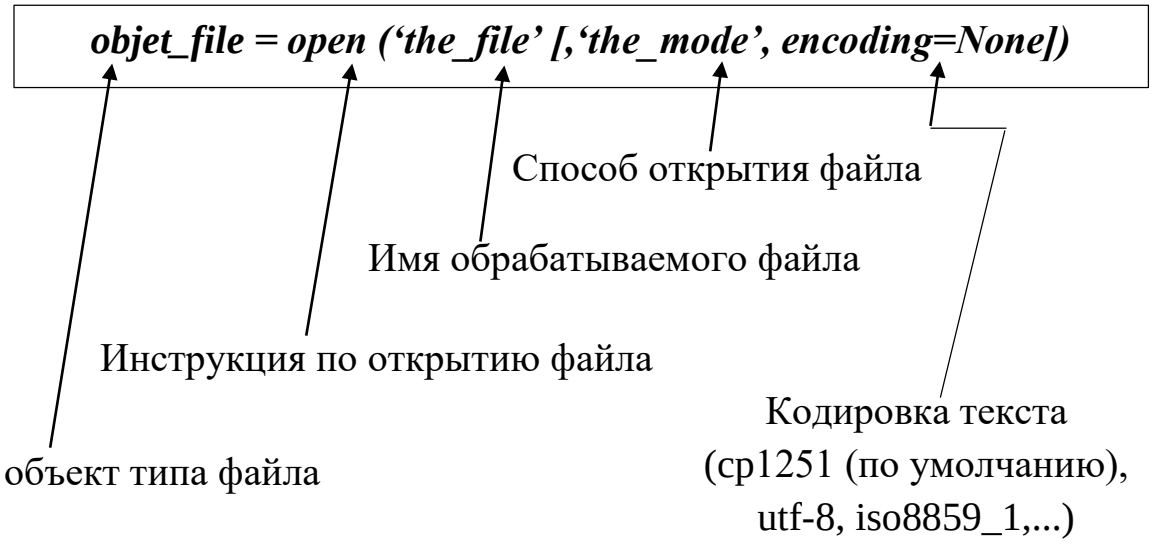


Рис. 1.1. Синтаксис инструкции «open»

Пример инструкции:

objet_file = open("the_file.txt", "r", encoding= "cp1251")

RU I.2. Чтение файла

Чтобы обрабатывать файлом, необходимо сначала узнать, как его читать.

Как читать файл

а) Для чтения мы можем использовать следующие из инструкций на выбор, в зависимости от выполняемой функции:

read() – эта инструкция позволяет, например, полностью скопировать весь текст или содержимое файла в переменную. Эта функция может иметь параметры. Например, если мы напишем инструкцию следующим образом: ***read(4)***, после выполнения указатель окажется в четвертой позиции (месте, где есть символ, включая клавишу **«Enter»**, то есть при переходе к следующей строке). Таким

образом, он будет смещен на 4 символа, оператор будет читать от начала до четвертого символа. Каждый раз мы можем двигаться вперед, увеличивая число (пока не будет достигнут конец файла). Если мы хотим узнать текущее положение указателя, мы можем использовать метод **«tell()»**.

Синтаксис инструкции « *read* »:

```
objet_file.read()
```

Упражнение 1.

Предположим, что имеются два файла с почти одинаковым содержимым (см. рис. 1.2 и 1.3), в каждом файле нет пробелов или пустых строк. После чтения каждого файла с помощью оператора *read()* определите, какова будет конечная позиция указателя после последнего символа в файле, равного «9»?

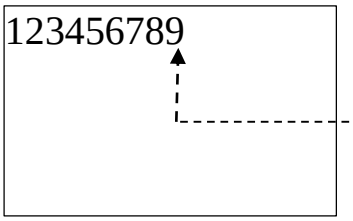


Рис. 1.2. текстовый файл с данными, помещенными в одну строку

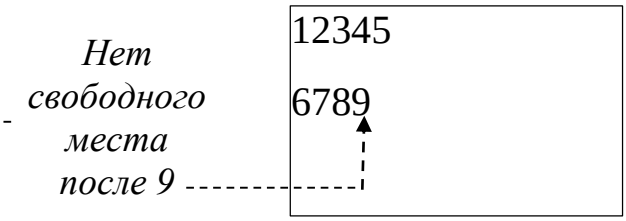


Рис. 1.3. текстовый файл с данными, расположенными в двух строках

Ответ

Как будет выглядеть код, чтобы определить эту конечную позицию указателя в файле:

```
1  with open("my_file.txt", "r") as the_file:
2      ligne = the_file.read()
3      posi = the_file.tell()
4      print('Position № = ', posi)
```

Для первого файла (см. рис. 1.1), позиция будет равна 9, а для второго файла (см. рис. 1.2), она будет равна 11.

readlines() – *оператор позволяет скопировать весь файл, чтобы поместить его в список, где каждая строка будет представлена в списке как отдельный элемент.*

Синтаксис инструкции readlines:

```
objet_file.readlines(size)
```

« size » – этот параметр является необязательным. Если количество возвращаемых байтов превышает указанное количество индексов (size), то никакие строки не будут возвращены. По умолчанию его значение равно ***-1***.

Когда мы не упоминаем индекс, то возвращаются все строки.

При использовании следующего кода результат будет таким, как показано на рисунке 1.5, тогда и только тогда, когда входной файл будет похож на файл на рисунке 1.4.

```
1  with  open(“my_file.txt”, encoding= “utf-8”) as objet_file
2      massive = objet_file.readlines()
3
```

```
Ligne 1
Ligne 2
Ligne 3
```

my_file.txt

Рис. 1.4. Текстовый файл с данными, растянулись на 3 линии

```
[‘Ligne 1\n’,  ‘Ligne 2\n’,  ‘Ligne 3’]
```

Результат

Рис. 1.5. содержимое идентификатора «massive» в коде программы

С этим результатом можно будет использовать различные методы в списке (см. Том 1) для управления данными входящего

файла. Например, вторая строка может быть удалена с помощью кода: ***massive[номер]*** *(massive[1]).*

readline() – инструкция позволяет (если это возможно) заносить определенную скопированную строку файла, дописывая ее в последовательность символов. С помощью этой функции мы также можем указать количество байтов возвращаемой строки, используя параметр «size».

«size» – как и в предыдущем, этот параметр является необязательным. Он определяет количество байтов возвращаемой строки. По умолчанию « size » имеет значение -1; Если ничего не указано или если у нас есть значение -1, то будет возвращена вся строка.

Синтаксис инструкции *readline*:

objet_file.readline(*size*)

RU I.3. Сохранение данных в файл

Как записать данные в файл

Синтаксис оператора *write:*

write() – позволяет записать указанный текст, строку или просто слово в файл. Эта функция зависит от выбранного способа написания таблицы №1.1.

Синтаксис оператора write:

objet_file.write(‘*txt_binaire’*)

или

objet_file.write("*txt_binaire"*)

txt_binaire – закодированный или не кодированный текст, который будет вставлен.

Чтобы писать в файле с определенной строки, необходимо будет при записи в файле ввести ключ « \n » на том уровне, на котором мы хотим начать новую строку.

Упражнение 1

Сохраните эту фразу «текст, строку или слово» в файле «*my_file.txt*».

```
1  with open("my_file.txt", "w", encoding="cp1251") as the_file:
2      a_enregistrer = "текст, строку или слово \n"
3      the_file.write(a_enregistrer)
4
5  # Список не может быть сохранен с помощью функции
6  # «write», вы должны использовать другую инструкцию.
7  # Кодировка по умолчанию «cp1251», т.е. encoding="cp1251".
```

writelines() –- эта функция позволяет брать элементы из списка и сохранять их в файле. Она также может сохранять данные, которых нет в списке. Синтаксис:

objet_file.writelines(*spisok*)

spisok – здесь мы вставляем информацию (простые или двоичные тексты) в список, который будет храниться в файле.

Упражнение 2

Имеется список (spisok), содержащий 3 элемента: строка №1, строка № 2 и строка №3. Необходимо сохранить эти данные в файле « *my_file.txt* ».

Решение:

```
1  with open("my_file.txt", "w", encoding="utf-8") as the_file:
2      spisok = ["строка №1\n", " строка №2\n", " строка №3"]
3      the_file.writelines(spisok)
4
5  # Кодирование может быть удалено из кода без каких-либо
6  # ошибок. Результат выполнения приведен на рисунке 1.6.
```

Ответ :

Строка №1
Строка №2
Строка №3

Рис. 1.6. содержимое файла «my_file.txt» после выполнения кодов

Мы также можем использовать функцию «print()» не для отображения данных на экране компьютера, а для хранения любой информации в файле. Синтаксис этой команды будет выглядеть следующим образом:

print(*txt_binaire*, file = *objet_file*)

txt_binary – закодированный или некодированный текст, который будет вставлен; *objet_file* – напоминаем, что это объект типа файла.

Что означает, что :

objet_file.write("*txt_binaire*") ↔
print("*txt_binaire*", file = *objet_file*)

RU I.4. Заключительные действия над файлом после его использования

Закрытие файла после использования

Когда открываем какой-либо файл для внесения изменений или когда создаем файл для его использования, в конце процедуры очень важно закрыть его с помощью инструкции «***close()***». Эта функция может не использоваться, если мы обращаемся к файлу с помощью инструкции «with». в случае, если она не используется, важно сопровождать ваши коды, используя инструкции ***Try – Except – Else – Finally***.

Синтаксис :

```
objet_file. close()
```

Удаление файла или каталога

Эта часть не входит в число шагов, которые необходимо выполнить для чтения или записи информации в файл. Но, учитывая, что мы уже знаем, как создавать и записывать информацию внутри файла, не будет лишним также узнать, как мы можем удалить определенные файлы на диске. Шаги следующие:

а) необходимо импортировать модуль операционной системы, который позволит реализовать его:

```
import os
```

б) необходимо проверить, существует ли файл или каталог с помощью этого кода:

```
if os.path.exists("my_file.txt"):
```

в) удалить файл или каталог, если он существует

```
os.remove("my_file.txt")
```

```
os.rmdir("MY_FOLDER")
```

Задача 1 с решением

Напишите различные коды, позволяющие стереть файл или каталог с любого диска, где известен путь.

Ответ 1-Удалить файл без инструкций Try, ...

```
import os

if os.path.exists("my_file.txt"):
    os.remove("my_file.txt")
    print("Файл был удален")
else:
    print("Файл не существует, его невозможно удалить.")

```

Ответ 2 – Удалить файл также с помощью инструкции Try ...

```
import os

try:
    os.remove("my_file.txt")
    print("Файл был удален")
except FileNotFoundError:
    print("Файл не существует, его невозможно удалить.")

```

Ответ 3 – Удалить папку без инструкций Try, ...

```
import os

if os.path.exists("my_file.txt"):
    os.remove("my_file.txt")
    print("Папка была удалена")
else:
    print("Папка не существует, её невозможно удалить.")

```

Ответ 4 – Удалить папку также с помощью инструкции Try ...

```
import os

try:
    os.rmdir("my_file.txt")
    print("Папка была удалена")
except FileNotFoundError:
    print("Папка не существует, её невозможно удалить ")

```

Задача 2 с решением

Используйте всю полученную информацию (как читать, записывать и удалять информацию в файле) для чтения (построчно) различными способами данных из любого текстового файла:

а) используйте функцию *readline()* для чтения текстового файла построчно;

б) используйте функцию *readlines()* для чтения строк и преобразования их в список;

в) чтение файла по строкам из объекта «File» без функций *readline()* и *readlines()*;

d) чтение файла по строкам из объекта «File» с помощью функции «readline()» и оператора «while» без каких-либо условий «if»;

e) чтение файла по строкам из объекта «File» с помощью функции «readline()» и с условием «if»;

Ответ 1. Используйте функцию *readline()* для чтения текстового файла построчно.

```
1  the_file = open("c:\\work\\my_file.txt", "r")
2  k=0
3  while True:
4      ligne = the_file.readline()
5      if not ligne:
6          break
7      k+=1
8      print('Строка №{0}'.format(k),ligne.strip())
9
10 le_fichier.close()
```

Комментарий к предыдущим кодам
#1 мы получаем объект, который является файлом, в режиме чтения. *#2 Инициализации счетчика.* *#3 Начало цикла-выполняется до тех пор, пока не найдет прерывание в цикле.* *#4 прочитайте строку и запишите содержимое в переменную с именем «ligne».* *#6 мы прерываем цикл, если строка пуста (#5).* *#7 давайте увеличим счетчик на один шаг, этот будет номером следующей выходной строки.* *#8 выводим номер строки вместе с ключевым словом «Строка» и соответствующей ему строкой.* *#10 Закрытие файла*

Ответ 2. Используйте функцию *readlines()* для чтения строк и преобразования их в список

```
1  the_file = open("c:\\work\\my_file.txt", "r")
2  lines = the_file.readlines()
3  print('Текст, прочитанный из списка: ',lines)
4  k = 1
5  for ligne in lines:
6      print('Строка №{0}'.format(k),ligne.strip())
7      k+=1
8
9  the_file.close()
```

Этой части может и не быть.

Комментарий к предыдущим кодам
#1 Получаем объект, который является файлом, в режиме чтения. *#2 Читаем все строки с функцией readlines().* *#3 Вытаскиваем прочитанный текст, он будет в списке.* *#4 Инициализации счетчика.* *#5 Открываем цикл и выполняем процесс итерации по каждой строке.* *#6 Выводим номер строки вместе с ключевым словом «Строка» и соответствующей ему строкой.* *#7 давайте увеличим счетчик на один шаг, этот будет номером следующей выходной строки.* *#9 Закрытие файла.* *Штрихованная часть может отсутствовать в кодах, поскольку строка № 2 считывает информацию из файла, чтобы поместить ее в список. Из списка мы можем делать с ними все, что хотим.*

Ответ 3. чтение файла по строкам из объекта файла без функций *readline()* и *readlines()*.

```
1  k=1
2  with open("c:\\work\\my_file.txt", "r") as the_file:
3      for ligne in the_file:
4          print('Строка №{0}'.format(k),ligne.strip())
5          k+=1
6
7  the_file.close()
```

Комментарий к предыдущим кодам
#1 Инициализации счетчика. *#2 Получаем объект, который является файлом, в режиме чтения (данные из файла сохраняются в "the_file". Они помещаются в список как массив данных).* *#3 Открываем цикл и выполняем процесс итерации по каждой строке.* *#4 Выводим номер строки вместе с ключевым словом «Строка» и соответствующей ему строкой.* *#5 Необходимо увеличить счетчик на один шаг, этот будет номером следующей выходной строки.* *#7 Закрытие файла – процесс не является обязательным из - за наличия инструкции «with».*

Ответ 4. Чтение файла по строкам из объекта файла с помощью функции «readline()» и оператора « while » без условия « if ».

```
1  k=0
2  with open("c:\\work\\my_file.txt", "r") as the_file:
3      ligne = the_file.readline()
4      while ligne:
5          k+=1
6          print('Строка №{0}'.format(k), ligne.strip())
7          # или
8          print('Строка №{0}'.format(k), ligne.strip(), end="  ")
9          ligne = the_file.readline()
10 le_fichier.close()
```

Комментарий к предыдущим кодам
#1 Инициализации счетчика. *#2 Ожидаем появления объекта типа файл, в режиме чтения (записи о файле сохраняются в "the_file". Они расположены списком, как массив данных).* *#3 Прочитать строку и записать содержимое в переменную с именем «ligne»* *#4 Начало цикла – выполняется каждый раз, когда он находит новую строку в файле* *#5 Необходимо увеличить счетчик на 1 шаг, это будет указанием на номер следующей выходной линии* *#6 Выводим номер строки вместе с ключевым словом «Строка» и соответствующей ему строкой* *#8 Выводим все строки с номерами в одной строке, добавляя команду « end=" " » в конце* *#9 Прочитать строку и записать содержимое в переменную с именем «ligne»* *#10 Закрытие файла – это не является обязательным из-за инструкции «with».*

Ответ 5. Чтение файла по строкам из объекта File с помощью функции « readline() » и с условием « if ».

```
1  k=0
2  with open("c:\\work\\my_file.txt", "r") as the_file:
3      while True:
4          ligne = the_file.readline()
5          if not ligne:
6              break
7          k+=1
8          print('Строка №{0}'.format(k), ligne.strip(), end="  ")
9          #print('Строка №{0}'.format(k), ligne.strip())
10 the_file.close()
```

Комментарий к предыдущим кодам
#1 Инициализации счетчика. *#2 Получаем объект, который является файлом, в режиме чтения (данные в файле сохраняются в "the_file". Они помещаются в список как массив данных.)* *#3 Начало цикла: выполняется до тех пор, пока не найдет прерывание в цикле.* *#4 Прочитать строку и записать содержимое в переменную с именем « ligne ».* *#5 Проверить, нет ли больше строк для чтения.* *#6 Перейти, в случае, если строка пуста – прервать цикл.* *#7 Увеличим счетчик на один шаг, результатом будет номер следующей выходной строки.* *#8 Необходимо, каждый раз выводить полученную строку с меткой и номером, добавляя в конец команду « end=" " ».* *#9 то же, что и в предыдущем, за исключением отсутствия команды « end=" " »* *#10 закрытие файла – оно не является обязательным из - за наличия инструкции «with».*

В заключение мы можем посоветовать вам чаще и предпочтительнее использовать инструкцию « with » при работе с файлами, что гарантирует автоматически их закрытие в случае появления ошибки. Эта команда хорошо работает с исключениями, которые могут возникнуть. Мы можем использовать инструкцию « with », также как мы можем использовать блок «try ... finally» для борьбы с ошибками, но наличие «with» будет намного эффективнее при реализации большой программы.

Учитывая, что мы уже знаем, как читать и записывать определенную информацию в файл, в дальнейшем необходимо научиться анализировать и вносить изменения в уже существующие тексты.

RU I.5. Вопросы о понимании курса

1) В чем преимущество и сложность использования файлов, закодированных в "двоичной" форме, по сравнению с не кодированными текстовыми файлами?

2) Какой вид или способ обращения мы будем использовать в файле « my_file2.txt » после выполнения следующей команды:

with open(“my_file.txt", encoding= "koi8_r") в качестве_файла

и после записи двух цифр с помощью функции write()?

3) Почему мы должны закрывать файл после его открытия для различных манипуляций?

4) Каким будет тип кодировки текста после выполнения этой команды:

with open(“my_file1.txt”, “w+”) as objet_file,

или мы увидим ошибку, прочитав строку в файле «my_file1.txt» с помощью функции « readline() »?

5) В чем разница между следующими функциями: read(), readline(), readlines() ?

6) В чем разница между следующими функциями: write(), writeline() ?

7) Можем ли мы обойтись без функций readline() и readlines() для чтения любого текстового файла ?

8) Какова полезность инструкции «with» при работе с файлами ?

Продолжение на странице 85

EN CHAPTER I. READING, WRITING AND OTHER ACTIONS IN THE FILE

EN Introduction

In programming, there is often a need to read the contents of files. The read information can then be stored in variables (see Volume 1) and analyzed using conditional structures and interactive loops (see Volume 1). At the end, the results of such analyses are usually saved, written to a file directly through Python.

We can define a file as a part of memory in which there is any information. The information stored in such a part of memory can be represented in various ways: text, executable file, image, video, sound, etc. It is important to remember that the information is stored as a sequence of bits (0 or 1). But if necessary, this information will be interpreted in such a way that it is understandable to the user.

Roughly speaking, we can divide these files into two categories:

- uncoded text files (a plain text file or a simple text file consisting of a set of characters or strings) that can be read by a person; example: a Python file with the extension ". *py*", a document file with the extension ". *txt*", etc.
- files encoded in binary form, consisting of a sequence of bits organized into a packet of eight units called bytes. Here, the content is not just a set of printable characters, it is also not an interpreted file; example: image file, audio file, video, etc.

The advantage of binary files compared to text files is that they are more compact in terms of space, as well as faster to read and write. To view them, you will need a special editor or hexadecimal, for example: HxD, 010 Editor and many others.

EN I.1. Creating an object to manage the file

Suppose that a file (text format TXT) is stored in the computer's memory.

Creating a Python object required to access the file

Before we can read, write, and erase information in a file, we need to create a specific Python object that will serve as a link to the file located on your computer (disk). To do this, we need to use the «***open***» function, which will take as an argument the (absolute or relative) localization of the file, as well as the opening mode (see Table 1.1) (which may vary). If the opening mode is not specified, then by default the function will consider it as « r ». Also, the «***open***» function returns a file type object to work with: as soon as the file object is created, we can call it to read, write or delete (update) information in this file. To complete, you need to use the «*close*» function to terminate the connection to the file.

Table 1.1 – Various modes of working with the file.

Key	Key description
r	Allows you to open a file in read-only mode. In this case, the pointer will be placed at the beginning of the file. As we said above, it is the default value
w	Allows you to open a file only in recording mode. Overwrites the contents of the file if it already exists, or creates a new file otherwise. Here the pointer is also placed at the beginning of the file
a	Allows you to open a file only in append mode (at the end). After that, the pointer is attached to the end of the file. If the file does not exist, it is created
rb	Allows you to open a file read-only in binary mode, that is, data is read in «bytes». In this case, the pointer will also be placed at the beginning of the file

Key	Key description
wb	Allows you to open a file only in write mode to save data in «bytes». Overwrites the contents of the file if it already exists, or creates a new file otherwise. Here the pointer is also placed at the beginning of the file.
ab	Allows you to open a file in order to add certain data in binary mode, that is, the data will be added as «bytes»
x	It works with Python versions 3 and higher. This mode allows you to open a file with backup changes if the file already exists. Thus, it will not reset what already existed. In this case, the file can only be accessed in the recording mode. It is convenient to use this key so as not to accidentally crop a file that already exists with the key «*a*» or «*w*»
t	Allows you to open a file in text mode (this mode is always used by default if it is not specified)
rt	This key will mean that reading will only be in text mode. It performs the same functions as the « r » key.
wt	This key will mean that the recording will be in text mode only. It performs the same functions as the « w » key.
r+	This allows you to open the file in both read and write mode at the same time. The contents of the file do not disappear if we want to add additional data. In this case, the file must have already been created in this mode before
w+	This allows you to open the file in both read and write mode at the same time. Overwrites the contents of the file if it already exists, or creates a new file otherwise
a+	This allows you to open the file in both read and write mode at the same time. The file should already exist in this mode. This key is used to add the necessary information to the end of the file
rb+	Allows you to open a file in read mode and in write mode in binary format. The file should already exist in this mode. The contents of the file do not disappear if we want to add additional data
wb+	Allows you to open a file in write mode and in read mode in binary format. Overwrites the contents of the file if it already exists, or creates a new file otherwise

Key	Key description
ab+	Allows you to open a file in write mode and in read mode in binary format. The file should already exist in this mode. The necessary information will be added at the end of the file.
x+	This mode allows you to open a file while still being able to make changes. So it doesn't create something that already exists. In this case, the file is available in both read and write mode.
rt+	It performs the same functions as the « r+ » key
wt+	It performs the same functions as the « w+ » key

The syntax of the «open» function looks like this (see Figure 1.1):

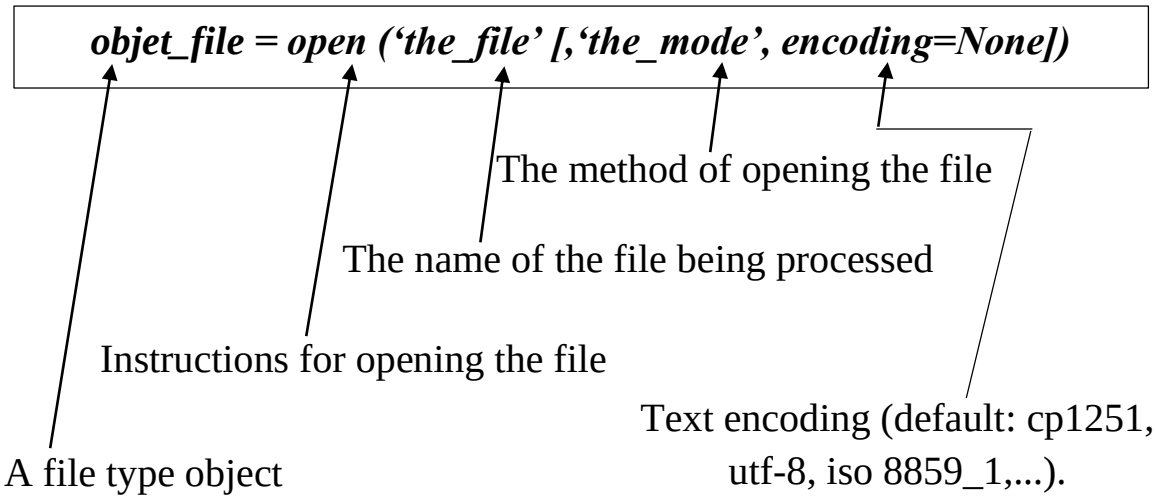


Fig. 1.1. Syntax of the «open» instruction.

Example instructions:

objet_file = open("the_file.txt", "r", encoding= "cp1251")

EN I.2. Reading a file

To process a file, you must first learn how to read it.

How to read a file?

a) For reading, we can use the following instructions to choose from, depending on the function being performed:

read() – this instruction allows, for example, to completely copy the entire text or contents of a file to a variable. This function can have parameters. For example, if we write an instruction like this: *read(4)*, after execution, the pointer will be in the fourth position (the place where there is a symbol, including the **«Enter»** key, that is, when moving to the next line). Thus, it will be shifted by 4 characters, the operator will read from the beginning to the fourth character. Each time we can move forward, increasing the number (until the end of the file is reached). If we want to know the current position of the pointer, we can use the **«tell()»** method.

Syntax of the « *read »* statement:

objet_file.read()

Exercise 1.

Suppose there are two files with almost identical contents (see fig. 1.2 and 1.3), there are no spaces or empty lines in each file. After reading each file using the *read()* operator, determine what will be the final position of the pointer after the last character in the file equal to «9»?

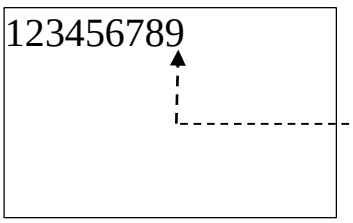


Fig. 1.2. A text file with data placed in one line.

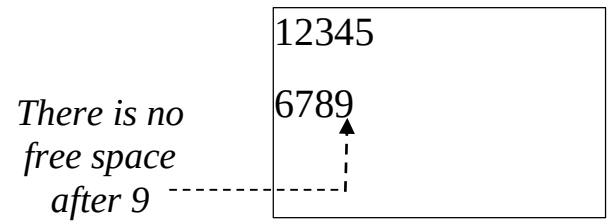


Fig. 1.3. A text file with data arranged in two lines.

Answer

What will the code look like to determine this final pointer position in the file:

```
1  with open("my_file.txt", "r") as the_file:
2      ligne = the_file.read()
3      posi = the_file.tell()
4      print('Position № = ', posi)
```

For the first file (see Fig. 1.1), the position will be 9, and for the second file (see Fig. 1.2), it will be 11.

readlines() – the operator allows you to copy the entire file to put it in a list, where each line will be presented in the list as a separate element.

Syntax of the *readlines* statement:

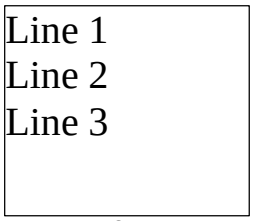
objet_file.readlines(*size*)

« size » – this parameter is optional. If the number of bytes returned exceeds the specified number of indexes (size), then no rows will be returned. By default, its value is ***-1***.

When we don't mention the index, all rows are returned.

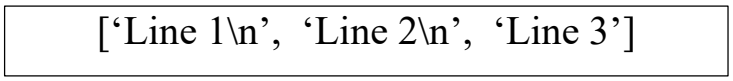
When using the following code, the result will be as shown in Figure 1.5, if and only if the input file is similar to the file in Figure 1.4.

```
1  with  open(“my_file.txt”, encoding= “utf-8”) as objet_file
2      massive = objet_file.readlines()
3
```



my_file.txt

Figure 1.4. A text file with data stretched over 3 lines



Result

Figure 1.5. the contents of the "massive" identifier in the program code

With this result, it will be possible to use the various methods in the list (see Volume 1) to manage the incoming file data. For example, the second line can be deleted using the code: massive[number] (massive[1]).

readline() – the instruction allows (if possible) to enter a certain copied line of the file, appending it to a sequence of characters. With this function, we can also specify the number of bytes of the returned string using the «size» parameter.

«size» – as in the previous one, this parameter is optional. It determines the number of bytes of the returned string. By default, «size» has a value of -1; If nothing is specified or if we have a value of -1, the entire string will be returned.

Syntax of the ***readline*** instruction:

objet_file.readline(*size*)

EN I.3. Saving data to a file

How to write data to a file?

The syntax of the ***write*** statement:

write() – allows you to write the specified text, string, or just a word to a file. This function depends on the chosen method of writing table №1.1.

The syntax of the ***write*** statement:

objet_file.write(‘*txt_binaire’*)

or

objet_file.write("*txt_binaire"*)

txt_binaire – encoded or uncoded text to be inserted.

To write in a file from a certain line, it will be necessary to enter the key « \n » at the level at which we want to start a new line when writing in the file.

Упражнение 1.

Сохраните эту фразу «текст, строку или слово» в файле «*my_file.txt*».

Exercise 1.

Save this phrase «text, string or word» in a file «*my_file.txt*».

```
1  with open("my_file.txt", "w", encoding="cp1251") as the_file:
2      a_enregistrer = " text, string or word \n"
3      the_file.write(a_enregistrer)
4
5  # The list cannot be saved using the "write" function,
6  # you must use a different instruction.
7  # The default encoding is "cp1251", i.e. encoding="cp1251".
```

writelines() *-- this function allows you to take items from a list and save them to a file. It can also save data that is not in the list.*

Syntax:

objet_file.writelines(*spisok*)

spisok – here we insert information (plain or binary texts) into a list that will be stored in a file.

Упражнение 2

Имеется список (spisok), содержащий 3 элемента: строка №1, строка № 2 и строка №3. Необходимо сохранить эти данные в файле « *my_file.txt* ».

Solution

Exercise 2

There is a list (spisok) containing 3 elements: row №1, row №2 and row №3. It is necessary to save this data in a file « *my_file.txt* ».

Solution:

```
1  with open("my_file.txt", "w", encoding="utf-8") as the_file:
2      spisok = ["row №1\n", "row №2\n", "row №3"]
3      the_file.writelines(spisok)
4
5  # Coding can be removed from the code without any errors.
6  # The result of the execution is shown in Figure 1.6.
```

Answer:

```
row №1
row №2
row №3
```

Figure 1.6. File contents *«my_file.txt»* after executing the codes

We can also use the «print()» function not to display data on a computer screen, but to store any information in a file. The syntax of this command will look like this:

```
print(txt_binaire, file = objet_file)
```

txt_binary – encoded or non–encoded text to be inserted; objet_file – we remind you that this is a file type object.

Which means that:

objet_file.write("*txt_binaire*") ↔ print("*txt_binaire*", file = *objet_file*)

EN I.4. The final actions on the file after its use

Closing the file after use

When we open a file to make changes or when we create a file to use it, at the end of the procedure it is very important to close it using the «***close()***» instruction. This function may not be used if we access the file using the "with" statement. in case it is not used, it is important to accompany your codes using ***Try – Except – Else – Finally*** instructions.

Syntax :

objet_file. close()

Deleting a file or directory

This part is not among the steps that need to be performed to read or write information to a file. But, considering that we already know how to create and write information inside a file, it would not be superfluous to also find out how we can delete certain files on disk. The steps are as follows:

a) you need to import an operating system module that will allow you to implement it:

import os

b) you need to check if a file or directory exists using this code:

```
if os.path.exists("my_file.txt"):
```

c) delete the file or directory if it exists:

```
os.remove("my_file.txt")
```

```
os.rmdir("MY_FOLDER")
```

Exercise 1 with a solution

Write various program codes that allow you to erase a file or directory from any disk where the path is known.

Answer 1 – Delete the file without ***Try*** instructions, ...

```
1  import os
2
3  if os.path.exists("my_file.txt"):
4      os.remove("my_file.txt")
5      print("The file has been deleted")
6  else:
7      print("The file does not exist, it cannot be deleted.")
8
```

Answer 2 – Delete the file also using the ***Try*** ... instructions

```
1  import os
2
3  try:
4      os.remove("my_file.txt")
5      print("The file has been deleted")
6  except FileNotFoundError:
7      print("The file does not exist, it cannot be deleted.")
8
```

Answer 3 – Delete the folder without ***Try*** ... instructions

```
import os

if os.path.exists("my_file.txt"):
    os.remove("my_file.txt")
    print("The folder has been deleted")
else:
    print("The folder does not exist, it cannot be deleted.")

```

Ответ 4 – Удалить папку также с помощью инструкции Try ...

```
import os

try:
    os.rmdir("my_file.txt")
    print("The folder has been deleted")
except FileNotFoundError:
    print("The folder does not exist, it cannot be deleted")

```

Exercise 2 with the solution

Use all the information received (how to read, write and delete information in a file) to read (line by line) data from any text file in various ways:

a) use the *readline()* function to read a text file line by line;

b) use the *readline()* function to read strings and convert them to a list;

c) reading a file by lines from the «File» object without the *readline()* and *readlines()* functions;

d) reading a file by lines from the "File" object using the «readline()» function and the «while» operator without any «if» conditions;

f) reading a file by lines from the "File" object using the «readline()» function and with the «if» condition;

Answer 1. Use the *readline()* function to read the text file line by line.

```
1  the_file = open("c:\\work\\my_file.txt", "r")
2  k=0
3  while True:
4      line = the_file.readline()
5      if not line:
6          break
7      k+=1
8      print('Row №{0}'.format(k), line.strip())
9
10 le_fichier.close()
```

Comment on previous codes
#1 we get the object, which is a file, in read mode. *#2 Initializing the counter.* *#3 Start of the loop-runs until it finds an interrupt in the loop.* *#4 Read the line and write the contents to a variable named «line».* *#6 We break the loop if the line is empty (#5).* *#7 Let's increase the counter by one step, this will be the number of the next output line.* *#8 Output the line number along with the keyword «Row» and the corresponding string.* *#10 Closing a file.*

Answer 2. Use the *readline()* function to read strings and convert them to a list.

```
1  the_file = open("c:\\work\\my_file.txt", "r")
2  lines = the_file.readlines()
3  print('The text read from the list: ',lines)
4  k = 1
5  for line in lines:
6      print('Row №{0}'.format(k),line.strip())
7      k+=1
8
9  the_file.close()
```

This part may not be there.

Comment on previous codes
#1 We get the object, which is a file, in read mode. *#2 Read all the lines with the «readline()» function.* *#3 Pull out the read text, it will be in the list.* *#4 Initializing the counter.* *#5 Open the loop and perform the iteration process on each line.* *#6 Output the line number along with the keyword «Row» and the corresponding line.* *#7 let's increase the counter by one step, this will be the number of the next output line.* *#9 Closing the file.* *The hatched part may be missing from the codes, because line №2 reads information from the file to put it in the list. From the list, we can do whatever we want with them.*

Answer 3. Reading a file by lines from a file object without the *readline()* and *readlines()* functions.

```
1  k=1
2  with open("c:\\work\\my_file.txt", "r") as the_file:
3      for line in the_file:
4          print('Row №{0}'.format(k),line.strip())
5          k+=1
6
7  the_file.close()
```

Comment on previous codes
#1 Counter initialization. *#2 We get the object, which is a file, in read mode (data from the file is saved in «the_file». They are placed in the list as an array of data).* *#3 Open the loop and perform the iteration process for each row.* *#4 Output the line number along with the keyword «Line» and the corresponding line.* *#5 It is necessary to increase the counter by one step, this will be the number of the next output line.* *#7 Closing the file – the process is optional due to the presence of the «with» instruction.*

Answer 4. Reading a file by lines from a file object using the «readline()» function and the « while » operator without the « if » condition.

```
1  k=0
2  with open("c:\\work\\my_file.txt", "r") as the_file:
3     line = the_file.readline()
4     while line:
5        k+=1
6        print('Row №{0}'.format(k), line.strip())
7        # or
8        print('Row №{0}'.format(k), line.strip(), end="  ")
9        line = the_file.readline()
10 le_fichier.close()
```

Comment on previous codes
#1 Counter initialization. *#2 We are waiting for the appearance of a file type object, in read mode (file entries are saved in «the_file». They are arranged in a list, like an array of data).* *#3 Read the string and write the contents to a variable named «line"* *#4 Start of the loop – executed every time it finds a new line in the file* *#5 It is necessary to increase the counter by 1 step, this will be an indication of the number of the next output line* *#6 Output the line number along with the keyword «Row» and the corresponding line* *#8 Output all lines with numbers in one line, adding the command « end=" " » at the end* *#9 Read the string and write the contents to a variable named «line»* *#10 Closing the file is optional because of the «with» statement.*

Answer 5. Reading a file by lines from the File object using the « readline() » function and with the « if » condition.

```
1  k=0
2  with open("c:\\work\\my_file.txt", "r") as the_file:
3      while True:
4          line = the_file.readline()
5          if not line:
6              break
7          k+=1
8          print('Row №{0}'.format(k), line.strip(), end="  ")
9          #print('Row №{0}'.format(k), line.strip())
10 the_file.close()
```

Comment on previous codes
#1 Counter initialization. *#2 We get the object, which is a file, in read mode (the data in the file is saved in «the_file». They are placed in the list as an array of data.)* *#3 Start of the loop: runs until it finds an interrupt in the loop.* *#4 Read the string and write the contents to a variable named «line».* *#5 Check if there are no more lines to read.* *#6 Go, in case the line is empty – break the loop.* *#7 Increase the counter by one step, the result will be the number of the next output line.* *#8 It is necessary, each time, to output the received string with a label and a number, adding the command « end=" " » to the end.* *#9 Is the same as in the previous one, except for the absence of the « end=" " » command* *#10 Closing the file is optional due to the presence of the «with» instruction.*

In conclusion, we can advise you to use the « with » instruction more often and preferably when working with files, which guarantees that they are automatically closed in case of an error. This command works well with exceptions that may occur. We can use the « with » statement, just as we can use the block «try ... finally» to deal with errors, but having « with » will be much more effective when implementing a large program.

Given that we already know how to read and write certain information to a file, in the future it is necessary to learn how to analyze and make changes to existing texts.

.

EN I.5. Questions about understanding the course

1) What is the advantage and complexity of using files encoded in "binary" form compared to non-encoded text files?

2) What kind or method of treatment will we use in the file «my_file2.txt» after executing the following command:

with open("my_file.txt", encoding= "koi8_r") as a file

and after writing two digits using the ***write()*** function?

3) Why should we close the file after opening it for various manipulations?

4) What will be the type of text encoding after executing this command:

with open(“my_file1.txt”, “w+”) as objet_file,

or we will see an error by reading a line in the file «my_file1.txt» using the « readline() » function?

5) What is the difference between the following functions: *read(), readline(), readlines()* ?

6) What is the difference between the following functions: *write(), writeline() ?*

7) Can we do without the *readline()* and *readlines()* functions to read any text file?

8) What is the usefulness of the «with» instruction when working with files?

Continued on page 102


FR Chapitre I. LIRE, ÉCRIRE ET AUTRES ACTIONS DANS LE FICHIER

FR Introduction

En programmation, il est souvent nécessaire de lire le contenu de fichiers. Les informations lues peuvent alors être stockées dans des variables (cf. Tom 1) et analysées par l'utilisation de structures conditionelles et de boucles itératives (cf. Tom 1). Finalement, on sauvegarde généralement les résultats de ces analyses en les écrivant directement via Python dans un fichiers.

Nous pouvons définir le fichier comme étant une partie de la mémoire qui possède une information quelconque. L'information qui se trouve dans cette partie de la mémoire peut être représentée de différentes manières: un texte, un exécutable, une image, une vidéo, un son, etc. Nous devons savoir que les informations sont stockées de la même manière comme étant une séquence de bits (0 ou1). Ces informations seront plus tard interprétées de manière à être compréhensible pour l'utilisateur.

En grosso-modo, nous pouvons scinder ces fichiers en deux catégories:

- les fichiers non codés en forme de texte (fichier texte brut ou fichier texte simple constitué d'une suite ou chaîne de caractères ou de lignes) compréhensible à l'œil nu; exemple: fichier Python à extension ".py", fichier document à extension ".txt" etc.

- les fichiers codés en forme binaire constitué d'une séquence de bits organisé en paquet de huit, appelés octets (bytes). Ici, le contenu n'est pas une simple suite de caractères imprimables, il n'est non plus un fichier interprétable; exemple: fichier image, fichier son, vidéo, etc.

L'avantage des fichiers binaires, par rapport aux fichiers textes est qu'ils sont plus compacts en termes d'espace occupé et également plus rapide à lire et écrire. Pour les visualiser, il faudra un éditeur special ou alors un éditeur hexadécimal deja l'exemple de: HxD, 010 Editor et plusieurs d'autres.

FR I.1. Création de l'objet pour la manipulation d'un fichier

Considérons que nous avons un fichier quelconque (format TXT) stocké en mémoire sur l'ordinateur.

Création de l'objet Python pour s'adresser au fichier

Avant que nous puissions avoir la possibilité de lire, écrire et effacer une information dans un fichier, nous devons créer l'objet Python qui pourra servir de lien avec le fichier localisé dans votre ordinateur (disque). Pour cela, nous devons utiliser la fonction «***open***», qui prendra en argument la localisation (absolue ou relative) du fichier ainsi que le mode (voir tableau 1.1) d'ouverture (qui peut différencier). Si le mode d'ouverture n'est pas indiqué, alors la fonction le considéra comme « r ». La fonction « open » renvoie alors un objet de type fichier, et une fois que l'objet fichier est créé, nous pouvons appeler ses méthodes pour lire, écrire ou effacer (mettre à jour) les informations dans ce fichier. A la fin de l'utilisation, la fonction «close» permettra d'achever la connexion avec le fichier.

Tableau 1.1 – Les différentes modes d'ouverture

La Clé	Description de la clé
r	Permet d'ouvrir un fichier uniquement en mode en lecture. Dans ce cas, le pointeur sera placé au début du fichier. Comme nous l'avons dit un peu plus haut, il représente la valeur par défaut
w	Permet d'ouvrir un fichier uniquement en mode écriture. Écrase le contenu du fichier s'il existe déjà, ou alors crée un nouveau fichier dans le cas contraire. Ici, le pointeur est également placé au début du fichier
a	Permet d'ouvrir un fichier uniquement en mode ajout (à la fin). Après cela, le pointeur est fixé à la fin du fichier. Si le fichier n'existe pas, alors il est créé

La Clé	Description de la clé
rb	Permet d'ouvrir un fichier uniquement pour lecture en mode binaire c'est-à-dire que les données sont lues sous formes d'octets (type « bytes »). Dans ce cas, le pointeur sera également placé au début du fichier
wb	Permet d'ouvrir un fichier uniquement en mode écriture pour enregistrer des données sous formes d'octets (type « bytes »). Écrase le contenu du fichier s'il existe déjà, ou alors crée un nouveau fichier dans le cas contraire. Ici, le pointeur est également placé au début du fichier
ab	Permet d'ouvrir le fichier dans le but d'ajouter certaines données en mode binaire c'est-à-dire que les données seront ajoutées sous formes d'octets (type « bytes »)
x	Il est fonctionnel avec les versions 3 et plus du langage Python. Ce mode permet d'ouvrir le fichier avec des réservations de modification s'il existe déjà. Donc, il ne crée pas ce qui a déjà été créé. Dans ce cas, le fichier est accessible uniquement en mode écriture. Il est commode d'utiliser cette clé pour éviter de tronquer accidentellement un fichier qui existe déjà avec la clé «*a*» ou «*w*»
t	Permet d'ouvrir le fichier en mode texte (ce mode est toujours utilisé par défaut, lorsqu'il n'est pas explicitement spécifié)
rt	Cette clé signifiera que la lecture se ferra en mode texte. Elle a les mêmes fonctions que la clé « r »
wt	Cette clé signifiera que l'écriture se ferra en mode texte. Elle a les mêmes fonctions que la clé « w »
r+	Il permet d'ouvrir le fichier en mode lecture et en mode écriture à la fois. Le contenu du fichier ne disparaît pas si nous voulons ajouter d'autres données. Le fichier doit déjà exister en ce mode
w+	Il permet d'ouvrir le fichier en mode lecture et en mode écriture à la fois. Écrase le contenu du fichier s'il existe déjà, ou alors crée un nouveau fichier dans le cas contraire
a+	Il permet d'ouvrir le fichier en mode lecture et en mode écriture à la fois. Le fichier doit déjà exister en ce mode. Il ajoute l'information voulue à la fin du fichier

La Clé	Description de la clé
rb+	Permet d'ouvrir le fichier en mode lecture et en mode écriture au format binaire. Le fichier doit déjà exister en ce mode. Le contenu du fichier ne disparaît pas si nous voulons ajouter d'autres données
wb+	Permet d'ouvrir le fichier en mode écriture et en mode lecture au format binaire. Écrase le contenu du fichier s'il existe déjà, ou alors crée un nouveau fichier dans le cas contraire
ab+	Permet d'ouvrir le fichier en mode écriture et en mode lecture au format binaire. Le fichier doit déjà exister en ce mode. Il ajoute l'information voulue à la fin du fichier
x+	Ce mode permet d'ouvrir le fichier avec des réservations de modification s'il existe déjà. Donc, il ne crée pas ce qui a déjà été créé. Dans ce cas, le fichier est accessible à la fois en mode lecture et en mode écriture
rt+	Elle a les mêmes fonctions que la clé « r+ »
wt+	Elle a les mêmes fonctions que la clé « w+ »

La syntaxe de la fonction open est la suivante (voir fig. 1.1):

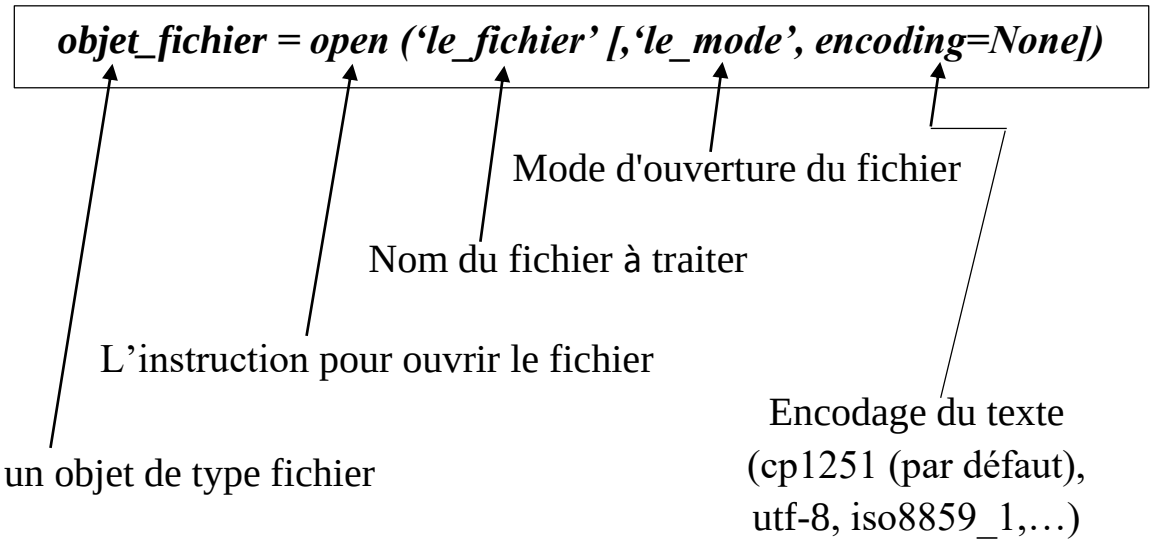


Fig. 1.1. La syntaxe de l'instruction « open »

Un exemple de l'instruction:

objet_fichier = open("le_fichier.txt", "r", encoding= "cp1251")

FR I.2. Lecture du fichier

Pour manipuler un fichier, il faut avant savoir comment le lire.

Comment lire un fichier

a) Pour lire, nous devons utiliser les instructions suivantes au choix selon le devoir à réaliser:

read() – cette instruction permet de copier intégralement tout le texte ou contenu d'un fichier dans une variable, par exemple. Cette fonction peut avoir des variables. Par exemple, si nous écrivons l'instruction de cette manière : *read(4)*, après avoir exécuté, le pointeur se retrouvera à la quatrième position (place où il y a un symbole y compris la touche « Enter » c'est-à-dire lorsqu'on va à la prochaine ligne). Donc il sera décalé de 4 caractères, l'instruction lira du début jusqu'au quatrième symbole. À chaque fois, nous pouvons avancer (augmenter de chiffre) jus-qu'atteindre la fin du fichier. Si nous voulons connaître la position actuelle du pointeur, nous pouvons utiliser la méthode *tell()*.

La syntaxe de l'instruction *read*:

objet_fichier.read()

Exercice 1.

Considérons que nous avons deux fichiers ayant presque les mêmes contenus (voir fig. 1.2 et 1.3). Dans chaque fichier, il n'y a aucune barre d'espace ou ligne vide. Après avoir lu chaque fichier avec l'instruction *read()*, déterminer quelle sera la position finale du pointeur après le dernier symbole du fichier qui est «9»?

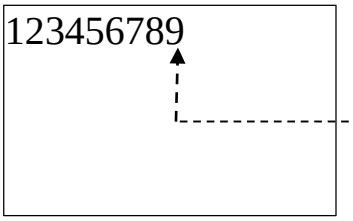


Fig. 1.2. Fichier texte avec les données étendues sur une ligne

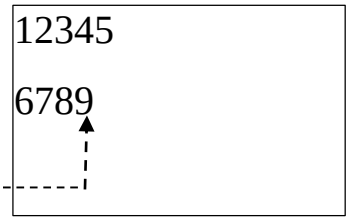


Fig. 1.3. Fichier texte avec les données étendues sur deux lignes

Réponse

A quoi va ressembler le code pour déterminer cette position finale du pointeur dans le fichier:

```
1  with open("my_file.txt", "r") as le_fichier:
2      ligne = le_fichier.read()
3      posi = le_fichier.tell()
4      print('Position № = ', posi)
```

Pour le premier fichier (voir fig. 1.1), la position sera égale à 9, et pour le deuxième fichier (voir fig. 1.2), elle sera égale à 11.

readlines() – l'instruction permet de copier l'intégralité du fichier pour la mettre dans une liste où chaque ligne sera représentée dans la dite liste comme étant un élément.

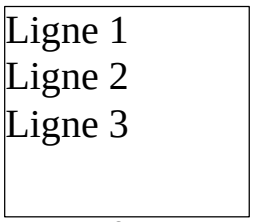
La syntaxe de l'instruction *readlines*:

```
objet_fichier.readlines(size)
```

« size » – ce paramètre est facultatif. Si le nombre d'octets renvoyés dépasse le nombre d'indices indiqué (*size*), alors aucune ligne ne sera renvoyée. Par défaut, sa valeur est égale à ***-1***. Lorsque nous ne mentionnons pas d'indice, alors toutes les lignes sont renvoyées.

Avec le code suivant, le résultat sera comme indique la figure 1.5, si et seulement si le fichier d'entrée sera semblable à celui de la figure 1.4.

```
1  with  open("my_file.txt", encoding= "utf-8") as objet_fichier
2      massive = objet_fichier.readlines()
3
```



my_file.txt

Fig. 1.4. Fichier texte avec les données étalées sur 3 lignes

['Ligne 1\n', 'Ligne 2\n', 'Ligne 3']

résultat

Fig. 1.5. Le contenu de l'identification « *massive* » dans le code

Avec ce résultat, il sera possible d'utiliser les différentes méthodes sur la liste (cf. Tom 1) pour manipuler les données du fichier entrant. Par exemple, la deuxième ligne pourra être retirée par le code: ***massive[numéro]*** *(massive[1])*.

readline() – l'instruction permet de copier progressivement si possible une seule ligne du fichier dans une chaine de caractères. Avec cette fonction, nous pouvons également spécifier le nombre d'octets de la ligne à renvoyer, en utilisant le paramètre « size ».

« size » – comme le précédant, ce paramètre est facultatif. Il détermine le nombre d'octets de la ligne à renvoyer. Par défaut, « size » a la valeur -1; Si rien n'est indiqué ou alors si nous avons la valeur -1, alors toute la ligne sera renvoyée.

La syntaxe de l'instruction *readline*:

objet_fichier.readline(*size*)

FR I.3. L'enregistrement des données dans un fichier

Comment écrire dans un fichier

write() – permet d'écrire un texte spécifié, une chaîne ou alors juste un mot dans le fichier. Cette fonction dépend du mode d'écriture choisi du tableau №1.1.

La syntaxe de l'instruction *write*:

objet_fichier.write('*txt_binaire'*)

ou

objet_fichier.write("*txt_binaire"*)

txt_binaire – un texte codé ou non codé qui sera inséré.

Pour écrire par ligne, il faudra en sauvegardant dans le ficher, introduire la clé « \n » au niveau où nous voulons commencer une nouvelle ligne.

Exercice 1.

Enregistrer cette phrase *«un texte, une chaîne ou un mot»* dans le fichier «*my_file.txt*».

```
1  with open("my_file.txt", "w", encoding="utf-8") as le_fichier:
2      a_enregistrer = "un texte, une chaîne ou un mot \n"
3      le_fichier.write(a_enregistrer)
4
5      # Une liste ne peut pas se faire enregistrer par la fonction
6      #"write", il faut utiliser une autre instruction. Le codage par
7      # defaut «cp1251» ne reconnait pas le symbole «î».
```

writelines() – cette fonction permet de prendre les éléments d'une liste, pour les enregistrer dans le fichier. Elle peut également enregistrer les données qui ne sont pas dans une liste. La syntaxe est la suivante :

objet_fichier.writelines(*une_liste*)

une_liste - Nous avons les informations (textes simples ou binaires) dans une liste qui seront insérées dans le fichier.

Exercice 2.

Nous avons une liste (*une_liste*) contenant 3 éléments : ligne №1, ligne №2 et ligne №3. Le devoir consiste à enregistrer ces données dans le fichier « *my_file.txt* ».

Réponse:

```
1  with open("my_file.txt", "w", encoding="utf-8") as le_fichier:
2      une_liste = ["ligne №1\n", "ligne №2\n", "ligne №3"]
3      le_fichier.writelines(une_liste)
4
5      # Le codage peut être retiré sans que nous assistons à une
6      # erreur. Résultat de l'exécution à la figure 1.6.
```

Résultat :

```
Ligne №1
Ligne №2
Ligne №3
```

Fig. 1.6. Contenu du fichier «my_file.txt» après exécution des codes

Nous pouvons également utiliser la fonction « print() », pas pour afficher les données sur l'écran de l'ordinateur, mais pour stocker une information quelconque dans le fichier. La syntaxe de cette commande va ressembler à celle ci:

print(*txt_binaire*, file = *objet_fichier*)

txt_binaire – un texte codé ou non codé qui sera inséré ; *objet_fichier* – nous rappelons que c'est un objet de type fichier.

Ce qui veut dire que :

objet_fichier.write("*txt_binaire*") ↔ print("*txt_binaire*", file = *objet_fichier*)

FR I.4. Les actions finales sur le fichier après son utilisation

Fermeture du fichier après utilisation

Lorsque nous ouvrons un fichier quelconque pour apporter des modifications, ou alors lorsqu'on crée un fichier pour l'utiliser, à la fin de la procédure, il est très important de le fermer avec la l'instruction « ***close()*** ». Cette fonction peut ne pas être utilisée, si nous nous adressons au fichier avec l'instruction « ***with*** ». Dans le cas où elle n'est pas utiliser, il faudra accompagner vos codes en utilisant des instructions ***Try – Except – Else – Finally***.

La syntaxe est la suivante :

objet_fichier. close()

Supprimer un fichier ou un répertoire

Cette partie ne fait pas parti des étapes à suivre pour lire ou ecrire des informations dans un fichier. Mais, étant donnée que nous connaissons déjà comment créer et écrire les informations à l'intérieur du fichier, il ne sera pas inutile de savoir également comment nous pouvons supprimer certains fichiers dans le disque. Les étapes à suivre sont les suivantes:

a) Il faut importer le module du système d'exploitation qui permettra de la réaliser :

import os

b) Il faut vérifier, si le fichier ou le répertoire existe à travers ce code:

```
if os.path.exists("my_file.txt"):
```

c) Effacer le fichier ou le répertoire s'il existe

```
os.remove("my_file.txt")
```

```
os.rmdir("MY_FOLDER")
```

Exercice avec réponses

Écrire différents codes de programme qui permettent de supprimer un fichier ou un répertoire dans un disque quelconque.

Réponse 1 – Supprimer un fichier sans les instructions Try, ...

```
1  import os
2
3  if os.path.exists("my_file.txt"):
4      os.remove("my_file.txt")
5      print("Le fichier a été effacé")
6  else:
7      print("Le fichier n'existe pas, donc ineffaçable.")
8
```

Réponse 2 – Supprimer un fichier avec les instructions Try, ...

```
1  import os
2
3  try:
4      os.remove("my_file.txt")
5      print("Le fichier a été effacé")
6  except FileNotFoundError:
7      print("Le fichier n'existe pas, donc ineffaçable")
8
```

Réponse 3 – Supprimer un répertoire sans les instructions Try, ...

```
1  import os
2
3  if os.path.exists("my_file.txt"):
4      os.remove("my_file.txt")
5      print("Le répertoire a été effacé")
6  else:
7      print("Le répertoire n'existe pas, donc ineffaçable")
8
```

Réponse 4 – Supprimer un répertoire avec les instructions Try, ...

```
1  import os
2
3  try:
4      os.rmdir("my_file.txt")
5      print("Le répertoire a été effacé")
6  except FileNotFoundError:
7      print("Le répertoire n'existe pas, donc ineffaçable")
8
```

Exercice 2 avec réponses

Utiliser toutes informations reçues (comment lire, écrire et effacer une information dans un fichier) pour lire (ligne par ligne) de différentes manières les données d'un fichier texte quelconque:

a) Utiliser la fonction readline() pour lire un fichier texte ligne par ligne;

b) Utiliser la fonction readlines() pour lire les chaînes et les transformer en liste ;

c) Lecture du fichier ligne par ligne à partir de l'objet File sans les fonctions readline() et readlines() ;

d) Lecture du fichier ligne par ligne à partir de l'objet File avec la fonction « readline() » et l'instruction « while » sans condition « if » ;

e) Lecture du fichier ligne par ligne à partir de l'objet File en utilisant la fonction « readline() » et avec condition « if ».

Réponse 1. Utilisation de la fonction readline() pour lire un fichier texte ligne par ligne.

```
1  le_fichier = open("c:\\work\\my_file.txt", "r")
2  k=0
3  while True:
4      ligne = le_fichier.readline()
5      if not ligne:
6          break
7      k+=1
8      print('Phrase №{0}'.format(k),ligne.strip())
9
10 le_fichier.close()
```

Commentaire des codes précédents

#1 Nous obtenons l'objet qui est le fichier en mode lecture.
#2 Initialisation du compteur.
#3 Début de la boucle-s'exécute jusqu'à ce qu'il ne trouve pas une interruption dans la boucle.
#4 Lire la chaîne et écrire le contenu dans la variable qui a pour nom «ligne».
#6 Nous interrompons la boucle si la chaîne est vide(#5).
#7 Incrémentons le compteur d'un pas, celui ci sera le numéro de la prochaine ligne de sortie.
#8 Nous sortons le numéro de la ligne, accompagné du mot clé «Phrase» et la chaîne qui le correspond.
#10 Fermeture du fichier.

Réponse 2. Utiliser la fonction readlines() pour lire les chaînes et les transformer en liste

```
1  le_fichier = open("c:\\work\\my_file.txt", "r")
2  lines = le_fichier.readlines()
3  print('Texte lu dans une liste: ',lines)
4  k = 1
5  for ligne in lines:
6      print('Phrase №{0}'.format(k),ligne.strip())
7      k+=1
8
9  le_fichier.close()
```

Cette partie peut ne pas être.

Commentaire des codes précédents

#1 Nous obtenons l'objet qui est le fichier en mode lecture.
#2 Nous lisons toutes les lignes avec la fonction readlines().
#3 Faire sortir le texte qui a été lu, il se trouvera dans une liste.
#4 Initialisation du compteur.
#5 Nous ouvrons la boucle et effectuons un processus d'itération sur chaque ligne.
#6 Nous sortons le numéro de la ligne, accompagné du mot clé «Phrase» et la chaîne qui le correspond.
#7 Incrémentons le compteur d'un pas, celui ci sera le numéro de la prochaine ligne de sortie.

#9 Fermeture du fichier.

La partie hachurée peut ne pas figurer dans les codes , dans la mesure où la ligne numéro 2 lit les informations du fichier pour la mettre dans une liste. De la liste, nous pouvons faire d'elles ce que nous voulons.

Réponse 3. Lecture du fichier ligne par ligne à partir de l'objet File sans les fonctions readline() et readlines().

```
1  k=1
2  with open("c:\\work\\my_file.txt", "r") as le_fichier:
3      for ligne in le_fichier:
4          print('Phrase №{0}'.format(k),ligne.strip())
5          k+=1
6
7  le_fichier.close()
```

Commentaire des codes précédents
#1 Initialisation du compteur. *#2 Nous obtenons l'objet qui est le fichier en mode lecture (Les données du fichiers s'enregistrent dans "le_fichier". Elles sont positionnées en liste comme une massive de données).* *#3 Nous ouvrons la boucle et effectuons un processus d'itération sur chaque ligne.* *#4 Nous sortons le numéro de la ligne, accompagné du mot clé «Phrase» et la chaîne qui le correspond.* *#5 Incrémentons le compteur d'un pas, celui ci sera le numéro de la prochaine ligne de sortie.* *#7 Fermeture du fichier – elle n'est pas obligatoire à cause de la présence de l'instruction « with ».*

Réponse 4. Lecture du fichier ligne par ligne à partir de l'objet File avec la fonction « readline() » et l'instruction « while » sans condition « if ».

```
1  k=0
2  with open("c:\\work\\my_file.txt", "r") as le_fichier:
3      ligne = le_fichier.readline()
4      while ligne:
5          k+=1
6          print('Строка №{0}'.format(k), ligne.strip())
7          # ou
8          print('Строка №{0}'.format(k), ligne.strip(), end="  ")
9          ligne = le_fichier.readline()
10 le_fichier.close()
```

Commentaire des codes précédents
#1 Initialisation du compteur. *#2 Nous obtenons l'objet qui est le fichier en mode lecture (Les données du fichiers s'enregistrent dans "le_fichier". Elles sont positionnées en liste comme une massive de données).* *#3 Lire la chaîne et écrire le contenu dans la variable qui a pour nom «ligne».* *#4 Début de la boucle – s'exécute chaque fois qu'elle trouvera une nouvelle ligne dans le fichier.* *#5 Incrémentons le compteur de 1 pas, celui ci sera le numéro de la prochaine ligne de sortie.* *#6 Nous sortons le numéro de la ligne, accompagné du mot clé «Phrase» et la chaîne qui le correspond.* *#8 Nous sortons toutes les lignes avec numéros sur une seule ligne en ajoutant la commande « end=" " » à la fin.* *#9 Lire la chaîne et écrire le contenu dans la variable qui a pour nom «ligne».* *#10 Fermeture du fichier – elle n'est pas obligatoire à cause de l'instruction with*

Réponse 5. Lecture du fichier ligne par ligne à partir de l'objet File en utilisant la fonction « readline() » et avec condition « if »

```
1  k=0
2  with open("c:\\work\\my_file.txt", "r") as le_fichier:
3      while True:
4          ligne = le_fichier.readline()
5          if not ligne:
6              break
7          k+=1
8          print('Phrase №{0}'.format(k),ligne.strip(), end="  ")
9          #print('Phrase №{0}'.format(k),ligne.strip())
10 le_fichier.close()
```

Commentaire des codes précédents
#1 Initialisation du compteur. *#2 Nous obtenons l'objet qui est le fichier en mode lecture (Les données du fichiers s'enregistrent dans "le_fichier". Elles sont positionnées en liste comme une massive de données.)* *#3 Début de la boucle – s'exécute jusqu'à ce qu'il ne trouve pas une interruption dans la boucle.* *#4 Lire une chaîne et écrire le contenu dans la variable qui a pour nom « ligne ».* *#5 Vérifie s'il n'y a plus de ligne à lire.* *#6 Dans le cas où la chaîne est vide, nous interrompons la boucle.* *#7 Incrémentons le compteur de 1 pas, le résultat sera le numéro de la prochaine ligne de sortie.* *#8 Nous sortons chaque fois la ligne obtenue avec un label et numéro en ajoutant la commande « end=" " » à la fin.* *#9 Même chose que le précédent, à l'exception de l'absence de la commande « end=" " »* *#10 Fermeture du fichier – elle n'est pas obligatoire à cause de la présence de l'instruction « with ».*

En conclusion, nous pouvons vous conseiller d'utiliser de préférence l'instruction « with » lorsque vous traitez des fichiers, ce qui permettra de garantir leur fermeture si une erreur apparaîssait. Cette commande collabore bien avec des exceptions qui peuvent intervenir. Nous pouvons utiliser « with », comme nous pouvons utiliser le bloc *try…finally* pour combattre les erreurs, mais la présence de « with » sera beaucoup plus effective dans la mise en application du programme.

Étant donnée que nous connaissons déjà comment lire et écrire certaines informations dans un fichier, pour la suite, il est nécessaire maintenant d'apprendre à analyser et faire des modifications dans des textes existants.

FR I.5. Questions de compréhension du cours

1) Quel est l'avantage et la difficulté de l'utilisation des fichiers codés en forme "binaire" par rapport aux fichiers textes non codés?

2) Quel mode de rédaction assisterons nous dans le fichier « my_file2.txt » après l'exécution de la commande suivante :

with open("my_file.txt", encoding="koi8_r") as le_fichier,

et après avoir écrit deux chiffres par la fonction write() ?

3) Pourquoi devons nous fermer un fichier après l'avoir ouvert pour diverses manipulations ?

4) Quel sera le type de codage du texte après l'exécution de cette commande:

with open("my_file1.txt", "w+") as objet_fichier,

ou alors assisterons nous à une erreur, après avoir lu une ligne dans le fichier «my_file1.txt» avec la fonction « readline() » ?

5) Quelle est la différence entre les fonctions suivantes : read(), readline(), readlines() ?

6) Quelle est la différence entre les fonctions suivantes : write(), writeline() ?

7) Pouvons nous se passer des fonctions readline() et readlines() pour lire un fichier texte quelconque ?

8) Quelle est l'utilité de l'instruction « with » dans la manipulation des fichiers ?

Suite à la page 119


RU Глава II. ВЗАИМОДЕЙСТВИЕ С PDF ФАЙЛАМИ В PYTHON

RU II.1. Библиотека PDFplumber – чтение и/или запись в PDF-файл

PDFplumber - это инструмент, который мы можем использовать в Python для чтения и извлечения текста или другой информации из документа PDF. Модуль **PDFplumber** обладает уникальной особенностью извлечения информации из документа PDF. Эта визуальная особенность ориентирована на форматирование документа PDF. Внутри модуля находятся следующие пакеты: *pycryptodome* (низкоуровневый криптографический пакет), *Wand* (библиотека графической обработки), *pdfminer.six* (инструмент для извлечения информации из файла PDF), *PDFplumber* (библиотека позволяет обрабатывать информацию в файле в формате PDF).

а) последовательность действий по использованию модуля ***PDFplumber*** для извлечения информации (например, текста)

Этап №1 – Установка библиотеки, если она еще не была выполнена или если ее нет в хранилище модулей редактора Python.

Способ 1: pip install PDFplumber

Он будет работать во встроенной среде IDLE

Способ 2. Если вы используете **Anaconda,** то вы должны действовать в соответствии с инструкциями, если после этого модуль сможет работать от редактора *Anaconda* «***Spyder***»

Шаг 1 – запуск Anaconda Navigator

Шаг 2 – запуск ***Prompt CMD.exe*** или ***PowerShell***

Шаг 3 – Напишите команду «*pip install pdfplumber*» в терминале командной строки и выполните команду. sur le terminale d'exécution et lancer la commande.

Этап №2 – необходимо импортировать модуль PDFplumber

```
import  PDFplumber
```

б) Пример кода в Python для извлечения текста из файла PDF

```
1  import PDFplumber
2
3  with PDFplumber.open("document_path.PDF") as temp:
4      first_page = temp.pages[0]
5      print(first_page.extract_text())
```

Чтобы загрузить PDF-файл, защищенный паролем, необходимо указать в аргумент «password» секретное слово. Например, у нас будет следующая команда

```
pdfplumber.open("nom_fichier.pdf", password = "slovo_secret")
```

RU II.2. Библиотека PDFminer.six – чтение и / или запись в PDF-файл

PDFminer.six [2] также является инструментом, который может быть использован в Python для чтения и извлечения текста или другой информации (получение определенных данных шрифта, получение цвета текста, извлечение изображений и многое другое) из документа PDF. Давайте попробуем рассмотреть различные возможности этого модуля (как его установить и использовать) в нескольких пунктах.

а) последовательность действий по использованию модуля ***PDFminer.six*** для извлечения информации (например, текста).

Процедура для просмотра: установите, так называемую библиотеку, если её нет в хранилище модулей редактора Python.

```
Метод 1: pip install  PDFminer.six        (в  windows)
```

Можно проверить, установлен ли пакет ***PDFminer.six*** в интерпретаторе языка Python. Мы можем попытаться прочитать либо версию пакета по атрибуту __version__, либо его имя по атрибуту __name__, либо его имя основного файла, включая путь, по которому он был установлен атрибутом __file__. Например:

```
import pdfminer
print('Имя файла: ',pdfminer.__file__)
```

Если у вас нет ошибок, то в этом случае можно использовать pdfminer.six в качестве пакета языка Python. Только вам нужно будет знать, что пакет **PDFminer.six** устанавливается вместе с дополнительными инструментами, расширяющие функционал библиотеки. Чтобы проверить правильность установки этих инструментов, необходимо выполнить следующую команду в терминале:

Файл находится в каталоге, в котором установлен интерпретатор языка Python.

```
pdf2txt.py --version
```

Таким образом, результат должен выглядеть следующим образом:

```
pdfminer.six 20200517
```

Установленная версия

Как мы уже говорили чуть выше, в модуле **pdfminer.six** есть инструменты, которые можно использовать из командной строки. С помощью этих инструментов, некоторые пользователи могут извлекать текст из файла *PDF*. Давайте изучим некоторые из этих инструментов:

Инструмент **pdf2txt** – это программное обеспечение, которое используется в командной строке и может извлекать информацию в виде простого текста из файла PDF и отображать ее на экране (или в других файлах), где она была запущена. Файл содержит коды, который можно открыть и изучить для получения дополнительных знания в области программирования.

Как его запустить или использовать?

а) синтаксис команды выглядит следующим образом:

python Scripts/pdf2txt.py [-h] [--version] [--debug] [--disable-caching] [--page-numbers PAGE_NUMBERS [PAGE_NUMBERS ...]] [--pagenos PAGENOS] [--maxpages MAXPAGES] [--password PASSWORD] [--rotation ROTATION] [--no-laparams] [--detect-vertical] [--char-margin CHAR_MARGIN] [--word-margin WORD_MARGIN] [--line-margin LINE_MARGIN] [--boxes-flow BOXES_FLOW] [--all-texts] [--outfile OUTFILE] [--output_type OUTPUT_TYPE] [--codec CODEC] [--output-dir OUTPUT_DIR] [--layoutmode LAYOUTMODE] [--scale SCALE] [--strip-control] files [files ...]

Объяснение позиционных параметров (см. табл. 2.1)

Ключи или параметры – это данные, которые необходимо ввести для управления поведением библиотеки.

Таблица 2.1 – Объяснение параметров инструмента *pdf2txt*

№	Аргументы команды **pdf2txt**	Объяснение, роль
0	[ ]	То, что находится внутри скобок, может использоваться или может не использоваться при выполнении команды
1	-h или --help	Получить справку, информацию о синтаксическом описании команды на английском языке
2	-v или --version	Вывод версии библиотеки
3	-d или --debug	Позволяет использовать вести журнала отладки (по умолчанию: False = нет)
4	-C или --disable-caching	Если у нас есть кэширование или ресурсы, такие как шрифты, то их необходимо отключить. По умолчанию у нас есть значение “False”
Аргументы, используемые при анализе файла в формате PDF		
1	--page-numbers	Этот ключ полезен, когда у нас есть выбранный список страниц для анализа. Здесь мы можем указать только те страницы, которые нужно проанализировать. Эти номера страниц будут разделены пробелами
2	-p PAGENOS или --pagenos PAGENOS	Список номеров страниц, разделенных запятыми для анализа
3	-m MAXPAGES или --maxpages MAXPAGES	Максимальное количество страниц для сканирования

4	-P или --password PASSWORD	Пароль, используемый для декодирования файла в формате PDF
4	-R ROTATION или --rotation ROTATION,	Угол поворота в градусах страницы файла в формате PDF
Параметры, используемые при анализе макета страницы		
1	-n или --no-laparams	Используется, если необходимо игнорировать параметры анализа компоновки. По умолчанию оно равно “False”
2	-V или --detect-vertical	Используется, если при анализе макета необходимо учитывать вертикальный текст.
3	-M CHAR_MARGIN или --char-margin CHAR_MARGIN	Если два символа находятся ближе, чем поле, они считаются частью одной строки. Поле задается относительно ширины символа. По умолчанию она равна 2.0
4	-W WORD_MARGIN или --word-margin WORD_MARGIN	Если два символа в одной строке находятся дальше, чем поле, они считаются двумя отдельными словами, и для удобства чтения будет добавлено промежуточное пространство. Отметим, что поле задается относительно ширины символа. По умолчанию она равна 0.1
5	-L LINE_MARGIN или --line-margin LINE_MARGIN	Если две строки расположены близко друг к другу, они считаются частью одного абзаца. Отметим, что поле задается относительно высоты строки. По умолчанию она равна 0,5
6	-F BOXES_FLOW или	Указывает приоритет горизонтального и вертикального

	--boxes-flow BOXES_FLOW	положения текста при определении порядка строк. Значение должно быть в диапазоне от -1.0 (учитывается только горизонтальное положение) до +1.0 (учитывается только вертикальное положение). По умолчанию она равна 0.5
6	-F BOXES_FLOW или --boxes-flow BOXES_FLOW	
7	-A или --all-texts	Если анализ макета файла должен выполняться по тексту в виде рисунков
Используется при создании вывода информации		
1	-o или --outfile	Позволяет указать путь к файлу, где будет находиться информация, полученная из файла pdf.
	python pdf2txt.py ***--outfile*** *c:\work\res.txt c:\work\file_to_read.pdf* *другой пример команды кода:* *python pdf2txt.py* ***-o*** *c:\work\res.txt c:\work\file_to_read.pdf*	
2	-t или --output_type	Позволяет определить тип вывода, который будет генерироваться: “текст" (“text”), “html”, “xml”, “tag” (тег)
	python pdf2txt.py -o c:\work\res.html ***-t*** *html c:\work\file_to_read.pdf*	
3	-c или --codec	Здесь мы должны указать, какую текстовую кодировку будем использовать в выходном файле
4	-O или --output-dir	Мы используем эту команду, чтобы указать имя выходного каталога, в которым будут сохранены изображения, извлеченные из файла pdf. Если команда не указана с

		именем каталога, то изображения не будут извлечены.
	python pdf2txt.py -o c:\work\res.html -t html ***-O*** *c:\work\image_folder c:\work\file_with_image.pdf* *другой пример команды кода:* *python pdf2txt.py -o c:\work\res.txt* ***-O*** *c:\work\image_folder c:\work\file_with_image.pdf*	
5	-Y или --layoutmode	Определяет тип макета страницы, который будет использоваться при создании файла в формате html {normal, exact, loose}. Если это “normal”, то каждая строка размещается отдельно в html. Если это “exact”, то каждый символ позиционируется отдельно в html. Если это “ loose”, то получаем тот же результат, что и “normal”, с разницей в дополнительной новой строке после каждой строки текста. Этот ключ используется только тогда, когда тип «*output_type*» является html.
6	-s или --scale	Параметр изменения масштабирования, используемый при создании HTML-файла. Подобно предыдущему, этот ключ используется только тогда, когда тип *«output_type»* является html.
7	-S или --strip-control	Это позволяет удалить оператор управления из текста. Этот ключ используется только тогда, когда тип «output_type» является xml.

Файл для извлечения (см. рисунок 2.1)		
1	files [files ...]	В команде мы должны, как минимум и в обязательном порядке, ввести путь к файлу PDF. Можно указать один или несколько файлов в формате PDF.

Строка 1

Строка 2

Изображение 1

Строка 3

Изображение 2

Строка 4

Рис. 2.1. пример файла («*file_to_read.pdf*») с содержимым в формате PDF

б) Запустить терминал (см. рис. 2.2)

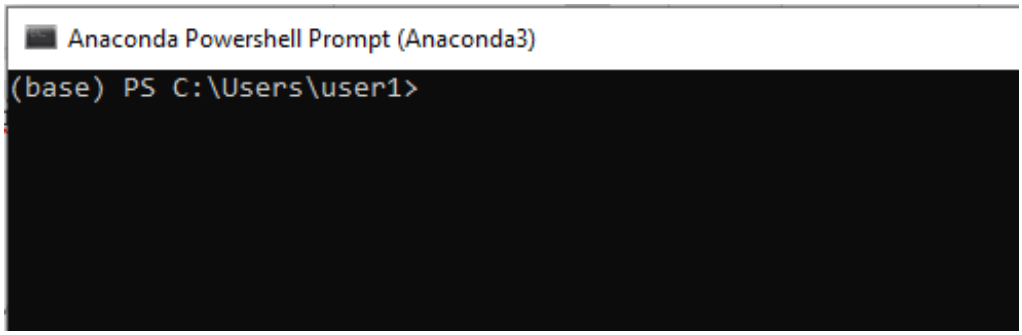


Рис. 2.2. Один из терминалов операционной системы Windows

в) напишите следующих команд в командной строке:

```
python  pdf2txt.py  file_to_read.pdf
```

После выполнения этой команды мы увидим, как на экране отобразится текст, находящийся в файле *file_to_read.pdf*.

Для правильной работы файл «*file_to_read.pdf*» должен находиться в каталоге «user1» для этого конкретного примера, иначе потребуется указать полный путь к файлу «*file_to_read.pdf*», если его нет в этом каталоге. В том же случае, если операционная система не сможет автоматически найти файл «***pdf2txt.py***», то необходимо указать команде для выполнения полный путь к рассматриваемому файлу.

python pdf2txt.py ***--outfile*** *c:\work\res.txt c:\work\file_to_read.pdf*

другой способ, который можем использовать:

python pdf2txt.py ***-o*** *c:\work\res.txt c:\work\file_to_read.pdf*

После выполнения команды будет создан новый файл «*res.txt*», если его нет; и внутри файла будет сохранена информация в виде текста (см. рис. 2.3).

Строка 1
Строка 2
Строка 3
Строка 4

Рис. 2.3. Новый файл «res.txt» без изображений в формате TXT

python pdf2txt.py -o c:\work\res.html ***-t*** *html c:\work\file_to_read.pdf*

После выполнения команды будет создан новый файл « *res.html* » если его нет; и внутри файла будет храниться информация, идентичная приведенной на рисунке 2.3, но в формате HTML.

*python pdf2txt.py -o c:\work\res.html -t html –**O** c:\work\image_folder c:\work\file_to_read.pdf*

другой способ, который можем использовать:

*python pdf2txt.py -o c:\work\res.txt -**O** c:\work\image_folder c:\work\file_to_read.pdf*

После выполнения команды будет создан новый файл « *res.html* », если его нет; и внутри файла будет храниться информация с таким же содержимым, как показано на рисунке 2.3, но в формате HTML. Кроме того, будет создан каталог под названием «*image_folder*», где будут находиться все изображения из входного файла «*file_to_read.pdf*».

Примечание. Если операционная система не может автоматически найти файл «pdf2txt***.py***», тогда необходимо указать в выполняемой команде полный путь к файлу.

Инструмент **dumppdf.py** – это программное обеспечение, которое используется в командной строке и может извлекать внутреннюю структуру файла PDF и отображать его на экране, где он был запущен. Этот инструмент предназначен и полезен также для всех, кто обрабатывает файлы в формате PDF.

Как его запустить или использовать?

а) ознакомиться с синтаксисом команды, который выглядит следующим образом:

dumppdf.py *[-h] [--version] [--debug] [--extract-toc | --extract-embedded EXTRACT_EMBEDDED] [--page-numbers PAGE_NUMBERS [PAGE_NUMBERS ...]] [--pagenos PAGENOS] [--objects OBJECTS] [--all] [--password PASSWORD] [--outfile OUTFILE] [--raw-stream | --binary-stream | --text-stream] files [files*

Описание параметров (см. табл. 2.2)

Ключи или параметры - это данные, которые необходимо ввести для управления поведением библиотеки.

Таблица 2.2 – Объяснение параметров инструмента *dumppdf*

№	Аргументы команды **dumppdf**	Объяснение или роль
0	[ ]	То, что находится внутри скобок, может использоваться или может не использоваться при выполнении команды
Необязательные ключи или аргументы		
1	-h или --help	Позволяет получить справку, т. е. информацию о синтаксическом описании команды
2	-v или --version	Позволяет определить версию программного обеспечения
3	-d или --debug	Используется для уровня ведения журнала отладки
4	-T или --extract-toc	Он позволяет извлечь структуру из контура
5	-E EXTRACT_EMBEDDED или --extract-embedded EXTRACT_EMBEDDED	Позволяет извлекать встроенные файлы

Ключи или аргументы, используемые при анализе файла в формате PDF		
1	--page-numbers PAGE_NUMBERS [PAGE_NUMBERS ...]	Этот ключ полезен, когда у есть выбранный список страниц для анализа. Здесь мы можем указать только те страницы, которые нужно проанализировать. Эти номера страниц будут разделены пробелами
2	--pagenos PAGENOS, -p PAGENOS	Позволяет выбрать страницу для сканирования. Использует номера страниц для более четкого ввода аргументов. Вы можете использовать список номеров страниц, который будет разделен запятыми
	*python dumppdf.py **--pagenos** 2 -o c:\work\res.txt c:\work\file_to_read.pdf*	
3	-i OBJECTS или --objects OBJECTS	Используется для извлечения информации об определенных объектах. Если требуется проанализировать или извлечь несколько объектов, то номера объектов будут разделены запятыми
	*python dumppdf.py **--objects** 1 -o c:\work\res.txt c:\work\file_to_read.pdf* другая версия, которую можем использовать: *python dumppdf.py **-i** 1 -o c:\work\res.txt c:\work\file_to_read.pdf*	
4	-a или --all	Позволяет получить всю информацию обо всех объектах в файле в формате PDF. Принимаем к извлечению структуру всего файла

	python dumppdf.py -o c:\work\res.txt ***--all*** *c:\work\file_to_read.pdf* другая версия, которую можем использовать: *python dumppdf.py -o c:\work\res.txt* ***-a*** *c:\work\file_to_read.pdf*	
5	-P PASSWORD или --password PASSWORD,	Пароль, используемый для расшифровки PDF-файла, если он существует
Используется при создании вывода информации		
1	-o или --outfile	Позволяет указать путь к файлу, в котором будет находиться информация, полученная из файла PDF
2	-r или --raw-stream	Позволяет записывать объекты потока без кодирования
3	-b или --binary-stream	Позволяет записывать объекты потока с двоичным кодированием
4	-t или --text-stream	Позволяет записывать объекты потока в виде обычного текста
Файл для извлечения (см. рисунок 2.1)		
1	files [files ...]	В команде должны как минимум и в обязательном порядке ввести путь к файлу PDF. Имеются один или несколько файлов в формате PDF.

б) запустить терминал (см. предыдущий рисунок 2.2)

в) написать одну из команд, рассмотренных ранее, в командной строке:

python dumppdf.py -o c:\work\res.txt ***--all*** *c:\work\file_to_read.pdf* autre version que nous pouvons utiliser: *python dumppdf.py -o c:\work\res.txt* ***-a*** *c:\work\file_to_read.pdf*

После выполнения команды, будет создан новый файл « *res.txt* », если его нет; и в файл будем сохранена всё внутренняя структура файла PDF « *file_to_read.pdf* » (см. рис. 2.3)

Если уберем команду « *-o c:\work\res.txt* ***–all***» или так « *-o c:\work\res.txt* ***-a*** », тогда на экране отобразится результат выполнения команды.

```
<trailer>
<dict size="6">
<key>Size</key>
<value><number>194</number></value>
<key>Root</key>
<value><ref id="1" /></value>
<key>Info</key>
<value><ref id="61" /></value>
<key>ID</key>
<value><list size="2">
<string size="16">&#140;&#186;&#190;&#250; ...</string>
<string size="16">&#140;&#186;&#190;&#250; ... </string>
</list></value>
<key>Prev</key>
<value><number>829799</number></value>
<key>XRefStm</key>
<value><number>829112</number></value>
</dict>
</trailer>

<trailer>
<dict size="8">
<key>Type</key>
...
```

Рис. 2.3. Часть результата выполнения предыдущей команды – структура файла PDF

*python dumppdf.py **--pagenos** 2 -o c:\work\res.txt c:\work\file_to_read.pdf*

После выполнения этой команды, мы увидим более детальный анализ аргументов на странице 2 внутренней структуры файла PDF «*file_to_read.pdf*» (см. рис. 2.3); результат будет сохранен в файле «*res.txt*».

После выполнения кода мы получим в файле «*res.txt*» информацию о первом объекте из файла PDF «*file_to_read.pdf*».

*python dumppdf.py **--objects** 1 -o c:\work\res.txt c:\work\file_to_read.pdf*

Примечание. Если операционная система не сможет автоматически найти файл «***dumppdf.py***», тогда необходимо указать в выполняемой команде полный путь к рассматриваемому файлу (см. рис. 2.4 и 2.5). Эти файлы в операционной системе Windows обычно находятся в каталоге «Script», где был установлен интерпретатор языка Python.

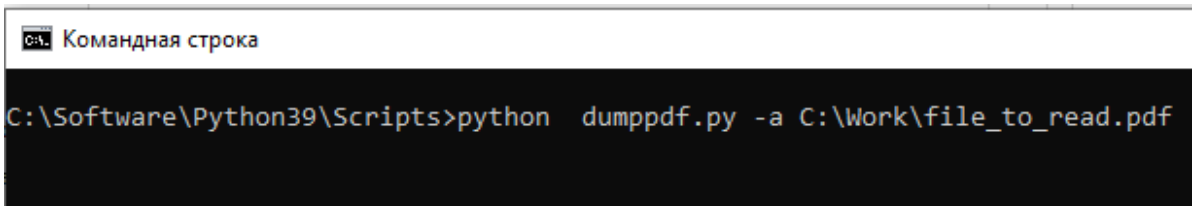


Рис. 2.4. Первый вариант команды, выполняемой в Windows

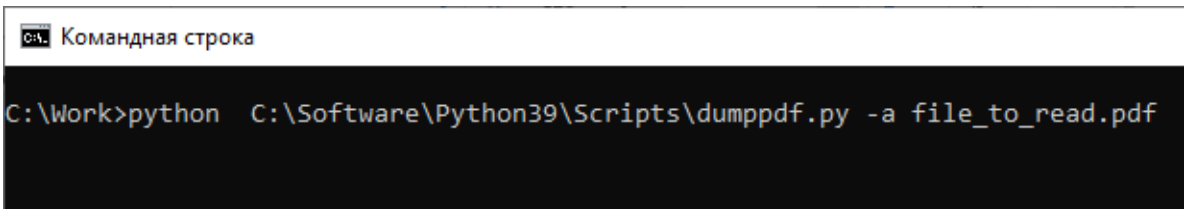


Рис. 2.5. Второй вариант команды, выполняемой в Windows

Результат выполнения предыдущей команды приведен на рисунке 2.6.

Выбрать Командная строка

```
C:\Software\Python39\Scripts>python  dumppdf.py -a C:\Work\file_to_read.pdf
<pdf><object id="1">
<dict size="5">
<key>Type</key>
<value><literal>Catalog</literal></value>
<key>Pages</key>
<value><ref id="2" /></value>
<key>Lang</key>
<value><string size="5">ru-RU</string></value>
<key>StructTreeRoot</key>
<value><ref id="8" /></value>
<key>MarkInfo</key>
<value><dict size="1">
<key>Marked</key>
<value><number>True</number></value>
</dict></value>
</dict>
</object>
...
<trailer>
<dict size="4">
<key>Size</key>
<value><number>21</number></value>
<key>Root</key>
<value><ref id="1" /></value>
<key>Info</key>
<value><ref id="7" /></value>
<key>ID</key>
<value><list size="2">
<string size="16">&#207;&#219;a0W&#147;@C&#134;&#143;&#129;&#248;&#139;&#130;&#
<string size="16">&#207;&#219;a0W&#147;@C&#134;&#143;&#129;&#248;&#139;&#130;&#
</list></value>
</dict>
</trailer>

</pdf>
```

Рис. 2.6. Часть результата выполнения предыдущей команды

Продолжение на странице 137

EN Chapter II. INTERACTION WITH PDF FILES IN PYTHON

EN II.1. PDFplumber Library – reading and/or writing to a PDF file

PDFplumber is a tool that we can use in Python to read and extract text or other information from a PDF document. The **PDFplumber** module has a unique feature of extracting information from a PDF document. This visual feature is focused on formatting a PDF document. The following packages are located inside the module: *pycryptodome* (a low-level cryptographic package), *Wand* (a graphics processing library), *pdfminer.six* (a tool for extracting information from a PDF file), *PDFplumber* (a library that allows you to process information in a PDF file).

a) the sequence of actions for using the ***PDFplumber*** module to extract information (for example, text)

Step №1. Installing the library if it has not been done yet or if it is not in the Python Editor module repository.

Method 1: pip install PDFplumber

It will work in the built-in IDLE environment.

Method 2. If you are using **Anaconda**, then you must act according to the instructions, if after that the module will be able to work from the Anaconda «***Spyder***» editor.

Step 1 – Launch Anaconda Navigator

Step 2 – Launch ***Prompt CMD.exe*** or ***PowerShell***

Step 3 – Write the command «*pip install pdfplumber*» in the command line terminal and run the command. sur le terminale d'exécution et lancer la commande.

Step №2. You need to import the PDF plumber module.

```
import  PDFplumber
```

b) Sample Python code for extracting text from a PDF file.

```
1  import PDFplumber
2
3  with PDFplumber.open("document_path.PDF") as temp:
4      first_page = temp.pages[0]
5      print(first_page.extract_text())
```

To download a password-protected PDF file, you must specify a secret word in the «password» argument. For example, we will have the following command.

```
pdfplumber.open("name_file.pdf", password = "slovo_secret")
```

EN II.2. PDFMiner.six library – reading and/or writing to a PDF file

PDFMiner.six [2] is also a tool that can be used in Python to read and extract text or other information (getting certain font data, getting text color, extracting images, and more) from a PDF document. Let's try to consider the various features of this module (how to install and use it) in several points.

a) a sequence of actions for using the ***PDFMiner.six*** module to extract information (for example, text).

Procedure for viewing: Install the so-called library if it is not in the Python editor module repository.

```
Method 1: pip install  PDFminer.six          (in  windows)
```

You can check whether the ***PDFMiner.six*** package is installed in the Python interpreter. We can try to read either the package version by the __version__ attribute, or its name by the __name__ attribute, or its main file name, including the path where it was installed by the __file__ attribute. For example:

```
import pdfminer
print('The file is:',pdfminer.__file__)
```

If you have no errors, then in this case you can use *pdfminer.six* as a Python package. Only you will need to know that the **PDFMiner.six** package is installed along with additional tools that extend the functionality of the library. To check the correct installation of these tools, you need to run the following command in the terminal:

The file is located in the directory where the Python interpreter is installed

```
pdf2txt.py --version
```

So the result should look like this:

```
pdfminer.six 20200517
```

Installed version

As we said just above, the **pdfminer.six** module has tools that can be used from the command line. With these tools, some users can extract text from a PDF file. Let's explore some of these tools:

The **pdf2txt** tool is a software that is used on the command line and can extract information in plain text from a PDF file and display it on the screen (or in other files) where it was launched. The file contains codes that can be opened and studied to gain additional knowledge in the field of programming.

How to run it or use it ?

a) the syntax of the command is as follows:

python Scripts/pdf2txt.py [-h] [--version] [--debug] [--disable-caching] [--page-numbers PAGE_NUMBERS [PAGE_NUMBERS ...]] [--pagenos PAGENOS] [--maxpages MAXPAGES] [--password PASSWORD] [--rotation ROTATION] [--no-laparams] [--detect-vertical] [--char-margin CHAR_MARGIN] [--word-margin WORD_MARGIN] [--line-margin LINE_MARGIN] [--boxes-flow BOXES_FLOW] [--all-texts] [--outfile OUTFILE] [--output_type OUTPUT_TYPE] [--codec CODEC] [--output-dir OUTPUT_DIR] [--layoutmode LAYOUTMODE] [--scale SCALE] [--strip-control] files [files ...]

Explanation of positional parameters (see Table 2.1)

Keys or parameters are data that must be entered to control the behavior of the library.

Table 2.1 – Explanation of the parameters of the *pdf2txt* tool

№	Arguments of the **pdf2txt** command	Explanation
0	[ ]	What is inside the brackets may or may not be used when executing the command.
1	-h or --help	Get help, information about the syntactic description of the command in English
2	-v or --version	Output of the library version

3	-d or --debug	Allows you to use debug logging (default: False = none)
4	-C or --disable-caching	If we have caching or resources such as fonts, then they need to be disabled. By default, we have the value “False”
Arguments used when analyzing a PDF file		
1	--page-numbers	This key is useful when we have a selected list of pages to analyze. Here we can specify only those pages that need to be analyzed. These page numbers will be separated by spaces
2	-p PAGENOS or --pagenos PAGENOS	A comma-separated list of page numbers for analysis
3	-m MAXPAGES or --maxpages MAXPAGES	The maximum number of pages to scan
4	-P or --password PASSWORD	The password used to decode the PDF file
4	-R ROTATION or --rotation ROTATION,	The rotation angle in degrees of the PDF file page
Parameters used when analyzing the page layout		
1	-n or --no-laparams	Used if you need to ignore the layout analysis parameters. By default, it is “False”
2	-V or --detect-vertical	Используется, если при анализе макета необходимо учитывать вертикальный текст.
3	-M CHAR_MARGIN or	If two characters are closer than the field, they are considered part of the

	--char-margin CHAR_MARGIN	same string. The field is set relative to the width of the character. By default, it is 2.0
4	-W WORD_MARGIN or --word-margin WORD_MARGIN	If two characters in the same line are further away than the field, they are considered two separate words, and an intermediate space will be added for readability. Note that the field is set relative to the width of the character. By default it is 0.1
5	-L LINE_MARGIN or --line-margin LINE_MARGIN	If two lines are located close to each other, they are considered part of the same paragraph. Note that the field is set relative to the line height. By default, it is 0.5
6	-F BOXES_FLOW or --boxes-flow BOXES_FLOW	Specifies the priority of the horizontal and vertical position of the text when determining the order of lines. The value should be in the range from -1.0 (only the horizontal position is taken into account) to +1.0 (only the vertical position is taken into account). By default, it is 0.5
6	-F BOXES_FLOW or --boxes-flow BOXES_FLOW	
7	-A or --all-texts	If the analysis of the file layout should be performed on the text in the form of drawings

Used when creating information output		
1	-o or --outfile	Allows you to specify the path to the file where the information obtained from the *pdf* file will be located
	python pdf2txt.py ***--outfile*** *c:\work\res.txt c:\work\file_to_read.pdf* *another example of a code command:* *python pdf2txt.py* ***-o*** *c:\work\res.txt c:\work\file_to_read.pdf*	
2	-t or --output_type	Allows you to define the type of output that will be generated: “text", “html”, “xml”, “tag”
	python pdf2txt.py -o c:\work\res.html ***-t*** *html c:\work\file_to_read.pdf*	
3	-c or --codec	Here we need to specify which text encoding we will use in the output file
4	-O or --output-dir	We use this command to specify the name of the output directory in which the images extracted from the pdf file will be saved. If the command is not specified with the directory name, then the images will not be extracted
	python pdf2txt.py -o c:\work\res.html -t html ***-O*** *c:\work\image_folder c:\work\file_with_image.pdf* *another example of a code command:* *python pdf2txt.py -o c:\work\res.txt* ***-O*** *c:\work\image_folder c:\work\file_with_image.pdf*	
5	-Y or --layoutmode	Defines the type of page layout to be used when creating a file in html format {normal, exact, loose}. If it is “normal”, then each line is placed

		separately in html. If it is “exact”, then each character is positioned separately in html. If it is “loose”, then we get the same result as “normal”, with a difference in an additional new line after each line of text. This key is used only when the *«output_type»* type is html.
6	-s or --scale	The zoom change parameter used when creating an HTML file. Like the previous one, this key is used only when the *«output_type»* type is html
7	-S or --strip-control	This allows you to remove the control operator from the text. This key is used only when the *«output_type»* type is xml.

File to extract (see Figure 2.1)		
1	files [files ...]	In the command, we must, at least and without fail, enter the path to the PDF file. You can specify one or more PDF files.

Line 1

Line 2

Picture 1

Line 3

Picture 2

Line 4

Figure 2.1. example of a file («*file_to_read.pdf*») with the contents in PDF format.

b) Launch the terminal (see Figure 2.2)

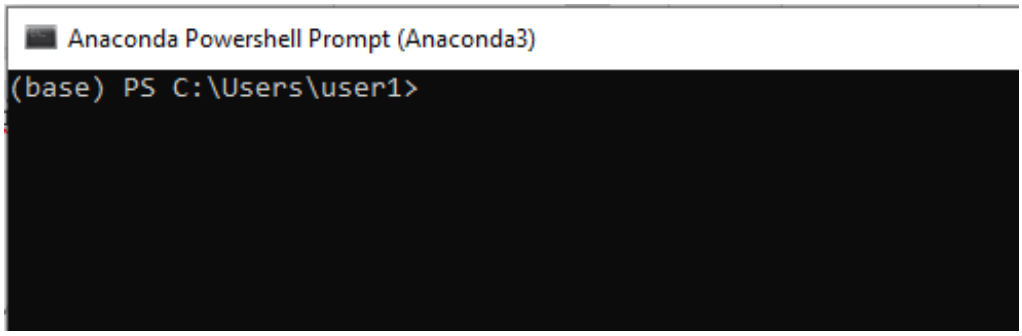


Figure 2.2. One of the Windows operating system terminals

c) Write the following commands in the command line:

```
python  pdf2txt.py  file_to_read.pdf
```

After executing this command, we will see how the text located in the *file_to_read.pdf* file will be displayed on the screen.

For proper operation, the file «*file_to_read.pdf*» must be located in the directory «user1» for this particular example, otherwise you will need to specify the full path to the file «*file_to_read.pdf*» if it is not in this directory. In the same case, if the operating system cannot automatically find the file «***pdf2txt.py***», then it is necessary to specify to the command to execute the full path to the file in question.

```
python pdf2txt.py --outfile c:\work\res.txt c:\work\file_to_read.pdf
```

another way we can use:

```
python pdf2txt.py -o c:\work\res.txt c:\work\file_to_read.pdf
```

After executing the command, a new file will be created «*res.txt*», if it is not present; and information will be saved inside the file in the form of text (see Figure 2.3).

Line 1
Line 2
Line 3
Line 4

Figure 2.3. New file «res.txt» without images in TXT format

python pdf2txt.py -o c:\work\res.html ***-t*** *html c:\work\file_to_read.pdf*

After executing the command, a new file will be created « *res.html* » if there is none; and inside the file will be stored information identical to the one shown in Figure 2.3, but in HTML format.

*python pdf2txt.py -o c:\work\res.html -t html –**O***
c:\work\image_folder c:\work\file_to_read.pdf

another way we can use:

python pdf2txt.py -o c:\work\res.txt ***-O***
c:\work\image_folder c:\work\file_to_read.pdf

After executing the command, a new file will be created « *res.html* », if there is none; and inside the file will be stored information with the same content as shown in Figure 2.3, but in HTML format. In addition, a directory called «*image_folder*» will be created, where all the images from the input file «*file_to_read.pdf*» will be located.

Note. If the operating system cannot find the file automatically «pdf2txt***.py***», then it is necessary to specify the full path to the file in the executed command.

Tool **dumppdf.py** – this is a software that is used on the command line and can extract the internal structure of a PDF file and display it on the screen where it was launched. This tool is designed and useful also for anyone who processes PDF files.

How to run it or use it?

a) familiarize yourself with the syntax of the command, which looks like this:

dumppdf.py *[-h] [--version] [--debug] [--extract-toc | --extract-embedded EXTRACT_EMBEDDED] [--page-numbers PAGE_NUMBERS [PAGE_NUMBERS ...]] [--pagenos PAGENOS] [--objects OBJECTS] [--all] [--password PASSWORD] [--outfile OUTFILE] [--raw-stream | --binary-stream | --text-stream] files [files*

Description of parameters (see Table 2.2)

Keys or parameters are data that must be entered to control the behavior of the library.

Table 2.2 – Explanation of the parameters of the *dumppdf* tool

№	Arguments of the **dumppdf** command	Explanation
0	[ ]	What is inside the brackets may or may not be used when executing the command

Continuation 1 of Table 2.2

Optional keys or arguments		
1	-h or --help	Allows you to get help, i.e. information about the syntactic description of the command
2	-v or --version	Allows you to determine the software version
3	-d or --debug	Used for debugging logging level
4	-T or --extract-toc	It allows you to extract the structure from the contour
5	-E EXTRACT_EMBEDDED or --extract-embedded EXTRACT_EMBEDDED	Allows you to extract embedded files
Keys or arguments used when analyzing a PDF file		
1	--page-numbers PAGE_NUMBERS [PAGE_NUMBERS ...]	This key is useful when you have a selected list of pages to analyze. Here we can specify only those pages that need to be analyzed. These page numbers will be separated by spaces
2	--pagenos PAGENOS, -p PAGENOS	Allows you to select a page to scan. Uses page numbers to enter arguments more clearly. You can use a comma-separated list of page numbers
	*python dumppdf.py **--pagenos** 2 -o c:\work\res.txt c:\work\file_to_read.pdf*	

Continuation 2 of Table 2.2

3	-i OBJECTS or --objects OBJECTS	It is used to extract information about certain objects. If you need to analyze or extract several objects, then the object numbers will be separated by commas
	*python dumppdf.py **--objects** 1 -o c:\work\res.txt c:\work\file_to_read.pdf* another way we can use: *python dumppdf.py **-i** 1 -o c:\work\res.txt c:\work\file_to_read.pdf*	
4	-a or --all	Allows you to get all the information about all objects in a PDF file. We accept the structure of the entire file for extraction
	*python dumppdf.py -o c:\work\res.txt **--all** c:\work\file_to_read.pdf* another way we can use: *python dumppdf.py -o c:\work\res.txt **-a** c:\work\file_to_read.pdf*	
5	-P PASSWORD or --password PASSWORD,	The password used to decrypt the PDF file, if it exists
Used when creating information output		
1	-o or --outfile	Allows you to specify the path to the file where the information obtained from the PDF file will be located
2	-r or --raw-stream	Allows you to record stream objects without encoding
3	-b or --binary-stream	Allows you to write stream objects with binary encoding

End of table 2.2

4	-t or --text-stream	Allows you to write stream objects in plain text

File to extract (see Figure 2.1)		
1	files [files ...]	The command must at least and without fail enter the path to the PDF file. There are one or more PDF files available.

b) launch the terminal (see previous Figure 2.2)

c) write one of the commands discussed earlier in the command line:

python dumppdf.py -o c:\work\res.txt ***--all*** *c:\work\file_to_read.pdf*

autre version que nous pouvons utiliser:

python dumppdf.py -o c:\work\res.txt ***-a*** *c:\work\file_to_read.pdf*

After executing the command, a new file will be created « *res.txt* », if there is none; and the entire internal structure of the PDF file « *file_to_read.pdf* » will be saved to the file (see Figure 2.3).

If we remove the command « *-o c:\work\res.txt* ***–all***» or so « *-o c:\work\res.txt* ***-a*** », then on the screen the result of the command execution is displayed.

```
<trailer>
<dict size="6">
<key>Size</key>
<value><number>194</number></value>
<key>Root</key>
<value><ref id="1" /></value>
<key>Info</key>
<value><ref id="61" /></value>
<key>ID</key>
<value><list size="2">
<string size="16">&#140;&#186;&#190;&#250; ...</string>
<string size="16">&#140;&#186;&#190;&#250; ... </string>
</list></value>
<key>Prev</key>
<value><number>829799</number></value>
<key>XRefStm</key>
<value><number>829112</number></value>
</dict>
</trailer>

<trailer>
<dict size="8">
<key>Type</key>
...
```

Figure 2.3. Part of the result of executing the previous command – The structure of the PDF file

python dumppdf.py ***--pagenos*** *2 -o c:\work\res.txt c:\work\file_to_read.pdf*

After executing this command, we will see a more detailed analysis of the arguments on page 2 of the internal structure of the PDF file «*file_to_read.pdf*» (see Figure 2.3); the result will be saved in the file «*res.txt*».

After executing the code, we will get in the file "res.txt " information about the first object from the PDF file «*file_to_read.pdf*».

python dumppdf.py ***--objects*** *1 -o c:\work\res.txt c:\work\file_to_read.pdf*

Note. If the operating system cannot find the file automatically «***dumppdf.py***», then it is necessary to specify in the executed command the full path to the file in question (see Figures 2.4 and 2.5). These files in the Windows operating system are usually located in the «Script» directory where the Python interpreter was installed.

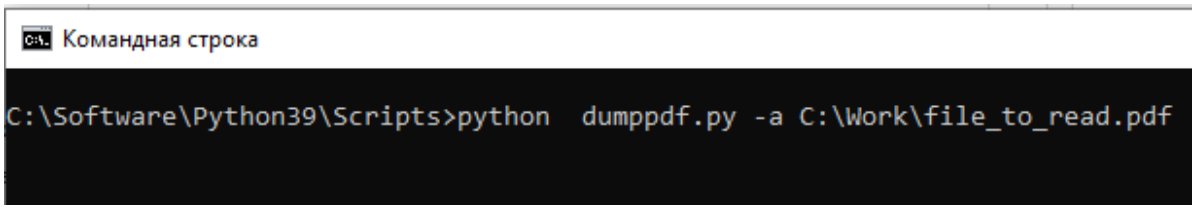


Figure 2.4. The first version of the command executed in Windows

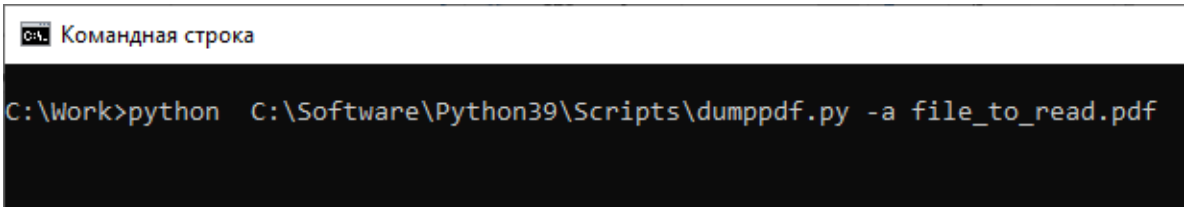


Figure 2.5. The second version of the command executed in Windows

The result of executing the previous command is shown in Figure 2.6.

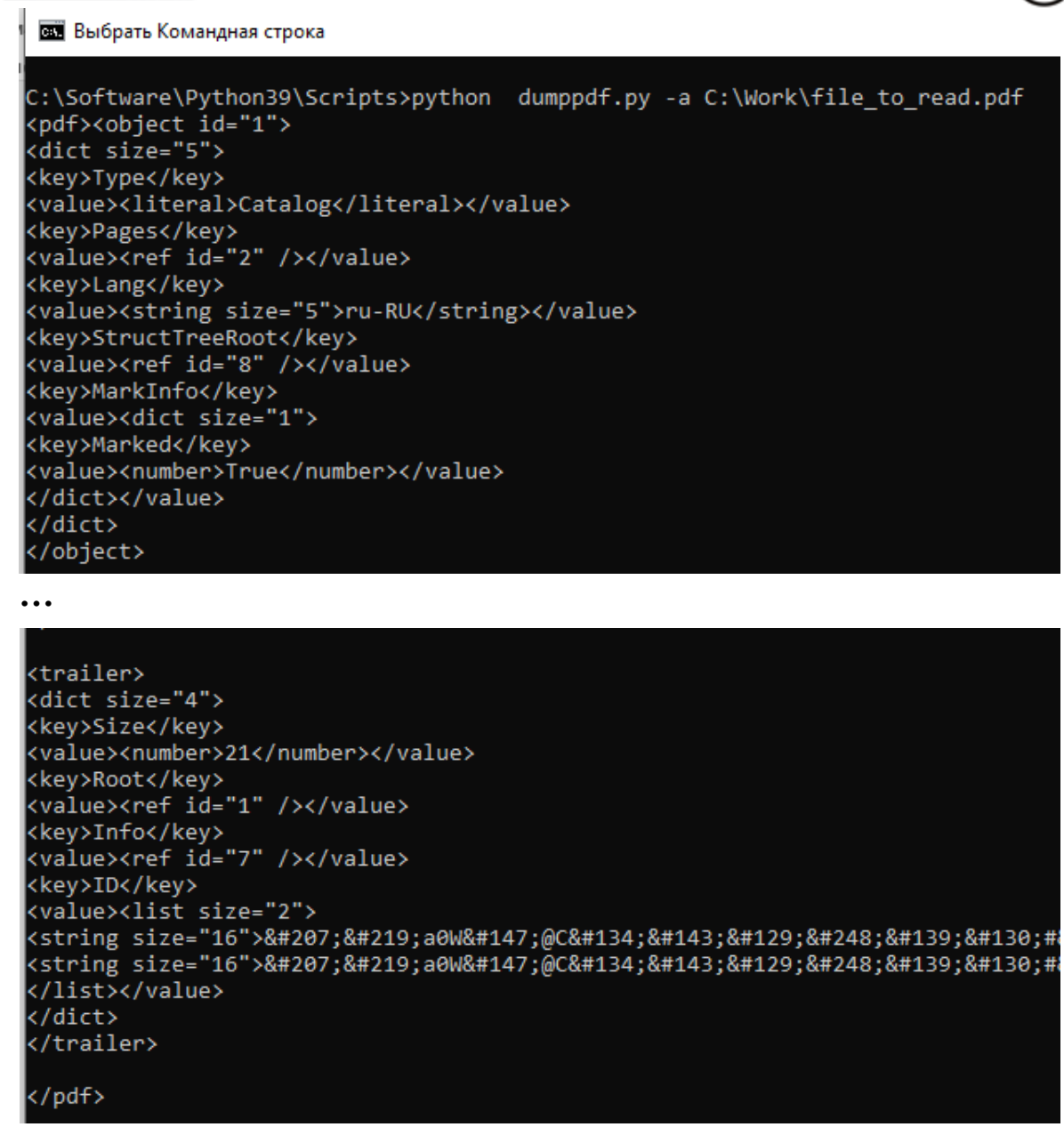


Figure 2.6. Part of the result of executing the previous command

Continued on page 177


FR Chapitre II. INTERACTION AVEC LES FICHIERS PDF DANS PYTHON

FR II.1. La bibliotheque PDFplumber – lire et/ou écrire dans un fichier PDF

PDFplumber est un instrument que nous pouvons utiliser dans Python pour lire et extraire un texte ou autres informations d'un document PDF. Le module ***PDFplumber*** a une particularité effective dans l'extraction des informations dans le document PDF. Cette particularité visuelle est axée sur le formatage. Ce module a à l'interieur, les paquets suivants: pycryptodome (Paquet cryptographique de bas niveau), Wand (permet de manipuler les images), pdfminer.six (outil d'extraction d'informations du fichier PDF), PDFplumber (cette bibliothèque permet de traiter des informations dans un ficher au format PDF.)

a) Séquences des étapes pour l'utilisation du module ***PDFplumber*** dans le but d'extraire une information (texte par exemple)

Étape №1 – Installation de la bibliothèque s'il n'a pas encore été fait ou alors qu'il ne se trouve pas dans le réservoir des modudes du rédacteur Python.

Méthode 1: pip install PDFplumber

Il fonctionnera dans l'environnement intégré IDLE

Méthode 2: si vous utilisez ***Anaconda***, alors il faut procéder selon les instructions si après pour que le module puisse fonctionner à partir du rédacteur «***Spyder***» d'***Anaconda***.

IIIar 1 – démarrer Anaconda Navigator

IIIar 2 – lancer ***Prompt CMD.exe*** ou alors ***PowerShell***

Шаг 3 – Écrire cette commande «*pip install pdfplumber*» sur le terminale d'exécution et lancer la commande.

Étape №2 – importation du module (bibliothèque) en question

```
import  PDFplumber
```

b) Exemple de code dans Python pour extraire du texte à partir d'un fichier PDF

```
1  import PDFplumber
2
3  with PDFplumber.open("document_path.PDF") as temp:
4      first_page = temp.pages[0]
5      print(first_page.extract_text())
```

Pour télécharger un fichier PDF qui est protégé par un mot de passe, il faudra introduire à l'argument «password» un mot secret quelconque. Par exemple, nous aurons la commande suivante:

```
pdfplumber.open("nom_fichier.pdf", password = "mot_secret")
```

FR II.2. La bibliotheque PDFminer.six – lire et/ou écrire dans un fichier PDF

PDFminer.six [2] est aussi un instrument que nous pouvons utiliser dans Python pour lire et extraire un texte ou autres informations (obtenir certaines données de la police, obtenir la couleur du texte, l'extraction des images, et d'autres) d'un document PDF. Essayons de voir les différentes possibilités de ce module (comment l'installer et utiliser) en quelques points.

a) Séquences des étapes pour l'utilisation du module ***PDFminer.six*** dans le but d'extraire une information (texte par exemple).

Procédure à parcourir : Installation de la dite bibliothèque dans le cas où il ne se trouve pas dans le réservoir des modudes du rédacteur Python.

```
Méthode 1: pip install  PDFminer.six          (avec windows)
```

Il est possible de vérifier si le package ***PDFminer.six*** a été installé dans l'interpretateur du langage Python. Nous pouvons chercher à lire, soit la version du package par l'attribut __version__, soit son nom par l'attribut __name__, ou soit son nom du fichier principale y compris le chemin où il a été installé par l'attribut __file__. Par exemple:

```
import pdfminer
print('Le fichier est: ',pdfminer.__file__)
```

Si vous ne rencontrez pas d'erreurs, alors dans ce cas, il est possible d'utiliser pdfminer.six en tant que package du langage Python. Seulement, il faudra savoir que le package **PDFminer.six** s'installe avec quelques outils de ligne de commande utiles. Pour savoir si ces outils sont correctement installés, il faudra faire la vérification suivante en exécutant la commande suivante sur un terminale:

Le fichier se trouve dans répertoire du disque où est installé l'interprétateur du langage Python.

```
pdf2txt.py --version
```

Ainsi, le résultat doit ressembler à l'affiche suivante:

```
pdfminer.six 20200517
```

Version installée

Comme nous l'avions dit un peu plus haut, le module **pdfminer.six** a également des outils qui peuvent être utilisés à partir de la ligne de

commande. À travers ces outils, certains utilisateurs peuvent extraire du texte d'un fichier *PDF*. Essayons d'étudier quelques un de ces outils:

L'outil ***pdf2txt*** – est un logiciel qui s'utilise à la ligne de commande et peut extraire les informations en forme de texte simple d'un fichier PDF et l'afficher sur l'écran (ou alors dans d'autres fichiers) où il a été exécuté. Le fichier contient des codes qui peuvent être consultés pour satisfaire sa curiosité ou alors pour l'acquisition de certaines connaissances dans la programmation.

Comment l'exécuter ou l'utiliser?

a) La syntaxe de la commande est la suivante:

python Scripts/pdf2txt.py [-h] [--version] [--debug] [--disable-caching] [--page-numbers PAGE_NUMBERS [PAGE_NUMBERS ...]] [--pagenos PAGENOS] [--maxpages MAXPAGES] [--password PASSWORD] [--rotation ROTATION] [--no-laparams] [--detect-vertical] [--char-margin CHAR_MARGIN] [--word-margin WORD_MARGIN] [--line-margin LINE_MARGIN] [--boxes-flow BOXES_FLOW] [--all-texts] [--outfile OUTFILE] [--output_type OUTPUT_TYPE] [--codec CODEC] [--output-dir OUTPUT_DIR] [--layoutmode LAYOUTMODE] [--scale SCALE] [--strip-control] files [files ...]

Explication des paramètres positionnels (voir tableau 2.1)

Les clés ou les mots en majuscule sont les données qu'il faut introduire pour achever le paramètre affecté.

Tableau 2.1 – Explication des paramètres de l'outil *pdf2txt*

№	Les arguments de la commande **pdf2txt**	Explication, rôle
0	[ ]	Ce qui est à l'intérieur des parenthèses peut être ou peut également ne pas être utilisé lors de l'exécution de la commande

Suite 1 du Tableau 2.1

1	-h ou --help	Recevoir de l'aide c'est-à-dire les informations sur la description syntaxique de la commande en anglais.
2	-v ou --version	Déterminer la version
3	-d ou --debug	Permet d'utiliser le niveau de journalisation de débogage (Par défaut : False = Faux)
4	-C ou --disable-caching	Si nous avons la mise en cache ou les ressources, telles que les polices, alors elles doivent être désactivées. Par défaut, nous avons la valeur "False".
Arguments utilisés pendant l'analyse du fichier au format PDF		
1	--page-numbers	Cette clé est utile lorsque nous avons une liste sélectionnée de pages à analyser. Ici nous pouvons indiquer juste les pages à analyser. Ces numéros de page seront séparés par des espaces.
2	-p PAGENOS ou --pagenos PAGENOS	Une liste de numéros de pages séparés par des virgules à analyser
3	-m MAXPAGES ou --maxpages MAXPAGES	Le nombre maximal de pages à analyser
4	-P ou --password PASSWORD	Le mot de passe à utiliser pour décoder le fichier au format PDF

Suite 2 du Tableau 2.1

4	-R ROTATION ou --rotation ROTATION,	Le nombre de degrés pour faire pivoter le fichier au format PDF avant d'autres types de traitement
Les paramètres utilisé lors de l'analyse de la mise en page		
1	-n ou --no-laparams	Est utilisé, si les paramètres d'analyse de mise en page doivent être ignorés. Par défaut, elle est égale à *False*
2	-V ou --detect-vertical	Est utilisé, si le texte vertical doit être pris en compte lors de l'analyse de la mise en page
3	-M CHAR_MARGIN ou --char-margin CHAR_MARGIN	Si deux caractères sont plus proches que la marge, ils sont considérés comme faisant partie de la même ligne. La marge est spécifiée par rapport à la largeur du caractère. Par défaut, elle est égale à 2.0
4	-W WORD_MARGIN ou --word-margin WORD_MARGIN	Si deux caractères sur la même ligne sont plus éloignés que la marge, ils sont considérés comme deux mots distincts et un espace intermédiaire sera ajouté pour plus de lisibilité. Notons que la marge est spécifiée par rapport à la largeur du caractère. Par défaut, elle est égale à 0,1

Suite 3 du Tableau 2.1

5	-L LINE_MARGIN ou --line-margin LINE_MARGIN	Si deux lignes sont proches l'une de l'autre, elles sont considérées comme faisant partie du même paragraphe. Notons que la marge est spécifiée par rapport à la hauteur d'une ligne. Par défaut, elle est égale à 0,5
6	-F BOXES_FLOW ou --boxes-flow BOXES_FLOW	Spécifie l'importance d'une position horizontale et verticale d'un texte lors de la détermination de l'ordre des lignes. La valeur doit être comprise entre -1,0 (seule la position horizontale compte) et +1,0 (seule la
6	-F BOXES_FLOW ou --boxes-flow BOXES_FLOW	position verticale compte). Par défaut, elle est égale à 0,5
7	-A ou --all-texts	Si l'analyse de la mise en page du fichier doit être effectuée sur un texte en figures
Est utilisé pendant la génération de la sortie d'informations		
1	-o ou --outfile	Permet d'indiquer le chemin d'accès au fichier où les informations obtenues du fichier pdf y seront
	python pdf2txt.py ***--outfile*** *c:\work\res.txt c:\work\file_to_read.pdf* *autre exemple de commande:* *python pdf2txt.py* ***-o*** *c:\work\res.txt c:\work\file_to_read.pdf*	
2	-t ou --output_type	Il permet de fixer le type de sortie à générer parmi : "text", "html", "xml", "tag"

Suite 4 du Tableau 2.1

	python pdf2txt.py -o c:\work\res.html ***-t*** *html c:\work\file_to_read.pdf*	
3	-c ou --codec	Ici, nous avons à indiquer l'encodage de texte à utiliser dans le fichier de sortie.
4	-O ou --output-dir	Nous utilisons cette commande pour fixer le nom du répertoire de sortie dans lequel les images extraites du fichier pdf seront envoyées. Si la commande n'est pas indiquée avec le nom du répertoire, alors les images ne seront pas extraites.
	python pdf2txt.py -o c:\work\res.html -t html ***-O*** *c:\work\image_folder c:\work\file_with_image.pdf* *autre exemple de commande:* *python pdf2txt.py -o c:\work\res.txt* ***-O*** *c:\work\image_folder c:\work\file_with_image.pdf*	
5	-Y ou --layoutmode	Permet de déterminer le type de mise en page à utiliser pendant la génération du fichier à format html {normal, exact, lâche(loose)}. Si c'est «*normal*», chaque ligne est positionnée séparément dans le code html. Si c'est «*exact*», chaque caractère est positionné séparément dans le code html. S'il c'est lâche(«*loose*»), nous avons le même résultat que la normale, avec la différence d'une nouvelle ligne supplémentaire après chaque ligne de texte. Cette clé est utilisée si et seulement si le type «*output_type*» est html.

Fin du Tableau 2.1

6	-s ou --scale	Le paramètre de la modification du zoom à utiliser lors de la génération d'un fichier HTML. De même comme le précédent, cette clé est utilisée si et seulement si le type «*output_type*» est html.
7	-S ou --strip-control	Il permet de supprimer l'instruction de contrôle du texte. cette clé est utilisée si et seulement si le type «output_type» est xml.
Fichier à extraire (voir figure 2.1)		
1	files [files ...]	Dans la commande, nous devons au minimum et obligatoirement introduire un chemin d'accès au fichier PDF. Nous avons un ou plusieurs fichiers au format PDF.

Ligne 1
Ligne 2
image 1
Ligne 3
image 2
Ligne 4

Fig. 2.1. Un exemple du fichier («*file_to_read.pdf*») avec contenu au format PDF

b) Lancer un terminal (voir figure 2.2)

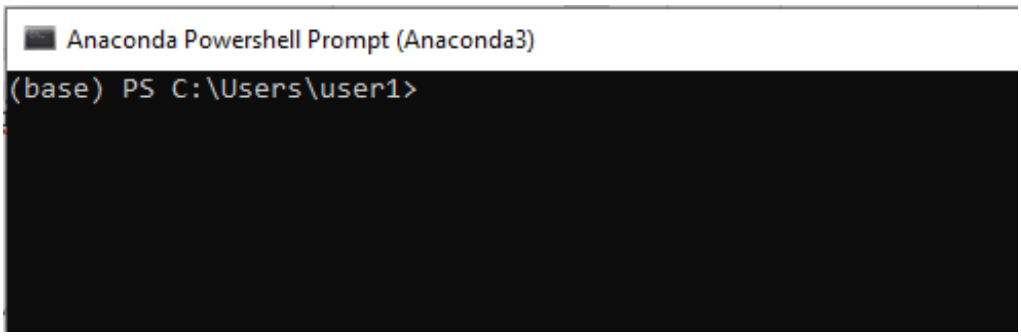


Fig. 2.2 Un des terminals du système opérationnel Windows

c) Écrire une des commandes suivantes parmi tant d'autres à la ligne de commande :

```
python  pdf2txt.py  file_to_read.pdf
```

Après l'exécution de cette commande, nous verrons afficher sur l'écran le texte qui se trouve dans le fichier *file_to_read.pdf*.

Pour son bon fonctionnement, le fichier « *file_to_read.pdf* » doit être dans le répertoire «user1» pour cet exemple précis, ou alors il faudra indiquer le cheminement complet du fichier « *file_to_read.pdf* » s'il ne se trouve pas dans ce répertoire. Dans le même cas, si le système opérationnel n'arrive pas à trouver automatiquement le fichier «***pdf2txt.py***», alors il faut indiquer à la commande à exécuter le chemin complet au fichier en question.

```
python pdf2txt.py --outfile c:\work\res.txt c:\work\file_to_read.pdf
```

autre version que nous pouvons utiliser:

```
python pdf2txt.py -o c:\work\res.txt c:\work\file_to_read.pdf
```

Après l'exécution de cette nouvelle commande, nous allons observer la création d'un nouveau fichier « *res.txt* »s'il n'existe pas ; et à l'intérieur du fichier va s'enregistrer les informations en forme de texte (voir la figure 2.3).

Ligne 1
Ligne 2
Ligne 3
Ligne 4

Fig. 2.3. Le nouveau fichier « *res.txt* » sans images au format TXT

python pdf2txt.py -o c:\work\res.html ***-t*** *html c:\work\file_to_read.pdf*

Après l'exécution de cette nouvelle commande, nous allons observer la création d'un nouveau fichier « *res.html* »s'il n'existe pas ; et à l'intérieur du fichier seront stockées les informations identiques à celle de la figure 2.3, mais au format HTML.

python pdf2txt.py -o c:\work\res.html -t html ***–O*** *c:\work\image_folder c:\work\file_to_read.pdf*

autre version que nous pouvons utiliser:

python pdf2txt.py -o c:\work\res.txt ***-O*** *c:\work\image_folder c:\work\file_to_read.pdf*

Après l'exécution de cette nouvelle commande, nous allons observer la création d'un nouveau fichier « *res.html* »s'il n'existe pas ; et à l'intérieur du fichier seront stockées les informations avec contenu identique à celle de la figure 6, mais au format HTML. En outre, nous assisterons à la création d'un répertoire appelé « *image_folder* » où toutes les images du fichier d'entrée « *file_to_read.pdf* » se trouveront.

NB. Si le système opérationnel n'arrive pas à retrouver automatiquement le fichier «pdf2txt***.py***», alors il faut indiquer à la commande à executer, le chemin complet au fichier en question

L' outil **dumppdf.py** – est un logiciel qui s'utilise à la ligne de commande et peut extraire la structure interne d'un fichier PDF et l'afficher sur l'écran où il a été exécuté. Cet outil est destiné et utile à tous ceux qui manipulent les fichiers au format PDF.

Comment l'exécuter ou l'utiliser?

a) La syntaxe de la commande est la suivante:

dumppdf.py *[-h] [--version] [--debug] [--extract-toc | --extract-embedded EXTRACT_EMBEDDED] [--page-numbers PAGE_NUMBERS [PAGE_NUMBERS ...]] [--pagenos PAGENOS] [--objects OBJECTS] [--all] [--password PASSWORD] [--outfile OUTFILE] [--raw-stream | --binary-stream | --text-stream] files [files*

Explication des paramètres positionnels (voir Tabl. 2.2)

Les clés ou les mots en majuscule sont les données qu'il faut introduire pour achever les arguments désignés.

Tableau 2.2 – Explication des paramètres de l'outil *dumppdf*

№	Les arguments de la commande **dumppdf**	Explication, rôle
0	[ ]	Ce qui est à l'intérieur des crochets peu être utilisé, comme il peut ne pas être utile lors de l'exécution de la commande

Suite 1 du Tableau 2.2

Les clés ou arguments facultatifs		
1	-h ou --help	Permet de recevoir de l'aide c'est-à-dire les informations sur la description syntaxique de la commande
2	-v ou --version	Permet de déterminer la version du logiciel
3	-d ou --debug	Est utilisée pour le niveau de journalisation de débogage
4	-T ou --extract-toc	Il fait extraire la structure du contour
5	-E EXTRACT_EMBEDDED ou --extract-embedded EXTRACT_EMBEDDED	Permet d'extraire des fichiers intégrés
Les clés ou arguments qui sont utilisés lors de l'analyse du fichier au format PDF		
1	--page-numbers PAGE_NUMBERS [PAGE_NUMBERS ...]	Cette clé est utile lorsque nous avons une liste sélectionnée de pages à analyser. Ici nous pouvons indiquer juste les pages à analyser. Ces numéros de page seront séparés par des espaces.
2	--pagenos PAGENOS, -p PAGENOS	Permet de choisir la page à analyser. Utilise les numéros de page pour avoir une entrée d'argument plus articulée. Vous pouvez utiliser une liste de numéros de pages qui sera séparée par des virgules
	*python dumppdf.py **--pagenos** 2 -o c:\work\res.txt c:\work\file_to_read.pdf*	

Suite 2 du Tableau 2.2

3	-i OBJECTS ou --objects OBJECTS	Permet de récupérer les informations sur certains objets. Si plusieurs objets sont à analyser ou à extraire, alors les numéros des objets seront séparés par des virgules
	*python dumppdf.py **--objects** 1 -o c:\work\res.txt c:\work\file_to_read.pdf* autre version que nous pouvons utiliser: *python dumppdf.py **-i** 1 -o c:\work\res.txt c:\work\file_to_read.pdf*	
4	-a ou --all	Permet de récupérer toutes les informations sur tous les objets du fichier au format PDF. Nous assistons à l'extraction de la structure de tout le fichier
	*python dumppdf.py -o c:\work\res.txt **--all** c:\work\file_to_read.pdf* autre version que nous pouvons utiliser: *python dumppdf.py -o c:\work\res.txt **-a** c:\work\file_to_read.pdf*	
5	-P PASSWORD ou --password PASSWORD,	Le mot de passe à utiliser pour déchiffrer un fichier PDF s'il existe
Est utilisé pendant la génération de la sortie d'informations		
1	-o ou --outfile	Permet d'indiquer le chemin d'accès au fichier où les informations obtenues du fichier PDF y seront
2	-r ou --raw-stream	Permet d'écrire des objets de flux sans encodage

Fin du Tableau 2.2

3	-b ou --binary-stream	Permet d'écrire des objets de flux avec un codage binaire
4	-t ou --text-stream	Permet d'écrire des objets de flux en texte brut
Fichier à extraire (voir figure 2.1)		
1	files [files ...]	Dans la commande, nous devons au minimum et obligatoirement introduire un chemin d'accès au fichier PDF. Nous avons un ou plusieurs fichiers au format PDF.

b) Lancer un terminal (voir la figure 2.2 précédente)

c) Écrire une des commandes suivantes parmi tant d'autres à la ligne de commande :

python dumppdf.py -o c:\work\res.txt ***--all*** *c:\work\file_to_read.pdf*

autre version que nous pouvons utiliser:

python dumppdf.py -o c:\work\res.txt ***-a*** *c:\work\file_to_read.pdf*

Après l'exécution de cette nouvelle commande, nous allons observer la création d'un nouveau fichier « *res.txt* »s'il n'existe pas ; et à l'intérieur du fichier nous allons assister au stoquage de toute la structure interne du fichier PDF « *file_to_read.pdf* » (voir la figure 2.3)

Si nous enlevons la commande « *-o c:\work\res.txt* ***–all***» ou alors « *-o c:\work\res.txt* ***-a*** », alors le résultat de l'exécution de la commande va s'afficher sur l'écran.

```
<trailer>
<dict size="6">
<key>Size</key>
<value><number>194</number></value>
<key>Root</key>
<value><ref id="1" /></value>
<key>Info</key>
<value><ref id="61" /></value>
<key>ID</key>
<value><list size="2">
<string size="16">&#140;&#186;&#190;&#250; ...</string>
<string size="16">&#140;&#186;&#190;&#250; ... </string>
</list></value>
<key>Prev</key>
<value><number>829799</number></value>
<key>XRefStm</key>
<value><number>829112</number></value>
</dict>
</trailer>

<trailer>
<dict size="8">
<key>Type</key>
...
```

Fig. 2.3. Une partie du résultat de l'exécution de la commande précédente – la structure du fichuer PDF

python dumppdf.py ***--pagenos*** *2 -o c:\work\res.txt c:\work\file_to_read.pdf*

Après l'exécution de cette nouvelle commande, nous allons assister à l'analyse plus articulé uniquement d'argument à la page 2 de la structure interne du fichier PDF « file_to_read.pdf » (voir la figure 2.3) ; *le résultat est stocké dans le fichier « res.txt ».*

Après l'exécution de cette nouvelle commande, nous aurons dans le fichier « *res.txt* » les informations sur le premier objet venant du fichier PDF « *file_to_read.pdf* ».

python dumppdf.py ***--objects*** *1 -o c:\work\res.txt c:\work\file_to_read.pdf*

NB. Si le système opérationnel n'arrive pas à retrouver automatiquement le fichier «***dumppdf.py***», alors il faut indiquer à la commande à executer, le chemin complet au fichier en question (voir figure 2.4 et 2.5). Ces fichiers dans le système opérationnel Windows se trouvent en principe dans le catalogue «Script» où a été installé l'interprétateur du langage Python.

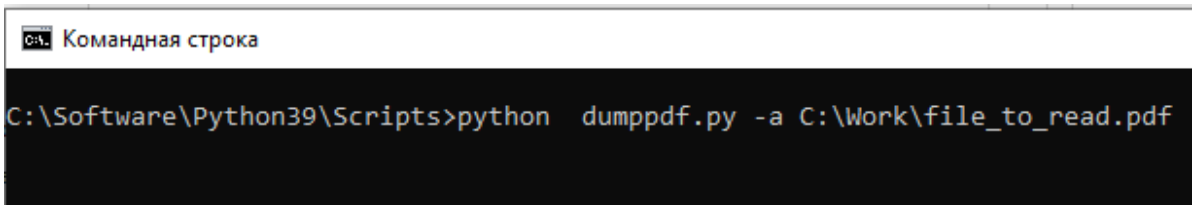


Fig. 2.4. Variante 1 de la commande à exécuter dans Windows

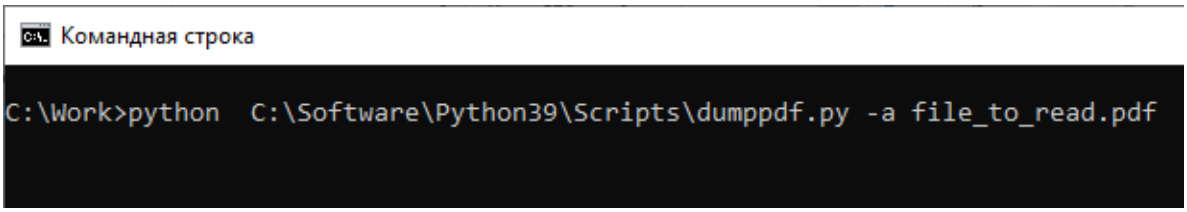


Fig. 2.5. Variante 2 de la commande à exécuter dans Windows

Le résultat de l'exécution de la commande précédente se trouve à la figure 2.6.

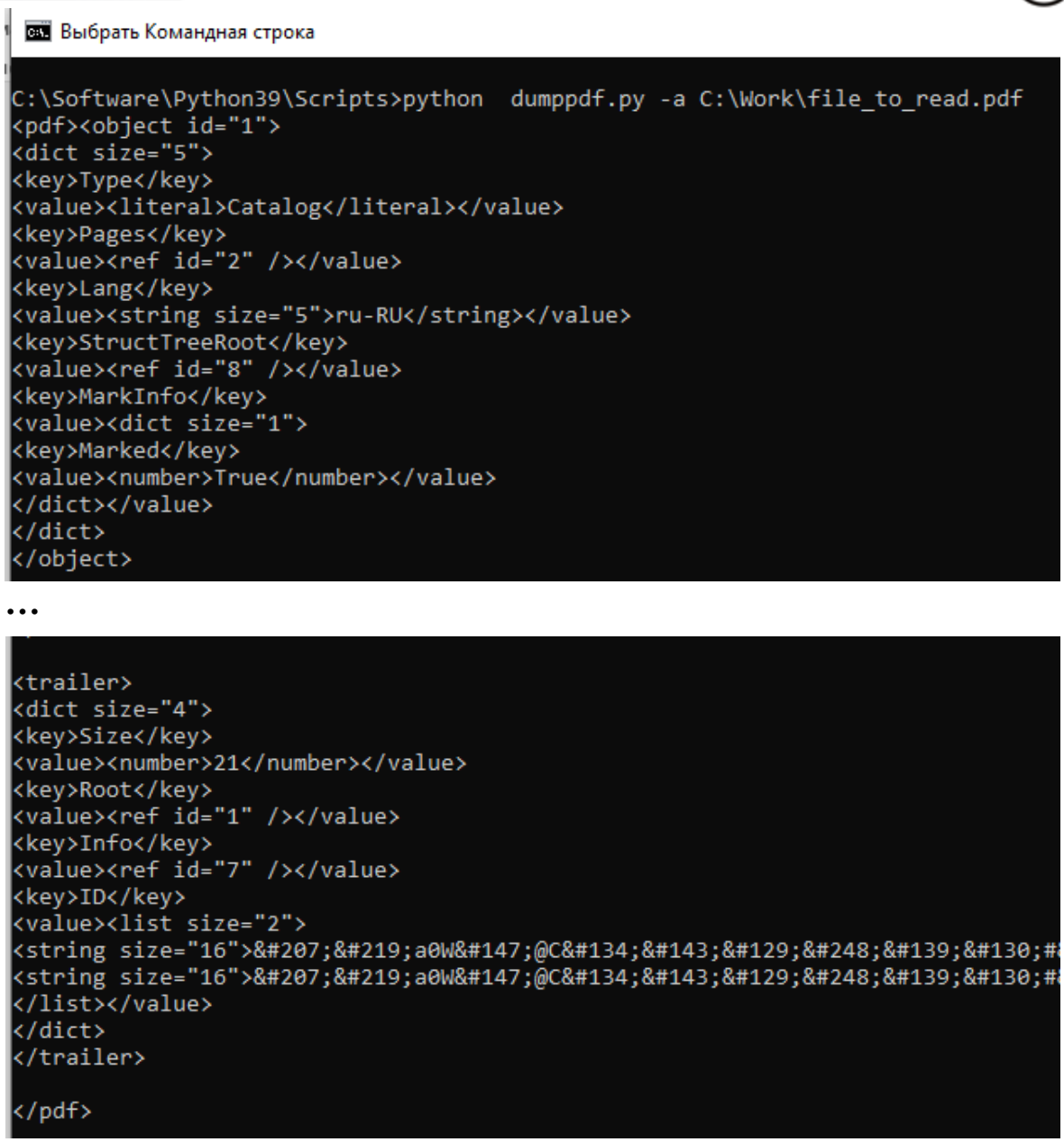


Fig. 2.6. Une partie du résultat de l'exécution de la commande précédente

Suite à la page 215


RU Глава III. ОБРАБОТКА СТРОК В PYTHON

RU Введение

Строка символов может определяться как упорядоченная последовательность символов. Его можно рассматривать как простые символы, в которых у каждого из них есть номер (для обработки) или конкретный код (для сохранения своего значения). Как мы уже говорили ранее, различаем два типа строк: первый тип строк, ограниченный 256 символами, где каждый символ имеет уникальный код в зависимости от системы кодирования; второй тип кодирования имеет только 2 типа символов, которые равны 0 и 1 – это действительно то, что имеем в виду по строкам байтов. Следует отметить, что каждый символ представлен числом байтов. Это число также зависит от используемого кодирования.

RU III.1. Процесс кодирования символов

Учитывая, что в мире существует несколько языков, растёт количество данных за счет использования компьютеров, и чтобы данные были кодифицированы в виде текста используют во всем мире созданные единые таблицы кодирования (одна часть находится в таблице 3.1). Каждому языку соответствует определенный тип или типы кодирования. Например, с целью визуального и безошибочного отображения русского текста будет правильным использовать кириллические кодировки. Рассмотрим практические примеры того, как в языке программирования, кодировки определяются с помощью двух небольших функций, а именно: *chr()* и *ord()*. В наших современных браузерах кодировка выбирается автоматически в зависимости от используемого или написанного языка. Это означает, что вмешательство пользователя было заранее устранено.

Таким образом, мы можем сказать, что кодировка символов – это способ обеспечить соответствие между тем, что пользователь видит на экране, и тем, что машина(компьютер) фактически хранит в памяти. Библиотека «chardet» – это модуль, который позволяет автоматически определять кодировку текста. Эта основная часть кода для автоматического обнаружения кодировки в браузере, известном как «Mozilla». Чтобы использовать его, его необходимо установить.

Таблица 3.1 – Таблица кодов ASCII и их соответствия

Кодировка ASCII	
Значение кода	Символ
32	space
33	!
…	…
48	0
49	1
50	2
…	…
97	a
98	b
…	
120	x
121	y
122	z
…	…
127	delete

Эта кодировка используется на большинстве локальных компьютеров, работающих на платформе Windows

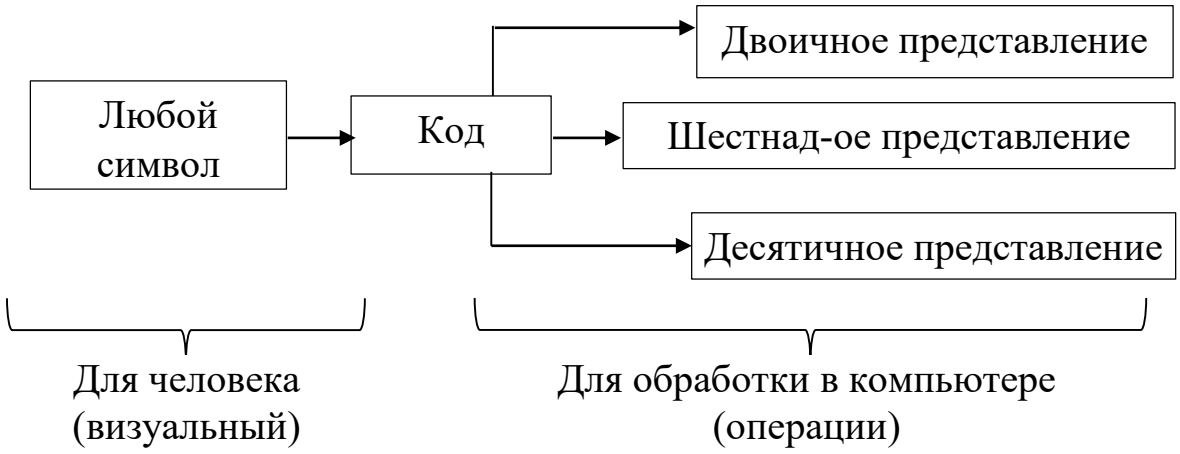


Рис. 3.1. Представление символа

Код уникален в любом используемом стандарте. Отметим, что расширенная таблица кода ASCII будет варьироваться в зависимости от используемой кодировки символов.

RU III.2. Управление клавиатурой компьютера

Для управления клавиатурой можем использовать различные модули, которые будут импортированы в код программы.

Модуль keyboard

С помощью этой библиотеки мы можем полностью контролировать клавиатуру. Мы можем создавать сочетания клавиш, делать записи, блокировать клавиатуру, ждать ввода данных и многие другие важные функции. Давайте попробуем понять, как с помощью этого модуля мы можем обнаруживать нажатия клавиш на клавиатуре. Здесь можно использовать три способа обнаружения этих нажатий клавиш на клавиатуре: *«read_key()», «is_pressed()»* и *«on_press_key()»*.

Вариант 1. Использование функции *«read_key()»*

Функция *«read_key()»* позволяет считывать, какую кнопку на клавиатуре нажал пользователь, то есть мы сравниваем функцию *«read_key()»* (которая возвращает символ) с символом клавиатуры.

Пример использования

Определить код ASCII нажатой клавиши на клавиатуре.

Дополнительная информация:

chr(код) – позволяет получить символ с помощью кода ASCII;

ord(символ) – позволяет получить код ASCII через его символ.

```
1  import keyboard, time
2
3  while True:
4      symbol = keyboard.read_key()
5      if len(str(symbol))>1 :
6          print("Клавиша «{0}» не имеет код".format(symbol))
7      else:
8          print("Код «{0}» соответствует «{1}»".format(symbol,
9                                                       ord(symbol)))
10     if symbol == "esc":
11         break
12     time.sleep(0.25)
```

Результат выполнения этой программы позволяет подтвердить и проверить данные, приведенные в таблице 3.1.

Комментарий к предыдущим кодам

#1 импорт модуля «keyboard» для определения ASCII-кода нажатия клавиши на клавиатуре, также импорт модуля «time» для добавления задержки в программу.
#3 происходит, если условие (инструкции) истинно.
#4 подождите, пока будет нажата клавиша на клавиатуре.
#5 определяет, является ли он контрольным символом, проверяя, строго ли его длина больше 1.
#6 печатает контрольный символ.
#7 Если это не контрольный символ.
#8 печатает нажатый символ клавиатуры и ее код ASCII.
#10 если на клавиатуре нажат символ Escape.
#11 Выход из цикла.
#12 добавляет время ожидания (1/4 секунды) для сенсорных клавиатур.

Вариант 2. Использование функции is_pressed()

Функция *is_pressed()* содержит входной параметр. Этот параметр считается любым символом, и если он соответствует кнопке, которую нажал пользователь, то в этом случае он вернет логическое значение «True», в противном случае он вернет значение «False».

Пример использования.

Определите код ASCII для клавиши, нажатой на клавиатуре

```
1  import keyboard
2  from time import sleep
3
4  print("Нажмите клавишу «Escape» чтобы завершения")
5  while True:
6      symbol = keyboard.read_key()
7      if keyboard.is_pressed("esc"):
8          print("Конец !")
9          break
10     elif len(symbol)>1:
11         print("Вы нажали кнопку управления {0}".format(symbol))
12     else :
13         print("Вы нажали на символ {0}".format(symbol))
14         print("Код «{0}» соответствует «{1}»".format(symbol,
15                                                      ord(symbol)))
16     sleep(0.5)
17
```

Комментарий к предыдущим кодам
#1 Импорт модуля «keyboard» для определения кода ASCII для нажатия клавиши на клавиатуре *#2 Импортировать функцию «sleep()», которая находится в модуле «time», чтобы добавить задержку в программу* *#5 Открытие цикла, если условие (инструкции) истинно* *#6 Использование функции «read_key()» для ввода символа с клавиатуры (ожидает нажатия клавиши на клавиатуре)* *#7 Проверяет, нажата ли клавиша «Escape»* *#9 Выход из цикла* *#10 Определяет, является ли он контрольным символом, проверяя, строго ли его длина больше 1* *#11 Печатает символ* *#13 Если это не символ* *#14 Выводит нажатый символ и его код ASCII* *#16 добавляет время ожидания (1/2 секунды) для сенсорных клавиатур (не обязательно).*

Вариант 3. Использование функции «on_press_key()»

Функция «on_press_key()» «использует в качестве входных данных два параметра: первый – это символ, который хотим получить или идентифицировать, а второй параметр – это функция типа «lambda _». В случае, если пользователь нажимает кнопку, соответствующую значению первого параметра, он будет выполнять только функцию, расположенную после «lambda _» второго параметра.

Пример использования

Необходимо показать полезность функции « *on_press_key()* ». Как можем использовать её по-другому.

```
1  import keyboard
2
3  def print_one_line():
4      print(' С помощью клавиши «esc», можно завершить.')
5
6  keyboard.on_press_key("esc", lambda _: imprime_une_phrase())
7  keyboard.on_press_key("s", lambda _: keyboard.write("Была
8                                       нажата клавиша «s»", delay = 0.1))
9  keyboard.on_press_key("z", lambda _: print("Была нажата
10                          клавиша «z»"))
```

Поскольку функция «keyboard» возвращает событие этой функции, необходимо использовать символ «_».

Комментарий к предыдущим кодам

#1 Импорт модуля «keyboard» для определения кода ASCII для нажатия клавиши на клавиатуре
#3 и 4 Функция для печати текста на экране

#6 Если нажата клавиша "Escape", то увидим выполнение функции " print_one_line ()"
#7 Если нажата клавиша «s», то наблюдаем выполнение функции «write» модуля «keyboard». Текст будет отображаться постепенно по символам с задержкой, которая будет определяться методом «delay()» функции «write()».
#9 если нажата клавиша «z», то наблюдаем выполнение функции «print». Разница между «print» и «write» заключается в том, что с помощью функции «write» в библиотеке «keyboard» текст печатается там, где находится курсор. Он может быть даже активен вне консоли, там, где была запущена программа.

Другие функции этого модуля:

keyboard.wait('esc')

Нажмите клавишу Esc, чтобы продолжить выполнение следующих инструкций. Вместо «esc», мы можем поместить туда любую клавишу на клавиатуре.

keyboard.press('caps_lock')

Позволяет программе нажимать клавишу «caps_lock» на клавиатуре. Вместо «caps_lock» мы можем поместить туда любую клавишу на клавиатуре.

keyboard.release('caps_lock')

Позволяет программе отпустить клавишу «caps_lock» на клавиатуре, которая была нажата.

keyboard.press_and_release('shift+s, space, =,space, 9')

Позволяет записать команду «S = 9». На их места мы можем поставить другие ключевые слова и символы.

keyboard.add_hotkey('enter', lambda: keyboard.write("Вы нажали клавишу «Enter»!!!"))

Если нажата клавиша «Enter», то мы наблюдаем выполнение функции «write» модуля « keyboard ». Вместо «keyboard.write», мы можем поместить в него другую функцию для выполнения. Точно так же вместо «enter» мы можем поставить другую клавишу или комбинацию клавиш: например, «shift+r» (для чего потребуется нажать как клавишу «shift», так и клавишу «r» одновременно)

keyboard.send('клавиша_клавиатуры')

Позволяет нажимать и отпускать клавишу, находящуюся в скобках функции.

keyboard.add_abbreviation(source_text, replacement_text, match_suffix = False, expiration_time = 2)

Позволяет ускорить или автоматизировать написание небольшого текста (***replacement_text***) с помощью ключевого слова (***source_text***) или символа. Вы вводите ключевое слово (символ), и после нажатия пробела оно напрямую заменяется другим, которое является « ***replacement_text*** ».

Пример : *keyboard.add_abbreviation('@ ', 'fiston@gmx.fr')*

list_ev = keyboard.record(until=' клавиша_клавиатуры')

Записывает все события клавиш на клавиатуре до тех пор, пока пользователь не нажмет указанную горячую клавишу (***touche_du_clavier***). После этого эти события сохраняются в списке (***list_ev***), который можно просмотреть или воспроизвести с помощью функции «play».

keyboard.play(list_ev, speed_factor=1.0)

Выводит все события, сохраняя относительные временные интервалы во время записи. Если параметр ***speed_factor*** меньше или равен **0**, то действия будут воспроизведены так быстро, как позволяет операционная система.

Пример использования « record » и « play »

```
1  import keyboard
2
3  try:
4     keyboard.write("Все, что вы собираетесь делать с
5                     клавиатурой, будет записано. Завершение
6                     записи произойдет после того, как можно будет
7                     нажать клавишу «escape». \n")
8     record = keyboard.record(until="esc", suppress=False,
9                     trigger_on_release=False)
10    print("Сохраненный список = ",record)
11    print("Точная копия того, что было записано")
12    keyboard.play(record)
13 except Exception as er:
14    print("У нас есть следующая ошибка: {0}".format(er.args[0]))
```

Комментарий к предыдущим кодам

#1 Импорт модуля «keyboard» для записи всех клавиш на клавиатуре.
#3 и 13 Используются для проверки кода, в котором могут быть ошибки
#4 Печатает текст с помощью функции write. Текст печатается там, где находится курсор. После того, как курсор переходит к строке, с помощью команды «\n »
 # 8 Записываются все события клавиш клавиатуры, пока пользователь не нажмет клавишу « esc ». Параметры « suppress» и «trigger_on_release» могут быть удалены.
#10 Печатает события, которые находятся в списке, и получаем результат на рисунке 2.
#12 Идентично показывает все события, которые были записаны.

Данные вводятся во время регистрации. >>> 12345
Saved_list = [KeyboardEvent(1 down), KeyboardEvent(1 up), KeyboardEvent(2 down), KeyboardEvent(2 up), KeyboardEvent(3 down), KeyboardEvent(3 up), KeyboardEvent(4 down), KeyboardEvent(4 up), KeyboardEvent(5 down), KeyboardEvent(5 up), KeyboardEvent(esc down)]
Результат воспроизведения записи в списке с помощью функции «*play()*» >>>12345

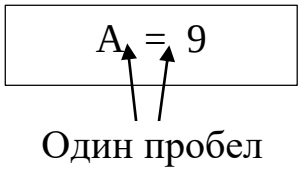
Рис. 3.2. Введенные данные и результаты записи, выполненные функцией «*record*» модуля «keyboard»

Модуль pynput

В сравнении с предыдущей библиотекой (keyboard) этот модуль является более функциональным и позволяет не только полностью контролировать клавиатуру, но и манипулировать движениями мыши по координатам. Давайте попробуем понять, как с помощью этого модуля мы можем обнаружить нажатия клавиш клавиатуры и мыши.

Как нажать клавишу и отпустить ее?

Метод « ***press*** » позволяет нажать любую клавишу, а метод « ***release*** » позволяет отпустить эту клавишу. В следующем примере рассмотрим, как их можно использовать для написания уравнения:



```
from pynput.keyboard import Key, Controller

try:
    keyboard = Controller()

    keyboard.press(Key.caps_lock)
    keyboard.press('a')
    keyboard.release('a')
    keyboard.press(Key.space)
    keyboard.release(Key.space)
    keyboard.press('=')
    keyboard.release('=')
    keyboard.press(Key.space)
    keyboard.release(Key.space)
    keyboard.press('9')
    keyboard.release('9')
    keyboard.release(Key.caps_lock)
except Exception as er:
    print("У нас есть следующая ошибка: {0}".format(er.args[0]))
```

Мы также знаем, что для написания заглавной буквы можно использовать клавишу «Shift» вместо «Caps lock». Таким образом, новая версия кода будет такой:

```
1  from pynput.keyboard import Key, Controller
2
3  try:
4      keyboard = Controller()
5
6      with keyboard.pressed(Key.shift):
7          keyboard.press('a')
8          keyboard.release('a')
9      keyboard.press(Key.space)
10     keyboard.release(Key.space)
11     keyboard.press('=')
12     keyboard.release('=')
13     keyboard.press(Key.space)
14     keyboard.release(Key.space)
15     keyboard.press('9')
16     keyboard.release('9')
17 except Exception as er:
18     print("Возникшие ошибки: ",er)
19
```

Комментарий к предыдущим кодам

#1 Импорт классов «Key» и «keyboard» из модуля «pynput.keyboard».
#4 Определяет « keyboard » как объектный контроллер клавиатуры, поступающий из модуля « pynput ».

#6 нажав клавишу «Shift», также нажимаем клавишу «a » (#7), а затем отпускаем ее (#8).
#9 ...

Как отобразить сообщение с помощью этого модуля?

```
1  from pynput.keyboard import Controller
2
3  le_clavier = Controller()
4  le_clavier.type("Любой текст!")
```

или

```
1  from pynput.keyboard import Controller
2
3  def ShowMessageKeyboard():
4      clavier = Controller()
5      clavier.type(" Любой текст!")
6
7  ShowMessageKeyboard()
```

Комментарий к предыдущим кодам
#4 ou #5 Мы просто получим следующее сообщение на экране: Любой текст!

Как визуально отобразить на консоли все нажатые кнопки с их кодами

Для этого нам нужно использовать библиотеку «Listeners» (слушатели), чтобы могли прослушивать нажатия клавиш.

```
from pynput.keyboard import Listener
def Reset_key(key):
    print("Клавиша %s была возвращена\n'%(key))

def WriteOnScreen (key):
    try:
        if str(key) == 'Key.esc' :
            return exit()
        print(' Вы нажали клавишу ',key)

        sym = str(key)
        if len(sym) == 3:
            sym = sym.replace("'","")
            val = str(ord(sym))
        else:
            sym = sym.rsplit('.')[1]
            sym = " "+sym+" "
            val = " None"

        print("Клавиша '% s' имеет значение кода '%s'" %(sym, val))
    except Exception as er:
        print("Имеем следующую ошибку: {0}".format(er.args[0]))

print("Нажмите на «esc», чтобы остановить программу!")
print("Нажмите любую клавишу, чтобы определить ее код. \n")

with Listener
            (on_press=WriteOnScreen, on_release=Reset_key) as lis:
    lis.join()
```

Комментарий к предыдущим кодам
#1 Импортируем клавиатура(keyboard) модуля «pynput» как объект типа слушателя (Listener) *#2 Функция отпускания нажатой клавиши* *#5 – 23 Функция для нажатия и определения его кода* *#7 Если нажмете клавишу «Escape», то программа должна остановиться (#9)* *#12 Проверяет, не является ли нажатая кнопка управляющей клавишей* *#13 Позволяет удалить апострофы или кавычки, которые сопровождают символ* *#14 Выводит код символа, который был нажат.* *#16 Разбивает строку «sym» на две части с помощью разделителя « . » и удерживает вторую часть цепочки.* *#17 Добавляет пробелы перед символом и после него, чтобы избежать визуальной путаницы с другими символами* *#18 Присвойте значение «None» кнопкам управления* *#20 Печать клавиши на экране(консоль)* *#27 – 28 Каждый раз с помощью параметра «on_press» (что означает нажатие клавиши) будем вызывать функцию «WriteOnScreen», а с помощью параметра «on_release» (что означает отпускание клавиши), будем выполнять функцию «Reset_key».*

Как вывести в файл все кнопки на клавиатуре, которые были нажаты без интерпретации

```
1  from pynput.keyboard import Listener
2
3  def Reset_key(key):
4      None
5
6  def SaveToFile(key):
7      try:
8          if str(key) == "Key.esc" :
9              return exit()
10         if str(key) == "Key.space" :
11             key=" "
12         sym = str(key)
13         if len(sym) <= 3:
14             sym = sym.replace(" ' ","")
15         else:
16             sym = sym.rsplit(".")[1]
17             sym = " "+sym+" "
18
19         with open("log_punput.txt", "a") as f:
20             f.write(sym)
21
22     except Exception as er:
23         print("Имеем следующую ошибку: {0}".format(er.args[0]))
24
25 print("Нажмите на «esc», чтобы остановить программу!")
26 print("Нажмите любую клавишу, чтобы определить ее код. \n")
27
28 with Listener (on_press=SaveToFile, on_release=Reset_key) as lis:
29     lis.join()
```

Коды для сохранения символа в файле.

Комментарий к предыдущим кодам
#1 Импортируем клавиатура(keyboard) модуля «pynput» как объект типа слушателя (Listener) *#3 Функция отпускания нажатой клавиши* *#6-23 Функция для нажатия и определения его кода* *#8 Если нажмете клавишу «Escape», то программа должна остановиться (#9)* *#13 Проверяет, не является ли нажатая кнопка управляющей клавишей* *#14 Позволяет удалить апострофы или кавычки, которые сопровождают символ* *#15 Выводит код символа, который был нажат.* *#16 Разбивает строку «sym» на две части с помощью разделителя « . » и удерживает вторую часть цепочки.* *#17 Добавляет пробелы перед символом и после него, чтобы избежать визуальной путаницы с другими символами* *#19-20 Каждый раз нажатие клавиши сохраняется в файле* *#25-26 Печать клавиши на экране(консоль)* *#28 Каждый раз с помощью параметра «on_press» (что означает нажатие клавиши) будем вызывать функцию «SaveToFile», а с помощью параметра «on_release» (что означает отпускание клавиши), будем выполнять функцию «Reset_key».* *#29 Итеративно соединяет все нажатые кнопки в конце строки.*

С помощью этих кодов мы понимаем, как можно контролировать все манипуляции, выполняемые на клавиатуре. Эти манипуляции могут отображаться на консоли программного обеспечения Вашего компьютера, точно так же они могут дискретно храниться и выполняться из любого файла.

Мы можем импортировать библиотеку другим способом:

« ***from pynput import keyboard*** »

Таким образом, программа должна претерпеть небольшие изменения в кодах строки 28:

« with ***keyboard.***Listener (on_press=SaveToFile, on_release=Reset_key) as lis: ». Следует отметить, что две инструкции здесь идентичны, когда мы используем метод 1 (см. таблицу 3.2):

Таблица 3.2 – Разница в импорте модулей (строка и объект)

Метод 1	Метод 2
from pynput import keyboard	*from pynput.keyboard import Listener*
if key == keyboard.Key.esc:	-
if str(key) == 'Key.esc':	*if str(key) == 'Key.esc':*

Упражнение 1

Измените код таким образом, чтобы введенные данные совпадали с данными, хранящимися в файле. То есть перед их сохранением необходимо будет интерпретировать определенные клавиши и комбинации клавиш. Например, одновременно нажимая «Shift» и «**s**», Мы должны получить в файле заглавную букву «**S**»; нажав клавишу «Enter», мы должны перейти к строке; и так далее для других клавиш и комбинаций на клавиатуре. Для информации используйте таблицу 3.3.

Таблица 3.3 – Кнопки управления клавиатурой

Название в Punpyt	Официальное название и позиция на клавиатуре
Key.ctrl_l	Ctrl слева
Key.ctrl_r	Ctrl вправо
Key.alt_l	Alt слева
Key.alt_gr	Alt вправо
Key.shift	Shift слева
Key.shift_r	Shift вправо

Окончание табл. 3.3

Key.enter	Enter
Key.backspace	Backspace
Key.delete	Delete
Key.space	Space
Key.caps_lock	Caps lock
Key.home	Home
Key.end	End
Key.page_up	Page up
Key.page_down	Page down
Key.num_lock	Num lock
Key.pause	Pause
Key.print_screen	Print screen
Key.insert	Insert
Key.tab	Tab
Key.f1	F1
Key.f2	F2
Key.f3	F3
Key.f4	F4
Key.f5	F5
Key.f6	F6
Key.f7	F7
Key.f8	F8
Key.f9	F9
Key.f10	F10
Key.f11	F11
Key.f12	F12
Key.up	Up
Key.down	Down
Key.left	Left
Key.right	Right
Key.esc	Esc – Échap
Key.cmd	⊞ Win
Key.menu	Меню

RU III.3. Манипулирование компьютерной мышью

Мышь («mouse» на английском языке) – это указывающее устройство, которое позволяет перемещать курсор по экрану компьютера с координатами x и y (см. рисунок 3.3). оно также позволяет манипулировать и выбирать объекты с помощью «щелчков» или действий по двум кнопкам (одна слева, а другая справа). Также можно переместить этот курсор из кодов в Python. Для этого нам нужно использовать библиотеку по своему выбору: модуль «mouse», «pynput» и другие. В нашей книге, попытаемся понять, как манипулировать мышью, используя библиотеку «mouse», которая находится в модуле «pynput».

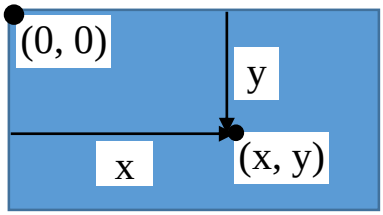


Рис. 3.3. положение координат указателя мыши на экране

Как изменить положение мыши из одной точки в другую

```
1  from pynput.mouse import Controller
2
3  def mouvMouse(x, y):
4      try:
5          mouse = Controller()
6          mouse.position = (x, y)
7      except Exception as er:
8          print("Имеем следующую ошибку: {0}".format(er.args[0]))
9
10 def inputcoord():
11     global x,y
12     x = input("Введите значение x: ")
13     y = input("Введите значение y: ")
14
15 inputcoord()
16 mouvMouse(x,y)
```

Комментарий к предыдущим кодам

#1 Импортируем мышь (mouse) из модуля «pynput» как объект управления (Controller)
#3 Функция для изменения положения указателя
#5 Делаем мышь (mouse) объектом управления
#6 Мы переместим положение наведения мыши на координаты x и y, которые появятся
#10 Функция для присвоения значений переменным x и y (координаты)
#11 Сделайте x и y глобальными переменными
#12 Запрос на введение значения x
#13 Запрос на введение значения y

...

Комментарий к предыдущим кодам

#15 Вызывает функцию «inputcoord()», чтобы ввести координаты x и y
#16 Вызывает функцию «mouvMouse()» для перемещения указателя мыши

Можно обновить предыдущий код без использования глобальных переменных. Они будут так выглядеть:

```
1  from pynput.mouse import Controller
2
3  def mouvMouse(x, y):
4      try:
5          mouse = Controller()
6          mouse.position = (x, y)
7      except Exception as er:
8          print("Имеем следующую ошибку: {0}".format(er.args[0]))
9
10 def inputXcoord():
11     x = input("Введите значение x: ")
12     return x
13
14 def inputYcoord():
15     y = input("Введите значение y: ")
16     return y
17
18 mouvMouse(inputXcoord(),inputYcoord())
```

Комментарий к предыдущим кодам

#10 Функция для присвоения значения переменной x (координата)
#14 Функция для присвоения значения переменной y (координата)
#18 Вызывает функцию «mouvMouse()» для перемещения указателя мыши после получения координаты: inputXcoord() и inputYcoord()

Как определить координаты указателя мыши во время его движения

```
1  from pynput.mouse import Listener
2
3  def ShowTheCoord(x, y):
4      print("Текущее положение мыши: {0}".format((x,y)))
5
6
7  with Listener (on_move=ShowTheCoord) as lis:
8      lis.join()
```

Комментарий к предыдущим кодам

#1 Импортируем мышь (mouse) из модуля «pynput» как объект типа слушателя (Listener).
#3-4 Функция, которая позволяет печатать на консоли программного обеспечения координаты указателя мыши во время его движения.

#7 Каждый раз с помощью параметра «on_move» (что означает перемещение указателя мыши) мы будем вызывать функцию «ShowTheCoord».
#8 Итеративно соединяет все координаты указателя мыши каждый раз в конец строки.

После запуска этого файла у нас будут отображаться координаты указателя мыши только при его перемещении. Пример этой визуализации приведен на рис. 3.4.

```
Текущее положение мыши: (562, 349)
Текущее положение мыши: (570, 349)
Текущее положение мыши: (577, 349)
Текущее положение мыши: (583, 347)
Текущее положение мыши: (586, 344)
Текущее положение мыши: (590, 340)
Текущее положение мыши: (594, 338)
Текущее положение мыши: (595, 336)
Текущее положение мыши: (596, 333)
Текущее положение мыши: (596, 331)
Текущее положение мыши: (596, 328)
Текущее положение мыши: (594, 325)
Текущее положение мыши: (590, 323)
Текущее положение мыши: (584, 319)
Текущее положение мыши: (576, 316)
Текущее положение мыши: (567, 312)
Текущее положение мыши: (552, 309)
```

Рис. 3.4. Результат выполнения предыдущего кода программы

Как иначе получить положение указателя мыши с помощью встроенного модуля «ctype»

Давайте определим, в каких координатах находится указатель мыши с заданным интервалом времени. Будем использовать модуль «time» для фиксации времени, а модуль «ctypes» (*который позволяет общаться или взаимодействовать с кодами языка «C», вызывать функции в файлах с расширением «dll» или вызывать некоторые сторонние функции*), чтобы найти эти координаты.

```
1  import time
2  import ctypes
3
4  def SourisGetPointeur():
5      tochka = le_format_xy()
6      Interface.GetCursorPos(ctypes.byref(tochka))
7      if int(tochka.L)==0 and int(tochka.H)==0:
8          global la_fin
9          la_fin = True
10     return {"L": tochka.L, "H": tochka.H}
11
12 print("Чтобы остановить программу, наведите указатель мыши
13                     на координаты \nx=0 и y=0 (т. е. в верхнем
14                     левом углу экрана)")
15 while True:
16     la_fin = False
17     Interface = ctypes.windll.user32
18     class le_format_xy(ctypes.Structure):
19         _fields_ = [("L", ctypes.c_uint), ("H", ctypes.c_uint)]
20
21     coord = SourisGetPointeur()
22     print(coord)
23
24     if la_fin:
25         print("КОНЕЦ ПРОГРАММЫ !")
26         break
27
28     time.sleep(2)
```

RU III.4. Как определить кодировку текстового файла

Важно знать, какая кодировка использовалась для создания текста или веб-страницы. Эти знания могут позволить нам создавать программное обеспечение, которое в зависимости от используемого языка может автоматически выбирать тип кодирования для правильного отображения информации на экране компьютера.

Ниже приведены примеры кодов, которые используются для определения кодирования одного или нескольких текстовых файлов в заданном каталоге.

Вариант 1. Случай, когда нам нужно проанализировать один файл

```
1  from chardet.universaldetector import UniversalDetector
2
3  the_detector = UniversalDetector()
4  puth = 'c:\\work\\my_file.txt'
5  the_detector.reset()
6
7  with open(puth, 'rb') as ligne:
8      for elt in ligne:
9          the_detector.feed(elt)
10         if the_detector.done:
11             break
12     the_detector.close()
13     the_answer = the_detector.result
14
15     le_code = the_answer['encoding']
16     le_lang = the_answer['language']
17     print('Кодировка = ',le_code, 'и Язык = ',le_lang)
```

Комментарий к предыдущим кодам

#1 Импорт класса «UniversalDetector» модуля «chardet», который позволяет обнаруживать кодировки символов
#3 Создание детектора
#4 указывает имя и полный путь к файлу, соответствующий требуемому шаблону
#5 Возвращает детектор в исходное состояние (удаляет старые данные)
#7 Получаем объект, который является файлом, в режиме чтения (данные из файла сохраняются в «ligne». Они помещаются в список как массив данных)
#8 Открываем цикл и выполняем процесс итерации по каждой строке
#9 – 10 Анализирует строку, полученную методом «feed»
#12 Закрытие детектора после использования
#13 Фиксирует полученный результат в переменной с именем «the_answer». Этот результат хранится в словаре из 3 элементов
#15 Удаляет имя из кодировки в словаре «the_answer», используя ключевое слово «encoding»
#16 Удаляет название языка из словаря «the_answer», используя ключевое слово «language»
#17 Печатает кодировку и полученный язык.

Вариант 2. ***Случаи, когда приходится проанализировать большое количество файлов с расширением «txt»***

```
1  import glob
2  from chardet.universaldetector import UniversalDetector
3
4  the_detector = UniversalDetector()
5  all_path = glob.glob('c:\\work\\*.txt')
6  for pathfile in sorted(all_path):
7      print(В файле','«',pathfile,'», имеем:'.ljust(6), end=' ')
8      the_detector.reset()
9      for ligne in open(pathfile, 'rb'):
10         the_detector.feed(ligne)
11         if the_detector.done:
12             break
13     the_detector.close()
14     the_answer = the_detector.result
15     for cle in the_answer:
16         if cle == 'encoding':
17             print('Кодировка =',the_answer[cle], end=' et ')
18         if cle == 'language':
19             if the_answer[cle]==None:
20                 print("Язык = Не удалось обнаружить ",end='.\n')
21             elif len(the_answer[cle])<=1:
22                 print("Язык = Не удалось обнаружить ",end='.\n')
23             else:
24                 print("Язык = ",the_answer[cle], end='.\n')
```

Комментарий к предыдущим кодам

#1 Импорт модуля «glob»: позволяет рекурсивно выбирать файлы в каталоге в соответствии с заданным шаблоном стиля Unix
#2 импорт класса «UniversalDetector» модуля «chardet», который позволяет обнаруживать кодировки символов
#4 Создание детектора
#5 Чтение всех файлов, находящихся в каталоге «work» на диске "C" (в операционной системе Windows), и сохранение их в списке с именем «all_file» в соответствии с указанным шаблоном
#6 Открытие цикла, чтобы просмотреть весь список сохраненных файлов после сортировки.
#7 Распечатать файл, удаленный из каталога, добавив пробелы
#8 Возвращает детектор в исходное состояние (удаляет старые данные)
#9 Открытие цикла для чтения из выбранного файла всех данных в режиме “rb”, построчно.
#10 и # 11 Анализируют линию, полученную детектором. Метод «feed()» используется для анализа части текста, чтобы обнаружить искомую информацию. Когда детектор получает эту информацию, затем команда «Detector.done» принимает значение «True», что позволяет нам завершить анализ
#13 Закрытие детектора после использования
#14 Фиксирует полученный результат в переменной с именем «the_answer». Этот результат хранится в словаре из 3 элементов.
#15 Открытие цикла для удаления типа кодировки и языка, используемого в активном файле
 # С 16 по 24 строки, печатаем кодировку и язык, используя ключевые слова “словаря” (‘encoding’ et ‘language’).

Вариант 3

```
1  import glob
2  from chardet.universaldetector import UniversalDetector
3
4  the_detector = UniversalDetector()
5  puth = 'c:\\work\\*.txt'
6  all_path = glob.glob(puth)
7  for pathfile in all_path:
8      print('В файле','«',pathfile,'», имеем:'.ljust(6), end=' ')
9      the_detector.reset()
10     with open(pathfile, 'rb') as ligne:
11         for ll in ligne:
12             the_detector.feed(ll)
13             if the_detector.done:
14                 break
15     the_detector.close()
16     the_answer = the_detector.result
17
18     le_code = the_answer['encoding']
19     le_lang = the_answer['language']
20     print('Кодировка = ',le_code, 'и Язык = ',le_lang)
```

Комментарий к предыдущим кодам
#1 Импорт модуля «glob»: позволяет рекурсивно выбирать файлы в каталоге в соответствии с заданным шаблоном стиля Unix *#2 Импорт класса «UniversalDetector» модуля «chardet», который позволяет обнаруживать кодировки символов* *#4 Создание детектора* ...

Commentaire des codes précédents

#5 Указывает имя и полный путь к файлу, соответствующий требуемому шаблону
#6 Чтение всех файлов, находящихся в каталоге «work» на диске "C" (в операционной системе Windows), и сохранение их в списке с именем «all_file» в соответствии с указанным шаблоном
#7 Открытие цикла, чтобы просмотреть весь список сохраненных файлов
#8 Распечатать файл, удаленный из каталога, добавив пробелы
#9 Возвращает детектор в исходное состояние (удаляет старые данные)
#10 Получаем объект, который является файлом, в режиме чтения (данные из файла сохраняются в «ligne». Они помещаются в список как массив данных)
#11 Открываем цикл и выполняем процесс итерации по каждой строке
#12-14 Анализирует линию, полученную детектором
#15 Закрытие детектора после использования
#14 Фиксирует полученный результат в переменной с именем «the_answer». Этот результат хранится в словаре из 3 элементов.
#15 Открытие цикла для удаления типа кодировки и языка, используемого в активном файле
#16 Фиксирует полученный результат в переменной с именем «the_answer». Этот результат хранится в словаре из 3 элементов.
#18 удаляет имя из кодировки в словаре «the_answer», используя ключевое слово «encoding»
#19 Удаляет название языка из словаря «the_answer», используя ключевое слово «language»
#20 Печатает кодировку и полученный язык.

Задание 1.

Мы только что представили 3 варианта определения кодировки и языка созданного файла. Вам необходимо использовать все три версии для создания четвертого варианта. Определите четвертый вариант решения предыдущей задачи, используя информацию, полученную из версий 1, 2 и 3. Усовершенствуйте коды, используя инструкции: try, except, else и/или finally (см. Том 1).

RU III.5. Как определить кодировку веб-страницы

Важно знать, что если вы откроете веб-страницу и обнаружите, что обычные символы, которые вы ожидаете, выглядят как необычные символы, как странные символы (см. рис. 3.5: А, Б, В, Г), которые не имеют ничего общего с простым текстом (см. рис. 3.5: Д) то, что вы ожидаете, означает, что у вас проблема с кодировкой символов в вашей веб-странице. Эта проблема в настоящее время встречается редко, поскольку браузеры, в которые используются для просмотра, стали практически интеллектуальными, чтобы избежать этого неправильного отображения символов на веб-страницах.

RU III.5.1. Представление кодировки на веб-странице

Любая информация в виде текста записывается в документе строго определенными символами, имеющими определенную кодировку. Веб-страницы не являются исключением. Наиболее распространенными и используемыми являются:

- KOI8-R (8-битная кодовая страница – для кодирования букв кириллицы (букв русского алфавита))
- KOI8-U (8-битная кодовая страница – для кодирования букв кириллицы (букв украинского алфавита))
- Windows 1251 (cp1251 – для операционной системы Windows - русская версия)

- ISO-8859-1 (иногда его называют latin1 – для французского языка)
- ISO-8859-15 (иногда его называют latin9 – вариация предыдущего)
- UTF-8 (8 представляет минимальную длину в битах, которую может иметь символ. Его размер составляет от 1 до 4 байтов); UTF-16 (минимальная длина составляет 16 бит); UTF-32 (здесь, нет минимального размера байта, он фиксирован, и каждый символ занимает 4 байта в памяти). Таким образом, Юникод теоретически позволяет кодировать все языки: французский, английский, китайский, арабский и многие другие, не говоря уже о русском. Это стандарт, который в настоящее время используется языками программирования.

Кодировка в веб-документе объявляется следующим образом: во все HTTP-заголовки, отправляемые сервером вместе с веб-страницей, необходимо вводить их в стандартной формулировке.

В веб-документе html вы сможете вставлять в заголовки следующие коды:

```
<meta http-equiv="content-type" content="text/html; charset=utf-8">
```

В любом случае мы должны помнить, что информация, предоставляемая тегом «мета», является вторичной и не имеет привилегии быть первой. Сначала будет прочитана информация, предоставленная сервером, то есть эта информация будет приоритетна, если она существует.

А – charset=koi8-r

Б – charset=koi8-u

В – charset=cp1251

Г – charset=ISO-8859-1

Д – charset=utf-8

Рис. 3.5. примеры отображения русского текста в браузере с использованием разных кодировок (А, Б, В, Г и Д)

RU III.5.2. Определение кодировки веб-страницы с помощью кода

На языке Python существует несколько способов определения кодировки веб-страницы. Для достижения поставленной целей необходимо использовать клиентскую библиотеку HTTP/1.1. С его помощью HTTP-соединение (HTTPConnection) проходит через ряд «состояний», которые определяют период времени, в течение которого

клиент может на законных основаниях сделать другой запрос или получить ответ, который был дан на конкретный запрос. Мы также используем модуль «request» (для запроса), который является расширяемой библиотекой для открытия URL-адресов с использованием различных протоколов.

В этом модуле функция «urlopen» принимает строку, содержащую URL-адрес или объект запроса. Он работает следующим образом: открывает URL-адрес и возвращает результаты как объект типа файла; у возвращаемого объекта есть несколько дополнительных методов, которые можем использовать: OpenerDirector, urlopen, Request.

Таким образом, можно сказать, что модуль «request» позволяет отправлять полный запрос на сервер.

Прежде чем получить кодировку какой-либо страницы, сначала необходимо проверить, существует ли эта страница или нет. Для этого мы будем использовать функцию «HTTPResponse», которая позволяет получить ответ на запрос, запрошенный сервером. Таким образом, с помощью методов «status», «reason» и «code», сможем получить состояние веб-страницы, то есть определить, существует ли оно или нет, используя данные из таблицы 3.4.

Таблица 3.4 – Некоторые ориентировочные данные при поиске веб-страницы на сайте

Статус (status)	404	301-303, 307	200	405	400	406
Вердикт (reason)	Не найдено	Временно пере-мещен	Ok	Не допус-кается	Неудач-ный запрос	Неприем-лемо
Код (code)	404	301-302	200	405	400	406
Смысловое значение	Страница не существует	Ошибки перена-правления	Найденная страница	Несан-кциони-рованная страница	Неверный запрос	Нет приемлемого ответа

Вариант 1

```
1  import http.client as siteweb
2  import urllib.request
3  from chardet.universaldetector import UniversalDetector
4
5  web_link = "http://dl.papacha.ru/index.php/registratsiya1"
6
7  union = siteweb.HTTPConnection('dl.papacha.ru')
8  union.request("HEAD", web_link)
9  answer = union.getresponse()
10 union.close()
11
12 print("Статус веб-страницы = ",answer.status)
13 print("Доступность = ",answer.reason)
14
15 if answer.code != 404:
16     web_link1 = urllib.request.urlopen(web_link)
17     the_detector = UniversalDetector()
18     for line in web_link1.readlines():
19         the_detector.feed(line)
20         if the_detector.done: break
21     the_detector.close()
22     web_link1.close()
23     print(' Кодирование = ',the_detector.result['encoding'], 'и
24                             Язык = ',the_detector.result['language'])
25 else:
26     print("Веб-страница не существует из-за: ", answer.reason)
```

Вариант 2.

```
1  import urllib.request
2  import chardet
3  web_link = "http://dl.papacha.ru/index.php/registratsiya"
4  dataweb = urllib.request.urlopen(web_link).read()
5
6  chardet.detect(dataweb)
7  print(' Кодирование = ',chardet.detect(dataweb)['encoding'], 'и
8                  Язык = ',chardet.detect(dataweb)['language'])
```

Комментарий к предыдущим кодам
#1 Импорт модуля urllib. request *для открытия URL-адресов* *#2 Импорт модуля* chardet *для определения кодировки символов* *#3 Определение адреса, по которому можем попытаться обнаружить кодировку* *#4 Получает результат запроса на указанной странице в виде объекта* *#6 Прочитать выбранные данные, чтобы определить кодировку символов* *#7 Удаляет название кодировки и языка из "словаря", используя ключевые слова «encoding» и «language», чтобы отобразить его на экране.*

В заключение, в случае, если мы хотим обнаружить кодировку нескольких файлов по отдельности, необходимо повторно использовать один объект *UniversalDetector* в одном и том же коде. С помощью команды *detector.reset()* нам нужно будет сбросить детектор в начале каждого файла, прежде чем выполнять повторную проверку кодировки с помощью команды *detector.feed()* столько раз, сколько у нас будет новых файлов для обнаружения. Каждый раз вам нужно будет закрыть его, вызвав команду *detector.close()* и, наконец, удалить результат с помощью команды *detector.result*; этот результат будет сохранен в “словаре”.

Как вы уже заметили, мы использовали модуль обнаружения символов в Mozilla. Он работает следующим образом: определенный компонент получает байты в качестве входящих данных и опирается на другие данные, чтобы иметь возможность “угадывать” набор символов, используемых данных. Данный модуль не выполняет поиск информации помечены (например: *<meta http-equiv="content-type" content="text/html; charset=utf-8">*) в веб-страницы. Модуль «chardet» скорее угадывает, что означает, что иногда ответ неточен, то есть может быть неправильным. Таким образом, вы можете использовать его или вносить другие изменения, чтобы сделать результат на 100% надежным.

Продолжение на странице 253


EN Chapter III. ROW PROCESSING IN PYTHON

EN Introduction

A string of characters can be defined as an ordered sequence of characters. It can be considered as simple symbols in which each of them has a number (for processing) or a specific code (for storing its value). As we said earlier, we distinguish two types of strings: the first type of strings, limited to 256 characters, where each character has a unique code depending on the encoding system; the second type of encoding has only 2 types of characters, which are 0 and 1 – this is really what we mean by byte strings. It should be noted that each character is represented by a number of bytes. This number also depends on the encoding used.

EN III.1. The process of encoding characters

Given that there are several languages in the world, the amount of data is growing due to the use of computers, and in order for the data to be codified in the form of text, unified coding tables are used all over the world (one part is in Table 3.1). Each language corresponds to a certain type or types of coding. For example, in order to visually and accurately display the Russian text, it would be correct to use Cyrillic encodings. Let's consider practical examples of how in a programming language, encodings are determined using two small functions, namely: *chr()* and *ord()*. In our modern browsers, the encoding is selected automatically depending on the language used or written. This means that the user's interference has been eliminated in advance.

Thus, we can say that character encoding is a way to ensure a correspondence between what the user sees on the screen and what the machine (computer) actually stores in memory. The «chardet» library is a module that allows you to automatically determine the encoding of text. This

is the main part of the code for automatic encoding detection in a browser known as «Mozilla». To use it, it needs to be installed.

Table 3.1. – Table of ASCII codes and their correspondences

ASCII encoding	
Code value	Symbol
32	space
33	!
…	…
48	0
49	1
50	2
…	…
97	a
98	b
…	
120	x
121	y
122	z
…	…
127	delete

This encoding is used on most local computers running on the Windows platform

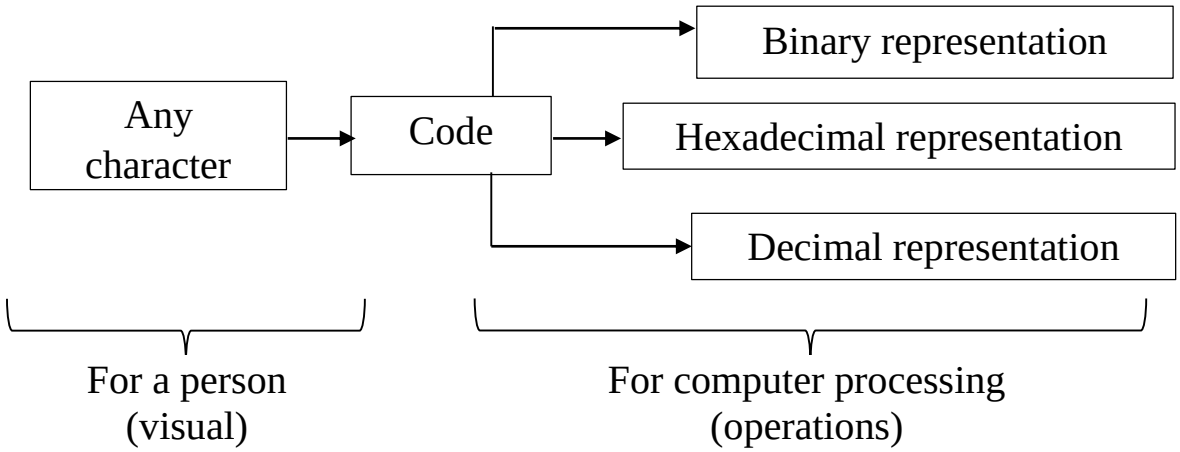


Figure 3.1. Representation of the symbol

The code is unique in any standard used. Note that the extended ASCII code table will vary depending on the character encoding used.

EN III.2. Computer keyboard control

To control the keyboard, we can use various modules that will be imported into the program code.

Keyboard module

With this library, we can fully control the keyboard. We can create keyboard shortcuts, make notes, lock the keyboard, wait for data entry and many other important functions. Let's try to understand how using this module we can detect keystrokes on the keyboard. Here you can use three ways to detect these keystrokes on the keyboard: *«read_key()», «is_pressed()»* and *«on_press_key()»*.

Option 1. Using the *«read_key()»* function

The *«read_key()»* function allows you to read which button on the keyboard the user pressed, that is, we compare the *«read_key()»* function (which returns a character) with the keyboard symbol.

Usage example

Determine the ASCII code of the pressed key on the keyboard.

Additional information:

chr(code) – allows you to get a character using an ASCII code;

ord (character) – allows you to get an ASCII code through its character.

```
1  import keyboard, time
2
3  while True:
4      symbol = keyboard.read_key()
5      if len(str(symbol))>1 :
6          print("The «{0}» key has no code".format(symbol))
7      else:
8          print("The code «{0}» corresponds to «{1}»".format(symbol,
9                                                  ord(symbol)))
10     if symbol == "esc":
11         break
12     time.sleep(0.25)
```

The result of this program allows you to confirm and verify the data given in Table 3.1.

Comments on previous program codes
#1 import of the «keyboard» module to determine the ASCII code of the keystroke on the keyboard, also import of the "time" module to add a delay to the program. *#3 occurs if the condition (instructions) is true.* *#4 wait until the keyboard key is pressed.* *#5 determines whether it is a control character by checking whether its length is strictly greater than 1.* *#6 prints the control character.* *#7 If it is not a control character.* *#8 prints the pressed keyboard character and its ASCII code.* *##10 if you press the Escape symbol on the keyboard.* *#11 Exit the loop.* *#12 adds standby time (1/4 second) for touch keyboards.*

Option 2. Using the *is_pressed()* function

The *is_pressed()* function contains an input parameter. This parameter is considered any character, and if it corresponds to the button that the user clicked, then in this case it will return the boolean value «True», otherwise it will return the value «False».

Usage example.

Determine the ASCII code for the key pressed on the keyboard.

```
1  import keyboard
2  from time import sleep
3
4  print("Press the «Escape» key to complete")
5  while True:
6      symbol = keyboard.read_key()
7      if keyboard.is_pressed("esc"):
8          print("End !")
9          break
10     elif len(symbol)>1:
11         print("You have pressed the control button {0}".format(symbol))
12     else :
13         print("You have pressed on the symbol{0}".format(symbol))
14         print("The code «{0}» corresponds to «{1}»".format(symbol,
15                                                    ord(symbol)))
16     sleep(0.5)
17
```

Comments on previous program codes

```
#1 Import of the «keyboard» module to determine the ASCII code for
   pressing a key on the keyboard
#2 Import the «sleep()» function, which is located in the «time» module,
   to add a delay to the program
#5 Opening the loop if the condition (instructions) is true
#6 Using the «read_key()» function to enter a character from the
   keyboard (waits for a key to be pressed on the keyboard)
#7 Checks if the «Escape» key is pressed
#9 Exit the loop
#10 Determines whether it is a control character by checking whether its
   length is strictly greater than 1
#11 Prints the character
#13 If it's not a symbol
#14 Outputs the pressed character and its ASCII code
#16 Adds a timeout (1/2 second) for touch keyboards (optional).
```

Option 3. Using the function «on_press_key()»

The function «on_press_key()» uses two parameters as input: the first is the character we want to get or identify, and the second parameter is a function of the type «lambda _». If the user clicks the button corresponding to the value of the first parameter, he will only perform the function located after the «lambda _» of the second parameter.

Usage example

It is necessary to show the usefulness of the «on_press_key()» function. How can we use it in a different way.

```
1  import keyboard
2
3  def print_one_line():
4      print(' Using the «esc» key, you can complete ')
5
6  keyboard.on_press_key("esc", lambda _: imprime_une_phrase())
7  keyboard.on_press_key("s", lambda _: keyboard.write("The «s» key
8                                          was pressed", delay = 0.1))
9  keyboard.on_press_key("z", lambda _: print("The «z» key was
10                                 pressed"))
```

Since the «keyboard» function returns the event of this function, it is necessary to use the «_» symbol.

Comments on previous program codes
#1 Import of the «keyboard» module to determine the ASCII code for pressing a key on the keyboard *#3 and 4 Function for printing text on the screen* *#6 If the «Escape» key is pressed, we will see the execution of the «print_on_line()» function* *#7 If the «s» key is pressed, then we observe the execution of the «write» function of the «keyboard» module. The text will be displayed gradually by characters with a delay, which will be determined by the «delay()» method of the «write()» function.* *#9 if the «z» key is pressed, then we observe the execution of the «print» function. The difference between «print» and «write» is that using the «write» function in the «keyboard» library, the text is printed where the cursor is located. It can even be active outside the console, where the program was launched.*

Other functions of this module:

keyboard.wait('esc')

Press Esc to continue with the following instructions. Instead of «esc», we can put any key on the keyboard there.

keyboard.press('caps_lock')

Allows the program to press the «caps_lock» key on the keyboard. Instead of «caps_lock» we can put any key on the keyboard there.

keyboard.release('caps_lock')

Allows the program to release the «caps_lock» key on the keyboard that was pressed.

keyboard.press_and_release('shift+s, space, =,space, 9')

Allows you to write the command «S = 9». We can put other keywords and symbols in their places.

keyboard.add_hotkey('enter', lambda: keyboard.write("You pressed the «Enter» key!!!"))

If the «Enter» key is pressed, then we observe the execution of the «write» function of the « keyboard » module. Instead of «keyboard.write», we can put another function to execute in it. Similarly, instead of «enter», we can put another key or key combination: for example, «shift+r» (for which you will need to press both the «shift» key and the «r» key at the same time)

keyboard.send('keyboard_key')

Allows you to press and release the key located in the brackets of the function.

keyboard.add_abbreviation(source_text, replacement_text, match_suffix = False, expiration_time = 2)

Allows you to speed up or automate the writing of a small text (***replacement_text***) using a keyword (***source_text***) or a symbol. You enter a keyword (character), and after pressing the space bar, it is directly replaced by another one, which is « ***replacement_text*** ».

Example: *keyboard.add_abbreviation('@ ', 'fiston@gmx.fr')*

list_ev = keyboard.record(until='keyboard_key')

Records all key events on the keyboard until the user presses the specified hotkey (*keyboard_key*). After that, these events are saved in a list (***list_ev***), which can be viewed or played using the «play» function.

keyboard.play(list_ev, speed_factor=1.0)

Outputs all events, keeping relative time intervals during recording. If the ***speed_factor*** parameter is less than or equal to **0**, the actions will be reproduced as fast as the operating system allows.

Example of using **« record » and « play »**

```
1  import keyboard
2
3  try:
4     keyboard.write("Everything you are going to do with the
5                     keyboard will be recorded. The recording will be
6                     completed after you can press the «escape» key.\n")
7
8     record = keyboard.record(until="esc", suppress=False,
9                     trigger_on_release=False)
10    print("Saved list = ",record)
11    print("An exact copy of what was recorded ")
12    keyboard.play(record)
13 except Exception as er:
14    print("We have the following error: {0}".format(er.args[0]))
```

Comments on previous program codes
#1 Import the «keyboard» module to record all keys on the keyboard. *#3 and 13 are used to check the code, which may contain errors* *#4 Prints text using the write function. The text is printed where the cursor is located. After the cursor moves to the line, using the command "\n"* *#8 All keyboard key events are recorded until the user presses the «esc» key. The «suppress» and «trigger_on_release» parameters can be deleted.* *#10 Prints the events that are in the list, and we get the result in Figure 2.* *#12 Identically shows all the events that were recorded.*

The data is entered during registration >>> 12345
Saved_list = [KeyboardEvent(1 down), KeyboardEvent(1 up), KeyboardEvent(2 down), KeyboardEvent(2 up), KeyboardEvent(3 down), KeyboardEvent(3 up), KeyboardEvent(4 down), KeyboardEvent(4 up), KeyboardEvent(5 down), KeyboardEvent(5 up), KeyboardEvent(esc down)]
The result of playing a record in the list using the «*play()*» function >>>12345

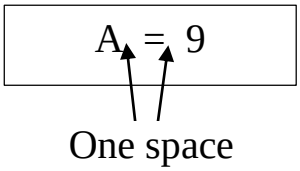
Fig. 3.2. Entered data and recording results performed by the «*record*» function of the «keyboard» module

Pynput module

In comparison with the previous library (keyboard), this module is more functional and allows not only to fully control the keyboard, but also to manipulate mouse movements by coordinates. Let's try to understand how using this module we can detect keyboard and mouse keystrokes.

How to press a key and release it?

The « ***press*** » method allows you to press any key, and the « ***release*** » method allows you to release this key. In the following example, let's look at how they can be used to write an equation:




```
1  from pynput.keyboard import Key, Controller
2
3  try:
4      keyboard = Controller()
5
6      keyboard.press(Key.caps_lock)
7      keyboard.press('a')
8      keyboard.release('a')
9      keyboard.press(Key.space)
10     keyboard.release(Key.space)
11     keyboard.press('=')
12     keyboard.release('=')
13     keyboard.press(Key.space)
14     keyboard.release(Key.space)
15     keyboard.press('9')
16     keyboard.release('9')
17     keyboard.release(Key.caps_lock)
18 except Exception as er:
19     print("We have the following error: {0}".format(er.args[0]))
```

```
1  from pynput.keyboard import Key, Controller
2
3  try:
4      keyboard = Controller()
5
6      with keyboard.pressed(Key.shift):
7          keyboard.press('a')
8          keyboard.release('a')
9      keyboard.press(Key.space)
10     keyboard.release(Key.space)
11     keyboard.press('=')
12     keyboard.release('=')
13     keyboard.press(Key.space)
14     keyboard.release(Key.space)
15     keyboard.press('9')
16     keyboard.release('9')
17 except Exception as er:
18     print("Errors encountered:",er)
19
```

Comments on previous program codes

#1 Import of the «Key» and «keyboard» classes from the «pynput.keyboard» module.
#4 Defines «keyboard» as the keyboard object controller coming from the «pynput» module.

#6 By pressing the «Shift» key, we also press the «a» key (#7), and then release it (#8).
#9 ...

How to display a message using this module?

```
1  from pynput.keyboard import Controller
2
3  the_ keyb = Controller()
4  the_ keyb.type("Any text!")
```

or

```
1  from pynput.keyboard import Controller
2
3  def ShowMessageKeyboard():
4      keyb = Controller()
5      keyb.type(" Any text!")
6
6  ShowMessageKeyboard()
```

Comments on previous program codes
#4 ou #5 We will just get the following message on the screen: *Any text!*

How to visually display all the pressed buttons with their codes on the console

To do this, we need to use the «Listeners» library so that we can listen to keystrokes.

```
from pynput.keyboard import Listener
def Reset_key(key):
    print("The %s key has been returned \n'%(key))

def WriteOnScreen (key):
    try:
        if str(key) == 'Key.esc' :
            return exit()
        print('You pressed the key',key)

        sym = str(key)
        if len(sym) == 3:
            sym = sym.replace("'","")
            val = str(ord(sym))
        else:
            sym = sym.rsplit('.')[1]
            sym = " "+sym+" "
            val = " None"

        print("The '%s' key has the code value '%s'" %(sym, val))
    except Exception as er:
        print("We have the following error: {0}".format(er.args[0]))

print("Press «esc» to stop the program!")
print("Press any key to determine its code.\n")

with Listener
            (on_press=WriteOnScreen, on_release=Reset_key) as lis:
    lis.join()
```

Comments on previous program codes
#1 Import the keyboard of the «pynput» module as an object of the Listener type *#2 The function of releasing the pressed key* *#5 – 23 Function for pressing and determining its code* *#7 If you press the «Escape» key, the program should stop (#9)* *#12 Checks whether the pressed button is not a control key* *#13 Allows you to remove apostrophes or quotation marks that accompany the character* *#14 Outputs the code of the character that was pressed.* *#16 Splits the string «sym» into two parts using the separator «.» and holds the second part of the chain.* *#17 Adds spaces before and after the character to avoid visual confusion with other characters* *#18 Assign the value «None» to the control buttons* *#20 Printing a key on the screen (console)* *#27 – 28 Each time using the «on_press» parameter (which means pressing a key), we will call the «WriteOnScreen» function, and using the «on_release» parameter (which means releasing the key), we will perform the «Reset_key» function.* *#29 Iteratively combines all the pressed buttons at the end of the line.*

How to output to a file all the buttons on the keyboard that were pressed without interpretation

```
1  from pynput.keyboard import Listener
2
3  def Reset_key(key):
4      None
5
6  def SaveToFile(key):
7      try:
8          if str(key) == "Key.esc" :
9              return exit()
10         if str(key) == "Key.space" :
11             key=" "
12         sym = str(key)
13         if len(sym) <= 3:
14             sym = sym.replace(" ' ","")
15         else:
16             sym = sym.rsplit(".")[1]
17             sym = " "+sym+" "
18
19         with open("log_punput.txt", "a") as f:
20             f.write(sym)
21
22     except Exception as er:
23         print("We have the following error: {0}".format(er.args[0]))
24
25 print("Press «esc» to stop the program!")
26 print("Press any key to determine its code. \n")
27
28 with Listener (on_press=SaveToFile, on_release=Reset_key) as lis:
29     lis.join()
```

Codes for saving a symbol in a file

Comments on previous program codes
#1 Import the keyboard of the «pynput» module as an object of the Listener type *#3 Release function of the pressed key* *#6-23 Function for clicking and determining its code* *#8 If you press the «Escape» key, the program should stop (#9)* *#13 Checks whether the pressed button is not a control key* *#14 Allows you to remove apostrophes or quotation marks that accompany the character* *#15 Outputs the code of the character that was pressed.* *#16 Splits the string «sym» into two parts using the separator «.» and holds the second part of the chain.* *#17 Adds spaces before and after the character to avoid visual confusion with other characters* *#19-20 Every time a key is pressed, it is saved in a file* *#25-26 Printing a key on the screen (console)* *#28 Every time using the «on_press» parameter (which means pressing a key), we will call the «SaveToFile» function, and using the «on_release» parameter (which means releasing the key), we will perform the «Reset_key» function.* *#29 Iteratively connects all the pressed buttons at the end of the line.*

With these codes, we understand how to control all manipulations performed on the keyboard. These manipulations can be displayed on your computer's software console, just as they can be discretely stored and executed from any file.

We can import the library in another way:

« ***from pynput import keyboard*** »

Thus, the program should undergo minor changes in the line 28 codes:

« with ***keyboard.***Listener (on_press=SaveToFile, on_release=Reset_key) as lis: ». It should be noted that the two instructions here are identical when we use method 1 (see Table 3.2):

Table 3.2 – Difference in module imports (row and object)

Method 1	Method 2
from pynput import keyboard	*from pynput.keyboard import Listener*
if key == keyboard.Key.esc:	-
if str(key) == 'Key.esc':	*if str(key) == 'Key.esc':*

Exercise 1

Change the code so that the entered data matches the data stored in the file. That is, before saving them, it will be necessary to interpret certain keys and key combinations. For example, by simultaneously pressing «Shift» and «s», we should get the uppercase letter «S» in the file; by pressing the «Enter» key, we should go to the line; and so on for other keys and keyboard combinations. For information, use Table 3.3.

Table 3.3 – Keyboard control buttons

Title in Punpyt	Official name and position on the keyboard
Key.ctrl_l	Ctrl слева
Key.ctrl_r	Ctrl вправо
Key.alt_l	Alt слева
Key.alt_gr	Alt вправо
Key.shift	Shift слева
Key.shift_r	Shift вправо
Key.enter	Enter
Key.backspace	Backspace
Key.delete	Delete

End of table 3.3

Key.space	Space
Key.caps_lock	Caps lock
Key.home	Home
Key.end	End
Key.page_up	Page up
Key.page_down	Page down
Key.num_lock	Num lock
Key.pause	Pause
Key.print_screen	Print screen
Key.insert	Insert
Key.tab	Tab
Key.f1	F1
Key.f2	F2
Key.f3	F3
Key.f4	F4
Key.f5	F5
Key.f6	F6
Key.f7	F7
Key.f8	F8
Key.f9	F9
Key.f10	F10
Key.f11	F11
Key.f12	F12
Key.up	Up
Key.down	Down
Key.left	Left
Key.right	Right
Key.esc	Esc – Échap
Key.cmd	⊞ Win
Key.menu	Меню

EN III.3. Manipulating a computer mouse

A mouse is a pointing device that allows you to move the cursor around the computer screen with ***x*** and ***y*** coordinates (see Figure 3.3). it also allows you to manipulate and select objects using «clicks» or actions on two buttons (one on the left and the other on the right). You can also move this cursor from the codes to Python. To do this, we need to use the library of our choice: the module «mouse», «pynput» and others. In our book, we will try to understand how to manipulate the mouse using the "mouse» library, which is located in the «pynput» module.

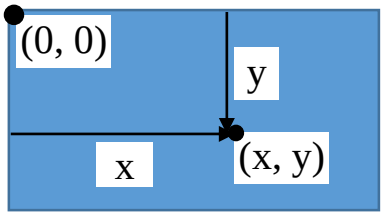


Figure 3.3. The position of the mouse pointer coordinates on the screen

How to change the mouse position from one point to another?

```
1  from pynput.mouse import Controller
2
3  def mouvMouse(x, y):
4      try:
5          mouse = Controller()
6          mouse.position = (x, y)
7      except Exception as er:
8          print("We have the following error: {0}".format(er.args[0]))
9
10 def inputcoord():
11     global x,y
12     x = input("Enter the value of x: ")
13     y = input("Enter the value of y:")
14
15 inputcoord()
16 mouvMouse(x,y)
```

Comments on previous program codes

#1 Import the mouse from the «pynput» *module as a control object (Controller)*
#3 Function to change the pointer position
#5 Making the mouse an object of control
#6 We will move the mouse hover position to the x and y coordinates that will appear
#10 Function for assigning values to variables x and y (coordinates)
#11 Make x and y global variables
#12 Request to enter the value of x
#13 Request to enter the value of y

.....

Comments on previous program codes

#15 Calls the «inputcoord()» function to enter the x and y coordinates
#16 Calls the «mouvMouse()» function to move the mouse pointer

You can update the previous code without using global variables. The result will look like this:

```
1  from pynput.mouse import Controller
2
3  def mouvMouse(x, y):
4      try:
5          mouse = Controller()
6          mouse.position = (x, y)
7      except Exception as er:
8          print("We have the following error: {0}".format(er.args[0]))
9
10 def inputXcoord():
11     x = input("Enter the value of x: ")
12     return x
13
14 def inputYcoord():
15     y = input("Enter the value of y:")
16     return y
17
18 mouvMouse(inputXcoord(),inputYcoord())
```

Comments on previous program codes

#10 Function to assign a value to variable x (coordinate)
#14 Function to assign a value to variable y(coordinate)
#18 Calls the "mouvMouse()" function to move the mouse pointer after getting the coordinates: inputXcoord() and inputYcoord()

How to determine the coordinates of the mouse pointer during its movement?

```
1  from pynput.mouse import Listener
2
3  def ShowTheCoord(x, y):
4      print("Current mouse position: {0}".format((x,y)))
5
6
7  with Listener (on_move=ShowTheCoord) as lis:
8      lis.join()
```

Comments on previous program codes

#1 Import the mouse from the «pynput» module as a Listener type object.
#3-4 A function that allows you to print the coordinates of the mouse pointer on the software console while it is moving.

#7 Every time using the «on_move» parameter (which means moving the mouse pointer), we will call the «ShowTheCoord» function.
#8 Iteratively connects all the mouse pointer coordinates each time to the end of the line.

After launching this file, we will display the coordinates of the mouse pointer only when it is moved. An example of this visualization is shown in Figure 3.4.

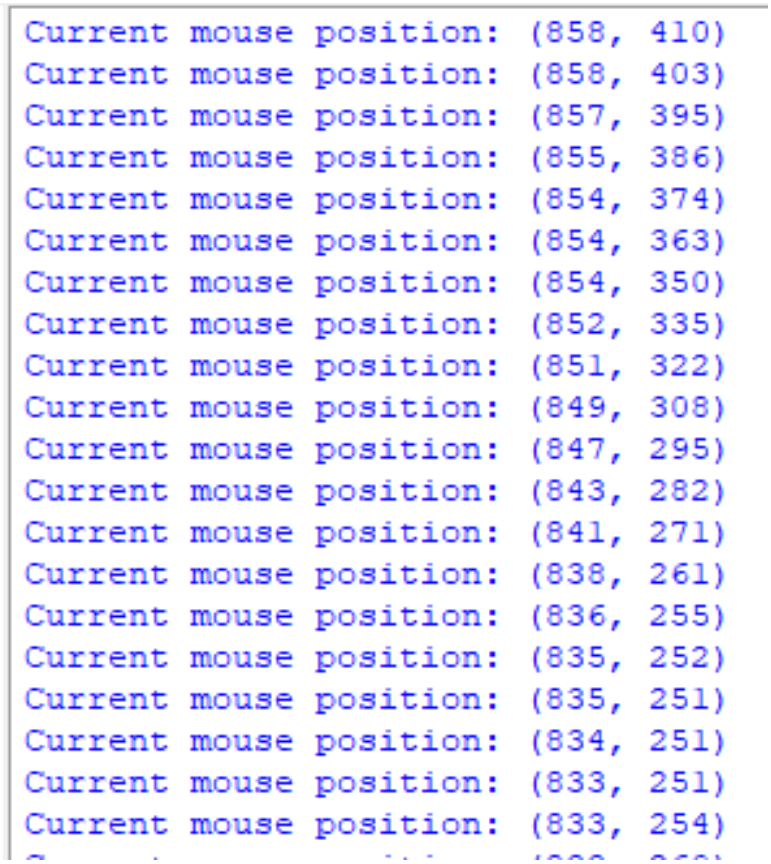


Figure 3.4. The result of executing the previous program code

How else to get the position of the mouse pointer using the built-in module «ctypes»?

Let's determine in which coordinates the mouse pointer is located at a given time interval. We will use the "time" module to fix the time, and the «ctypes» module (*which allows you to communicate or interact with the «C» language codes, call functions in files with the «dll» extension or call some third-party functions*) to find these coordinates.

```
1  import time
2  import ctypes
3
4  def SourisGetPointeur():
5      tochka = le_format_xy()
6      Interface.GetCursorPos(ctypes.byref(tochka))
7      if int(tochka.L)==0 and int(tochka.H)==0:
8          global la_fin
9          la_fin = True
10     return {"L": tochka.L, "H": tochka.H}
11
12 print("To stop the program, hover the mouse pointer over the
13                     coordinates\nx=0 and y=0 (that is, in the upper left
14                     corner of the screen)")
15 while True:
16     la_fin = False
17     Interface = ctypes.windll.user32
18     class le_format_xy(ctypes.Structure):
19         _fields_ = [("L", ctypes.c_uint), ("H", ctypes.c_uint)]
20
21     coord = SourisGetPointeur()
22     print(coord)
23
24     if la_fin:
25         print("END OF THE PROGRAM!")
26         break
27
28     time.sleep(2)
```

EN III.4. How to determine the encoding of a text file

It is important to know which encoding was used to create the text or web page. This knowledge can allow us to create software that, depending on the language used, can automatically select the type of encoding for the correct display of information on the computer screen.

The following are examples of codes that are used to determine the encoding of one or more text files in a given directory.

Option 1. The case when we need to analyze one file

```
1  from chardet.universaldetector import UniversalDetector
2
3  the_detector = UniversalDetector()
4  puth = 'c:\\work\\my_file.txt'
5  the_detector.reset()
6
7  with open(puth, 'rb') as ligne:
8      for elt in ligne:
9          the_detector.feed(elt)
10         if the_detector.done:
11             break
12     the_detector.close()
13     the_answer = the_detector.result
14
15     le_code = the_answer['encoding']
16     le_lang = the_answer['language']
17     print(' Encoding = ',le_code, 'and Language = ',le_lang)
```

Comments on previous program codes

#1 Import of the «UniversalDetector» class of the «chardet» module, which allows detecting character encodings
#3 Creating a detector
#4 specifies the name and full path to the file corresponding to the required template
#5 Returns the detector to its original state (deletes old data)
#7 We get the object, which is a file, in read mode (data from the file is saved in «ligne». They are placed in the list as an array of data)
#8 Open the loop and perform the iteration process for each row
#9 – 10 Analyzes the string obtained by the «feed» method
#12 Closing the detector after use
#13 Fixes the result in a variable named «the_answer». This result is stored in a dictionary of 3 elements
#15 Removes the name from the encoding in the dictionary «the_answer» using the keyword «encoding»
#16 Removes the name of the language from the dictionary «the_answer» using the keyword «language»
#17 Prints the encoding and the resulting language.

Option 2. Cases when you have to analyze a large number of files with the extension *«txt»*

```
1  import glob
2  from chardet.universaldetector import UniversalDetector
3
4  the_detector = UniversalDetector()
5  all_path = glob.glob('c:\\work\\*.txt')
6  for pathfile in sorted(all_path):
7      print('In the file ','«',pathfile,'»,we have:'.ljust(6), end=' ')
8      the_detector.reset()
9      for ligne in open(pathfile, 'rb'):
10         the_detector.feed(ligne)
11         if the_detector.done:
12             break
13     the_detector.close()
14     the_answer = the_detector.result
15     for cle in the_answer:
16         if cle == 'encoding':
17             print('Encoding =',the_answer[cle], end=' et ')
18         if cle == 'language':
19             if the_answer[cle]==None:
20                 print("Language = Could not be found",end='.\n')
21             elif len(the_answer[cle])<=1:
22                 print("Language = Could not be found ",end='.\n')
23             else:
24                 print("The language = ",the_answer[cle], end='.\n')
```

Comments on previous program codes

#1 Import of the «glob» module: allows you to recursively select files in a directory according to a given Unix style template
#2 Import of the «UniversalDetector» class of the «chardet» module, which allows detecting character encodings
#4 Creating a detector
#5 Reading all files located in the «work» directory on the "C" drive (in the Windows operating system) and saving them in a list named «all_file» according to the specified template
#6 Opening a loop to view the entire list of saved files after sorting.
#7 Print the file deleted from the directory by adding spaces
#8 Returns the detector to its original state (deletes old data)
#9 Opening the loop to read all data from the selected file in the "rb" mode, line by line.
#10 and #11 Analyze the line received by the detector. The «feed()» method is used to analyze a part of the text to find the information you are looking for. When the detector receives this information, then the "Detector.done" command takes the value «True», which allows us to complete the analysis
#13 Closing the detector after use
#14 Fixes the result in a variable named «the_answer». This result is stored in a dictionary of 3 elements.
#15 Opening a loop to delete the encoding type and language used in the active file
 # From 16 to 24 lines, we print the encoding and language using the keywords of the "dictionary" ('encoding' et 'language').

Option 3

```
1  import glob
2  from chardet.universaldetector import UniversalDetector
3
4  the_detector = UniversalDetector()
5  puth = 'c:\\work\\*.txt'
6  all_path = glob.glob(puth)
7  for pathfile in all_path:
8      print('In the file ','«',pathfile,'»,we have:'.ljust(6), end=' ')
9      the_detector.reset()
10     with open(pathfile, 'rb') as ligne:
11         for ll in ligne:
12             the_detector.feed(ll)
13             if the_detector.done:
14                 break
15     the_detector.close()
16     the_answer = the_detector.result
17
18     le_code = the_answer['encoding']
19     le_lang = the_answer['language']
20     print('Encoding = ',le_code, 'and Language = ',le_lang)
```

Comments on previous program codes
#1 Import of the «glob» module: allows you to recursively select files in a directory according to a given Unix style template *#2 Import of the «UniversalDetector» class of the «chardet» module, which allows detecting character encodings* *#4 Creating a detector* ...

Comments on previous program codes

#5 Specifies the name and full path to the file corresponding to the required template
#6 Reading all files located in the «work» directory on the "C" drive (in the Windows operating system) and saving them in a list named «all_file» according to the specified template
#7 Opening a loop to view the entire list of saved files
#8 Print the file deleted from the directory by adding spaces
#9 Returns the detector to its original state (deletes old data)
#10 We get the object, which is a file, in read mode (data from the file is saved in «ligne». They are placed in the list as an array of data)
#11 Open the loop and perform the iteration process for each row
#12-14 Analyzes the line received by the detector
#15 Closing the detector after use
#14 Fixes the result in a variable named «the_answer». This result is stored in a dictionary of 3 elements.
#15 Opening a loop to delete the encoding type and language used in the active file
#16 Fixes the result in a variable named «the_answer». This result is stored in a dictionary of 3 elements.
#18 removes the name from the encoding in the dictionary «the_answer» using the keyword «encoding»
#19 Removes the name of the language from the dictionary «the_answer» using the keyword «language»
#20 Prints the encoding and the resulting language.

Exercise 1.

We have just presented 3 options for determining the encoding and language of the created file. You need to use all three versions to create a fourth variant. Determine the fourth solution to the previous problem using the information obtained from options 1, 2 and 3. Improve the codes using the instructions: try, except, else and/or finally (see Volume 1).

EN III.5. How to determine the encoding of a web page

It is important to know that if you open a web page and find that the usual characters that you expect look like unusual characters, like strange characters (see Figure 3.5: A, B, C, D) that have nothing to do with plain text (see Figure 3.5: D) what you expect means that you have a problem with character encoding in your web page. This problem is currently rare because the browsers that are used for browsing have become practically intelligent to avoid this incorrect display of characters on web pages.

EN III.5.1. Encoding representation on a web page

Any information in the form of text is written in a document with strictly defined characters that have a certain encoding. Web pages are no exception. The most common and used are:

- KOI8-R (8-bit code page – for encoding Cyrillic letters (letters of the Russian alphabet));
- KOI8-U (8-bit code page – for encoding Cyrillic letters (letters of the Ukrainian alphabet));
- Windows 1251 (CP1251 – for Windows operating system - Russian version);
- ISO-8859-1 (sometimes called latin1 – for French);
- ISO-8859-15 (sometimes called latin9 – a variation of the previous one);

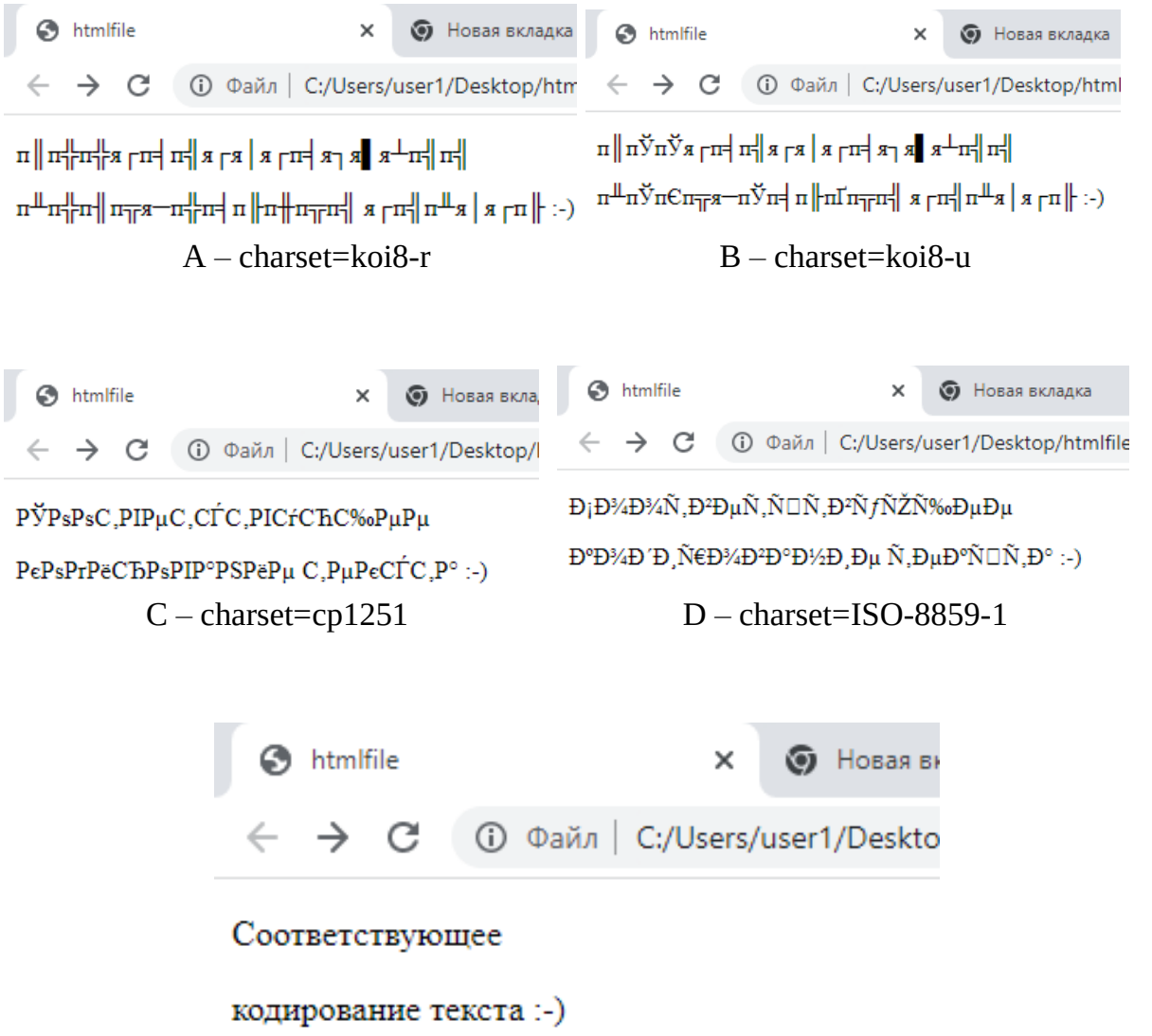
- UTF-8 (8 represents the minimum length in bits that a character can have. Its size is from 1 to 4 bytes); UTF-16 (the minimum length is 16 bits); UTF-32 (here, there is no minimum byte size, it is fixed, and each character occupies 4 bytes in memory). Thus, Unicode theoretically allows encoding all languages: French, English, Chinese, Arabic and many others, not to mention Russian. This is a standard that is currently used by programming languages.

The encoding in a web document is declared as follows: in all HTTP headers sent by the server together with the web page, they must be entered in the standard wording.

In an html web document, you can insert the following codes into the headers:

```
<meta http-equiv="content-type" content="text/html; charset=utf-8">
```

In any case, we must remember that the information provided by the «meta» tag is secondary and does not have the privilege of being the first. First, the information provided by the server will be read, that is, this information will be prioritized if it exists.



A – charset=koi8-r

B – charset=koi8-u

C – charset=cp1251

D – charset=ISO-8859-1

E – charset=utf-8

Figure 3.5. Examples of displaying Russian text in the browser using different encodings (A, B, C, D and E)

EN III.5.2. Determining the encoding of a web page using code

In Python, there are several ways to determine the encoding of a web page. To achieve these goals, it is necessary to use the HTTP/1.1 client library. With its help, an HTTP connection (HttpConnection) passes through a number of «states» that determine the period of time during which the

client can legitimately make another request or receive a response that was given to a specific request. We also use the «request» module, which is an extensible library for opening URLs using various protocols.

In this module, the «urlopen» function accepts a string containing a URL or a request object. It works as follows: opens the URL and returns the results as a file type object; the returned object has several additional methods that we can use: OpenerDirector, urlopen, Request.

Thus, we can say that the «request» module allows you to send a complete request to the server.

Before getting the encoding of any page, you first need to check whether this page exists or not. To do this, we will use the «HTTPResponse» function, which allows you to get a response to the request requested by the server. Thus, using the methods «status», «reason» and «code», we will be able to get the state of the web page, that is, to determine whether it exists or not using data from Table 3.4.

Table 3.4 – Some indicative data when searching for a web page on the site

Status	404	301-303, 307	200	405	400	406
Reason	Not found	Temporarily moved	Ok	Not allowed	Failed request	Unacceptable
Code	404	301-302	200	405	400	406
Semantic meaning	The page does not exist	Redirection Errors	The page found	Unauthorized page	Invalid request	There is no acceptable answer

Option 1

```
1  import http.client as siteweb
2  import urllib.request
3  from chardet.universaldetector import UniversalDetector
4
5  web_link = "http://dl.papacha.ru/index.php/registratsiya1"
6
7  union = siteweb.HTTPConnection('dl.papacha.ru')
8  union.request("HEAD", web_link)
9  answer = union.getresponse()
10 union.close()
11
12 print("Web Page Status = ",answer.status)
13 print("Availability = ",answer.reason)
14
15 if answer.code != 404:
16     web_link1 = urllib.request.urlopen(web_link)
17     the_detector = UniversalDetector()
18     for line in web_link1.readlines():
19         the_detector.feed(line)
20         if the_detector.done: break
21     the_detector.close()
22     web_link1.close()
23     print('Encoding = ',the_detector.result['encoding'], 'and
24                             Language = ',the_detector.result['language'])
25 else:
26     print("The web page does not exist due to: ", answer.reason)
```

Option 2.

```
1  import urllib.request
2  import chardet
3  web_link = "http://dl.papacha.ru/index.php/registratsiya"
4  dataweb = urllib.request.urlopen(web_link).read()
5
6  chardet.detect(dataweb)
7  print(' Encoding = ',chardet.detect(dataweb)['encoding'], 'and
8                  Language = ',chardet.detect(dataweb)['language'])
```

Comments on previous program codes
#1 Importing the urllib. request module to open URLs; *#2 Import the chardet module to determine the character encoding;* *#3 Determining the address at which we can try to detect the encoding;* *#4 Receives the query result on the specified page as an object;* *#6 Read the selected data to determine the character encoding;* *#7 Removes the encoding and language name from the "dictionary", using the keywords «encoding» and «language» to display it on the screen.*

In conclusion, if we want to detect the encoding of several files separately, we need to reuse one *UniversalDetector* object in the same code. With the *detector.reset()* command, we will need to reset the detector at the beginning of each file before re-checking the encoding with the *detector.feed()* command as many times as we have new files to detect. Each time you will need to close it by calling the *detector.close()* command and finally delete the result using the detector.result command; this result will be saved in the “dictionary”.

As you have already noticed, we used the character detection module in Mozilla. It works as follows: a certain component receives bytes as input data and relies on other data to be able to “guess” the set of characters used by the data. This module does not search for information marked (for example: <*meta http-equiv="content-type" content="text/html; charset=utf-8"*>) in web pages. The «chardet» module rather guesses, which means that sometimes the answer is inaccurate, that is, it may be incorrect. So you can use it or make other changes to make the result 100% reliable.

Continued on page 266


FR Chapitre III. TRAITEMENT DE LA CHAÎNE EN PYTHON

FR Introduction

Une chaîne de caractères peut se définir comme étant une suite ordonnée de caractères. Elle peut être traitée comme des caractères unis où chacun d'eux possède un numéro (pour être manipulé) ou code bien précis (pour garder sa valeur). Comme nous l'avions déjà dit, on distingue deux type de chaîne: une première type de chaîne limitée à 256 caractères où chaque caractères a un code unique selon le système de codage; la deuxième type de codage possède juste 2 types de caractères qui sont 0 et 1 – c'est effectivement ce que nous entendons par chaîne d'octets. Il faut noter que chaque caractère est représenté par un nombre d'octets. Ce nombre dépend également du codage utilisé.

FR III.1. Le processus du codage des caractères

Étant donnée qu'il existe plusieurs langages dans le monde, vu la numération croissante des données à travers l'utilisation des ordinateurs, et pour avoir les données codifiées sous forme de texte, il est utile de rappeler que des tables de codage uniformes ont été créées pour utilisation dans le monde (voir tableau 3.1). À chaque langue correspond un ou des types de codage bien précis. Par exemple, dans le but d'afficher visuellement et sans erreurs un texte russe, il sera correcte d'utiliser des encodages cyrilliques. Nous allons à travers des exemples pratiques voir comment par ce langage de programmation, ils se déterminent par le truchement de 2 petites fonctions à savoir: *chr()* et *ord()*. Dans nos navigateurs actuels qui existent, l'encodage est sélectionné automatiquement en fonction du langage désiré. Cela veut dire que l'intervention de l'utilisateur a été d'avance éliminée.

Ainsi, nous pourrons dire que le codage de caractères est une manière

d'assurer la correspondance entre ce que l'utilisateur voit à l'écran et ce que la machine(l'ordinateur) stocke réellement dans la mémoire. La bibliothèque «chardet», est un module qui permet de détecter automatiquement le codage d'un texte. Il est le port du code de détection automatique de l'encodage dans le navigateur connu du nom Mozilla. Pour l'utiliser, il va falloir l'installer.

Tableau 3.1 – Table des codes ASCII et leur correspondance

Encodage ASCII	
Code (en base 10)	Symbole
32	space
33	!
…	…
48	0
49	1
50	2
…	…
97	a
98	b
…	
120	x
121	y
122	z
…	…
127	delete

Ce codage est utilisé sur la plupart des Ordinateurs locaux fonctionnant sur la plate-forme Windows

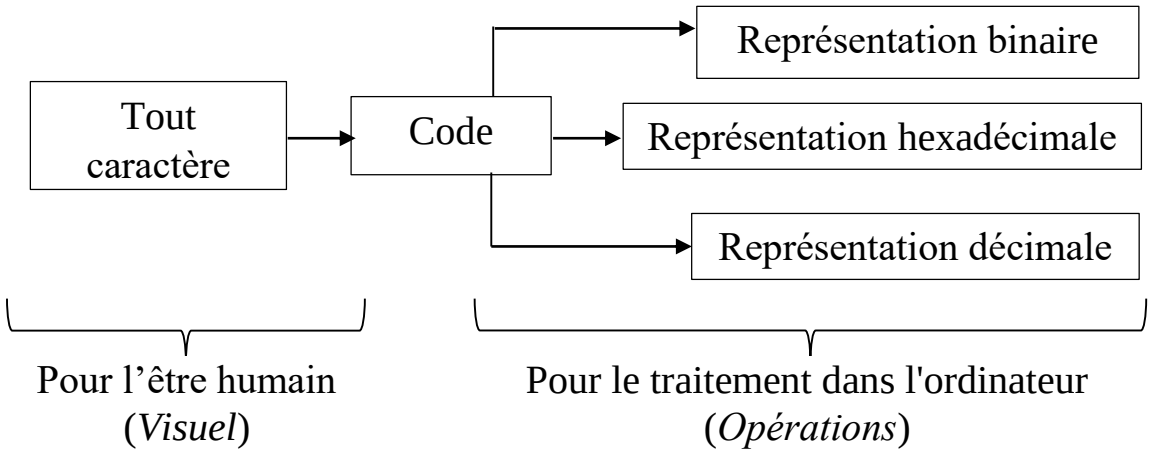


Fig. 3.1. Représentation d'un caractère

Le code est unique dans tout standard utilisé. Notons que le tableau de code ASCII étendu variera selon l'encodage de caractère utilisé.

FR III.2. Manipulation du clavier d'un ordinateur

Pour manipuler le clavier, nous pouvons utiliser différents modules qui seront importés dans le programme de codes.

Module keyboard

Avec cette bibliothèque, nous pouvons prendre le contrôle total du clavier. Nous pouvons créer des raccourcis clavier, faire des enregistrements, bloquer le clavier, attendre la saisie des données et bien d'autres fonctions importantes. Essayons de voir comment à travers ce module, nous pouvons détecter les pressions sur les touches du clavier. Ici, il est possible d'utliser trois méthodes pour détecter ces pressions sur les touches du clavier: *read_key(), is_pressed() et on_press_key()*.

Variante 1. Utilisation de la fonction read_key()

La fonction read_key() permet de lire sur quel bouton du clavier l'utilisateur a appuyé, c'est-à-dire que nous comparons la fonction read_key() (qui renvoie un caractère) à un symbole du clavier.

Exemple d'utilisation.

Déterminer le code ASCII de la touche appuyée sur le clavier

Information supplémentaire:

chr(code) – permet d'obtenir un caractère par son code ASCII;

ord(caractère) – permet de récupérer le code ASCII à travers son caractère.

```
1  import keyboard, time
2
3  while True:
4      symbol = keyboard.read_key()
5      if len(str(symbol))>1 :
6          print("La touche «{0}» est sans code".format(symbol))
7      else:
8          print("Le code de «{0}» est de «{1}»".format(symbol,
9                                                       ord(symbol)))
10     if symbol == "esc":
11         break
12     time.sleep(0.25)
```

Le résultat de l'exécution de ce programme de codes permet de confirmer et de vérifier les données qui se trouvent au tableau 3.1.

Commentaire des codes précédents
#1 Importation du module «keyboard» pour déterminer le code ASCII de la touche appuyée sur le clavier, importation également du module «time» pour ajouter un temps de latence au programme. *#3 Se déroule si la condition (des instructions) est vraie.* *#4 Attend, qu'une touche sur le clavier soit appuyée.* *#5 Détermine, si c'est un caractère de contrôle en vérifiant si sa longueur est strictement supérieur à 1.* *#6 Imprime le caractère de contrôle.* *#7 Si ce n'est pas un caractère de contrôle.* *#8 Imprime le caractère appuyé et son code ASCII.* *#10 Si le caractère appuyé est Escape.* *#11 Sortie de la boucle.* *#12 Ajoute un temps de latence (1/4 de seconde) pour les claviers sensibles au toucher.*

Variante 2. Utilisation de la fonction is_pressed()

La fonction *is_pressed()* prend un paramètre à entrée. Ce paramètre est considéré comme n'importe quel caractère, et s'il correspond au bouton sur laquelle l'utilisateur a appuyé, alors dans ce cas, il retournera la valeur booléenne « True », dans le cas contraire, il retourne la valeur « False ».

Exemple d'utilisation.

Déterminer le code ASCII de la touche appuyée sur le clavier

```
1  import keyboard
2  from time import sleep
3
4  print("Appuyez la touche «Escape» pour terminer")
5  while True:
6      symbol = keyboard.read_key()
7      if keyboard.is_pressed("esc"):
8          print("Fin !")
9          break
10     elif len(symbol)>1:
11         print("Vous avez appuyé le caractère de contrôle
12                                                 {0}".format(symbol))
13     else :
14         print("Vous avez appuyé le caractère {0}".format(symbol))
15         print("Le code ASCII de «{0}» est: «{1}»".format(symbol,
16                                                     ord(symbol)))
17     sleep(0.5)
```

Commentaire des codes précédents
#1 Importation du module «keyboard» pour déterminer le code ASCII de la touche appuyée sur le clavier. *#2 Importation juste de la fonction sleep() qui se trouve dans le module «time» pour ajouter un temps de latence au programme.* *#5 Ouverture de la boucle si la condition (des instructions) est vraie.* *#6 Utilisation de la fonction «read_key()» pour capter un caractère du clavier (attend, qu'une touche sur le clavier soit appuyée).* *#7 Vérifie si la touche appuyée est « Escape »* *#9 Sortie de la boucle.* *#10 Détermine, si c'est un caractère de contrôle en vérifiant si sa longueur est strictement supérieur à 1.* *#11 Imprime le caractère de contrôle.* *#13 Si ce n'est pas un caractère de contrôle.* *#15 Imprime le caractère appuyé et son code ASCII.* *#17 Ajoute un temps de latence (1/2 de seconde) pour les claviers sensibles au toucher (Pas obligatoire).*

Variante 3. Utilisation de la fonction on_press_key()

La fonction «on_press_key()» utilise deux paramètres comme entrée, le premier est le symbole ou le caractère que nous voulons obtenir ou identifier et le second est une fonction du type «lambda _». Dans le cas où l'utilisateur appuie sur le bouton qui correspond à la valeur du premier paramètre, alors il n'exécutera que la fonction qui est située après «lambda _» du deuxième paramètre.

Exemple d'utilisation.

Montrer l'utilité de la fonction « *on_press_key()* ». Comment pouvons-nous l'utiliser différemment.

```
1  import keyboard
2
3  def imprime_une_phrase():
4      print('Avec la touche «esc», nous pouvons marquer la fin.')
5
6  keyboard.on_press_key("esc", lambda _: imprime_une_phrase())
7  keyboard.on_press_key("s", lambda _: keyboard.write("La touche
8                                        «s» a été appuyée", delay = 0.1))
9  keyboard.on_press_key("z", lambda _: print("La touche «z» a été
10                                  appuyée"))
```

Du moment où la fonction «keyboard» renvoie l'événement à cette même fonction, il est indispensable d'utiliser le symbole «_».

Commentaire des codes précédents
#1 Importation du module «keyboard» pour déterminer le code ASCII de la touche appuyée sur le clavier. *#3 et 4 Définition de la fonction pour imprimer le texte* *#6 Si la touche "Escape" est appuyée, alors nous assistons à une exécution de la fonction "imprime_une_phrase()"* *#7 Si la touche "s" est appuyée, alors nous assistons à une exécution de la fonction «write» du module «keyboard». Le texte va s'afficher progressivement symbole par symbole avec un temps de latence qui sera déterminé par la méthode «delay()» de la fonction «write()».* *#9 Si la touche «z» est appuyée, alors nous assistons à une exécution de la fonction «print». La différence entre «print» et «write» se situe au niveau où avec la fonction «write» de la bibliothèque «keyboard», le texte est imprime là où le curseur se trouve. Il peut même être actif hors du console, là où a été exécuté le programme.*

Les autres fonctions de ce module:

keyboard.wait('esc')

Appuyez sur la touche Echap pour continuer d'executer les instructions suivantes. À la place de « esc », nous pouvons y mettre n'importe quelle touche du clavier.

keyboard.press('caps_lock')

Permet au programme d'appuyer la touche «caps_lock» du clavier. À la place de «caps_lock», nous pouvons y mettre n'importe quelle touche du clavier.

keyboard.release('caps_lock')

Permet au programme de libérer la touche 'caps_lock' du clavier qui a été appuyée.

keyboard.press_and_release('shift+s, space, =,space, 9')

Permet d'écrire la commande S = 9. À leurs places, nous pouvons mettre d'autres mots clés et symboles.

keyboard.add_hotkey('enter', lambda: keyboard.write("Vous aviez appuyé la touche Enter!!!"))

Si la touche "Enter" est appuyée, alors nous assistons à une exécution de la fonction « write » du module « keyboard ». À la place de *« keyboard.write »*, nous pouvons y mettre une autre fonction à executer. De même, à la place de « enter », nous pouvons mettre une autre touche ou alors une combinaison de touche : par exemple «shift+r» (dont, il faudra appuyer à la fois la touche «shift» et la touche «r»)

keyboard.send('touche_du_clavier')

Permet d'appuyer et de libérer la touche qui se trouve entre parenthèse de la fonction.

keyboard.add_abbreviation(source_text, replacement_text, match_suffix = False, délai d'expiration = 2)

Permet d'accélérer ou d'automatiser la rédaction d'un petit texte(***replacement_text***) par un mot clé(***source_text***) ou symbole. Vous

mettez le mot clé (*symbole*), et après avoir appuyé sur la barre d'espace, il est directement remplacé par l'autre qui est le « ***replacement_text*** ».

Exemple : *keyboard.add_abbreviation('@ ', 'fiston@gmx.fr')*

list_ev = keyboard.record(until='touche_du_clavier')

Enregistre tous les événements des touches du clavier jusqu'à ce que l'utilisateur appuie sur la touche de raccourci donnée (***touche_du_clavier***). Après, ces événements sont enregistrés dans une liste (***list_ev***) qui peut être consulter ou joué par la fonction "play".

keyboard.play(list_ev, speed_factor=1.0)

Ressort tous les événements en maintenant les intervalles de temps relatifs pendant l'enregistrement. Si le paramètre speed_factor est inférieur ou égal à 0, alors les actions seront rejouées aussi vite que le système d'exploitation le permet.

Exemple d'utilisation de « record » et « play »

```
1  import keyboard
2
3  try:
4      keyboard.write("Tout ce que vous allez faire du clavier sera
5                      enregistré. La fin de l'enregistrement se déroulera
6                      après que la touche «escape» puisse être
7                      appuyée.\n")
8      record = keyboard.record(until="esc", suppress=False,
9                      trigger_on_release=False)
10     print("Liste enregistrée = ",record)
11     print("Copie exacte de ce qui a été enregistrée")
12     keyboard.play(record)
13 except Exception as er:
14     print("Nous avons l'erreur suivante: {0}".format(er.args[0]))
```

Commentaire des codes précédents
#1 Importation du module «keyboard» pour faire un enregistrement de toutes les touches du clavier. *#3 et 13 Sont utilisés pour vérifier le code qui peut avoir des erreurs* *#4 Imprime le texte à travers la fonction write. Le texte est imprime là où le curseur se trouve. Après le curseur va à la ligne grâce à la commande « \n »* *#8 Enregistre tous les événements des touches du clavier jusqu'à ce que l'utilisateur appuie sur la touche « esc ». Les paramètres « suppress» et, «trigger_on_release» peuvent être éliminés.* *#10 Imprime les événements qui sont dans une liste et nous obtenons le résultat de la figure 2.* *#12 Ressort identiquement tous les événements qui ont été enregistrés.*

Données introduites ou saisies pendant l'enregistrement>>> 12345
Liste enregistrée = [KeyboardEvent(1 down), KeyboardEvent(1 up), KeyboardEvent(2 down), KeyboardEvent(2 up), KeyboardEvent(3 down), KeyboardEvent(3 up), KeyboardEvent(4 down), KeyboardEvent(4 up), KeyboardEvent(5 down), KeyboardEvent(5 up), KeyboardEvent(esc down)]
Résultat de la lecture d'un enregistrement sur une liste à l'aide de la fonction «*play()*» >>>12345

Fig. 3.2. Données saisies et résultats de l'enregistrement faite par la fonction «*record*» du module «keyboard»

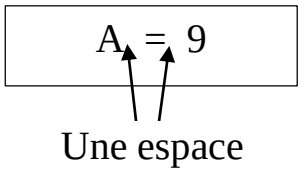
Module pynput

Comparativement à la précédente bibliothèque (keyboard), ce module est plus fonctionnel et permet non seulement de prendre également tout le contrôle du clavier, mais permet aussi de manipuler les mouvements de la

souris à travers les coordonnées. Essayons de voir comment à travers ce module, nous pouvons aussi détecter les pressions sur les touches du clavier et de la souris.

Comment appuyer un clavier et le relâcher ?

La méthode « ***press*** » permet d'appuyer une touche, alors que la méthode « ***release*** » permet de relâcher cette touche. A l'exemple qui suit, nous allons voir comment elles peuvent être utilisées pour écrire l'équation :




```
1  from pynput.keyboard import Key, Controller
2
3  try:
4      keyboard = Controller()
5
6      keyboard.press(Key.caps_lock)
7      keyboard.press('a')
8      keyboard.release('a')
9      keyboard.press(Key.space)
10     keyboard.release(Key.space)
11     keyboard.press('=')
12     keyboard.release('=')
13     keyboard.press(Key.space)
14     keyboard.release(Key.space)
15     keyboard.press('9')
16     keyboard.release('9')
17     keyboard.release(Key.caps_lock)
18 except Exception as er:
19     print("Nous avons l'erreur suivante: {0}".format(er.args[0]))
```

Nous savons également que pour écrire une lettre majuscule, il est possible d'utiliser la touche « Shift » à la place de « Caps lock ». Ainsi la nouvelle version du code sera ainsi :

```
1  from pynput.keyboard import Key, Controller
2
3  try:
4      keyboard = Controller()
5
6      with keyboard.pressed(Key.shift):
7          keyboard.press('a')
8          keyboard.release('a')
9      keyboard.press(Key.space)
10     keyboard.release(Key.space)
11     keyboard.press('=')
12     keyboard.release('=')
13     keyboard.press(Key.space)
14     keyboard.release(Key.space)
15     keyboard.press('9')
16     keyboard.release('9')
17 except Exception as er:
18     print("Erreurs rencontrées: ",er)
19
```

Commentaire des codes précédents

#1 Importation des classes « Key » et « keyboard » à partir du module «pynput.keyboard».
#4 Détermine « keyboard » comme étant le contrôleur objet du clavier venant du module « pynput ».

#6 En appuyant la touche « Shift », nous pressons également la touche « a » (#7) et après nous la lâchons(#8).
#9 ...

Comment afficher un message à partir de ce module?

```
1  from pynput.keyboard import Controller
2
3  le_clavier = Controller()
4  le_clavier.type("Un texte quelconque!")
```

ou bien

```
1  from pynput.keyboard import Controller
2
3  def ShowMessageKeyboard():
4      clavier = Controller()
5      clavier.type(" Un texte quelconque!")
6
7  ShowMessageKeyboard()
```

Commentaire des codes précédents
#4 ou #5 Nous aurons juste le message suivant à l'écran : Un texte quelconque!

Comment faire ressortir visuellement dans un console tous les boutons appuyés avec leurs codes

Pour cela, nous devons utiliser la bibliothèque «Listeners» (Auditeurs) pour pouvoir écouter les frappes du clavier.

```
from pynput.keyboard import Listener
def Reset_key(key):
    print('La touche %s a été relaché\n'%(key))

def WriteOnScreen (key):
    try:
        if str(key) == 'Key.esc' :
            return exit()
        print('Vous aviez appuyé la touche ',key)

        sym = str(key)
        if len(sym) == 3:
            sym = sym.replace("'","")
            val = str(ord(sym))
        else:
            sym = sym.rsplit('.')[1]
            sym = " "+sym+" "
            val = " None"

        print("La touche '%s' a pour code, la valeur '%s'" %(sym, val))
    except Exception as er:
        print("Nous avons l'erreur suivante: {0}".format(er.args[0]))

print("Veillez taper sur «esc» pour arrêter le programme!")
print("Appuyer sur une touche pour déterminer son code. \n")

with Listener
            (on_press=WriteOnScreen, on_release=Reset_key) as lis:
    lis.join()
```

Commentaire des codes précédents
#1 Nous importons le clavier(keyboard) du module «pynput» comme étant un objet de type auditeur (Listener) *#2 Fonction pour lâcher la touche appuyée* *#5-23 Fonction pour appuyer et déterminer son code* *#7 Si la touche appuyée est « Escape », alors arrêt du programme(#9)* *#12 Vérifie si la bouton appuyé n'est pas une touche de contrôle* *#13 Permet de retirer les apostrophes ou guillemets qui accompagnent le symbole* *#14 Retire le code du caractère qui a été appuyé.* *#16 Divise la chaîne «sym» en deux parties en utilisant le séparateur « . » et retient la deuxième partie de la chaîne.* *#17 Ajoute un espace en avant et après pour éviter une confusion visuelle avec les autres caractères* *#18 Donner la valeur «None» aux boutons de contrôle* *#20 Imprimer la touche sur l'écran(le console)* *#27-28 À chaque fois, avec le paramètre «on_press» (qui veut dire en appuyant une touche), nous allons appeler la fonction «WriteOnScreen» et avec le paramètre « on_release » (qui veut dire la relâche de la touche), nous allons exécuter la fonction « Reset_key ».* *#29 Joint de manière itérable tous les boutons appuyés à la fin de la chaîne*

Comment faire ressortir dans un fichier tous les boutons du clavier qui ont été appuyés sans interprétation

```
1  from pynput.keyboard import Listener
2
3  def Reset_key(key):
4      None
5
6  def SaveToFile(key):
7      try:
8          if str(key) == "Key.esc" :
9              return exit()
10         if str(key) == "Key.space" :
11             key=" "
12         sym = str(key)
13         if len(sym) <= 3:
14             sym = sym.replace(" ' ","")
15         else:
16             sym = sym.rsplit(".")[1]
17             sym = " "+sym+" "
18
19         with open("log_punput.txt", "a") as f:
20             f.write(sym)
21
22     except Exception as er:
23         print("Nous avons l'erreur suivante: {0}".format(er.args[0]))
24
25 print("Veillez taper sur «esc» pour arrêter le programme!")
26 print("Appuyer sur une touche pour déterminer son code. \n")
27
28 with Listener (on_press=SaveToFile, on_release=Reset_key) as lis:
29     lis.join()
```

Codes pour sauvegarder le caractère dans le fichier

Commentaire des codes précédents

#1 Nous importons le clavier(keyboard) du module «pynput» comme étant un objet de type auditeur (Listener)
#3 Fonction pour lâcher la touche appuyée
#6-23 Fonction pour appuyer et déterminer son code
#8 Si la touche appuyée est « Escape », alors arrêt du programme(#9)
#13 Vérifie si le bouton appuyé n'est pas une touche de contrôle
#14 Permet de retirer les apostrophes ou guillemets qui accompagnent le symbole

#15 Dans ce cas, retire le code du caractère qui a été appuyé.
#16 Divise la chaîne «sym» en deux parties en utilisant le séparateur « . » et retient la deuxième partie de la chaîne.
#17 Ajoute un espace en avant et après pour éviter une confusion visuelle avec les autres caractères
#19-20 À chaque fois sauvegarde dans le fichier la touche appuyée
#25-26 Imprimer un message sur l'écran(le console)
#28 À chaque fois, avec le paramètre «on_press» (qui veut dire en appuyant une touche), nous allons appeler la fonction «SaveToFile» et avec le paramètre « on_release » (qui veut dire la relâche de la touche), nous allons exécuter la fonction « Reset_key ».
#29 Joint de manière itérable tous les boutons appuyés à la fin de la chaîne

Avec ces codes, nous comprenons comment il est possible de contrôler toutes les manipulations faites sur le clavier. Ces manipulations peuvent être affichées sur le console du logiciel de votre ordinateur, comme il peuvent de manière discrète être stockées dans un fichier quelconque.

Nous pouvons importer la bibliothèque d'une autre manière :

« ***from pynput import keyboard*** »

Ainsi, le programme devrait subir une petite modification aux codes de la ligne 28 :

« with ***keyboard.***Listener (on_press=SaveToFile, on_release=Reset_key) as lis: ». Il faut noter que les deux instructions ici sont identique lorsque nous utilisons la Méthode 1 (Voir tableau 3.2):

Tableau 3.2 – Différence dans l'importation des modules (Chaîne et objet)

Méthode 1	Méthode 2
from pynput import keyboard	*from pynput.keyboard import Listener*
if key == keyboard.Key.esc:	-
if str(key) == 'Key.esc':	*if str(key) == 'Key.esc':*

Exercice 1

Essayez de modifier le code de manière à ce que les données entrées puissent être identique aux données stockées dans le fichier. C'est-à-dire qu'il faudra interpréter certaines touches et certaines combinaisons de touches avant de les sauvegarger. Par exemple, en appuyant simultanément sur «Shift» et «s», nous devrions obtenir «S» en majuscule dans le fichier ; en appuyant la touche « Enter », nous devrions aller à la ligne ; et ainsi de suite pour les autres touches et combinaisons sur le clavier. Pour y parvenir, vous pouviez utiliser les informations du tableau 3.3.

Tableau 3.3 – Les boutons de contrôle du clavier

Nom dans Punpyt	Nom officiel et position sur le clavier
Key.ctrl_l	Ctrl à gauche
Key.ctrl_r	Ctrl à droite
Key.alt_l	Alt à gauche
Key.alt_gr	Alt à droite
Key.shift	Shift à gauche
Key.shift_r	Shift à droite
Key.enter	Enter
Key.backspace	Backspace
Key.delete	Delete

Fin du tableau 3.3

Key.space	Space
Key.caps_lock	Caps lock
Key.home	Home
Key.end	End
Key.page_up	Page up
Key.page_down	Page down
Key.num_lock	Num lock
Key.pause	Pause
Key.print_screen	Print screen
Key.insert	Insert
Key.tab	Tab
Key.f1	F1
Key.f2	F2
Key.f3	F3
Key.f4	F4
Key.f5	F5
Key.f6	F6
Key.f7	F7
Key.f8	F8
Key.f9	F9
Key.f10	F10
Key.f11	F11
Key.f12	F12
Key.up	Up
Key.down	Down
Key.left	Left
Key.right	Right
Key.esc	Esc – Échap
Key.cmd	⊞ Win
Key.menu	Menu

FR III.3. Manipulation de la souris d'un ordinateur

La souris («mouse» en anglais) est un périphérique de pointage qui permet de déplacer le curseur sur l'écran d'un ordinateur avec des coordonnées ***x*** et ***y*** (Voir la figure 3.3). Il permet également de manipuler et de sélectionner des objets à partir des « clics » ou actions sur deux boutons principaux (un à gauche et l'autre à droite) déterminés. Il est possible également de faire déplacer ce curseur à partir des codes dans python. Pour cela, nous sommes dans l'obligation d'utiliser la bibliothèque de son choix parmi: le module «mouse», «pynput» et autres. Dans notre livre, nous allons essayer de voir comment manipuler la souris en utilisant la bibliothèque «mouse» qui se trouve dans le module «pynput».

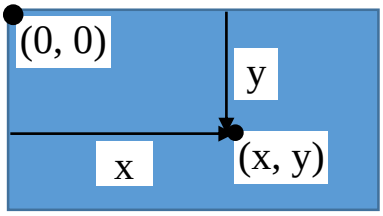


Fig. 3.3. Position des coordonnées du pointeur de la souris sur l'écran

Comment changer la position de la souris d'un point à un autre ?

```
1  from pynput.mouse import Controller
2
3  def mouvMouse(x, y):
4      try:
5          mouse = Controller()
6          mouse.position = (x, y)
7      except Exception as er:
8          print("Nous avons l'erreur suivante: {0}".format(er.args[0]))
9
10 def inputcoord():
11     global x,y
12     x = input("Introduire la valeur de x: ")
13     y = input("Introduire la valeur de y: ")
14
15 inputcoord()
16 mouvMouse(x,y)
```

Commentaires des codes précédents

#1 Nous importons la souris(mouse) du module «pynput» comme étant un objet de contrôle (Controller)
#3 La fonction pour changer la position du pointage
#5 Nous faisons de la souris(mouse) un objet de contrôle
#6 Nous allons déplacer la position du pointage de la souris aux coordonnées x et y qui arriverons
#10 La fonction pour donner aux variables x et y des valeurs(coordonnées)
#11 Faire de x et y des variables globales
#12 Demande d'introduction de la valeur x
#13 Demande d'ntroduction de la valeur y

...

Commentaires des codes précédents

#15 Appelle la fonction «inputcoord()» pour introduire les coordonnées x et y
#16 Appelle la fonction «mouvMouse()» pour déplacer l'indicateur (pointeur) visuel de la souris.

Nous pouvons modifier le code précédent en supprimant les deux variables globales *x* et *y*. Le résultat est le suivant:

```
1  from pynput.mouse import Controller
2
3  def mouvMouse(x, y):
4      try:
5          mouse = Controller()
6          mouse.position = (x, y)
7      except Exception as er:
8          print("Nous avons l'erreur suivante: {0}".format(er.args[0]))
9
10 def inputXcoord():
11     x = input("Introduire la valeur de x: ")
12     return x
13
14 def inputYcoord():
15     y = input("Introduire la valeur de y: ")
16     return y
17
18 mouvMouse(inputXcoord(),inputYcoord())
```

Commentaires sur les codes précédents

 # 10 Fonction pour attribuer une valeur à la variable x (coordonnée)
 # 14 Fonction pour attribuer une valeur à la variable y (coordonnée)
#18 Appelle la fonction «mouvMouse()» pour déplacer le pointeur de la souris après avoir obtenu les coordonnées: inputXcoord() et inputYcoord()

Comment déterminer les coordonnées du pointeur de la souris pendant son mouvement?

```
1  from pynput.mouse import Listener
2
3  def ShowTheCoord(x, y):
4      print("La position actuelle de la souris est: {0}".format((x,y)))
5
6
7  with Listener (on_move=ShowTheCoord) as lis:
8      lis.join()
```

Commentaires des codes précédents

#1 Nous importons la souris(mouse) du module «pynput» comme étant un objet de type auditeur (Listener).
#3-4 La fonction qui permet d'imprimer sur le console du logiciel les coordonnées du pointeur de la souris pendant son mouvement.

#7 À chaque fois, avec le paramètre «on_move» (qui veut dire en déplaçant le pointeur de la souris), nous allons appeler la fonction «ShowTheCoord»
#8 Joint de manière itérable toutes les coordonnées du pointeur de la souris chaque fois à la fin de la chaîne.

Après l'exécution de ce fichier, nous aurons l'affichage des coordonnées du pointeur de la souris lors uniquement de son mouvement. Un exemple de cette visualisasion se trouve sur la figure 3.4.

```
La position actuelle de la souris est: (567, 215)
La position actuelle de la souris est: (568, 215)
La position actuelle de la souris est: (569, 215)
La position actuelle de la souris est: (569, 215)
La position actuelle de la souris est: (570, 215)
La position actuelle de la souris est: (581, 217)
La position actuelle de la souris est: (582, 217)
La position actuelle de la souris est: (583, 217)
La position actuelle de la souris est: (584, 217)
La position actuelle de la souris est: (584, 217)
La position actuelle de la souris est: (584, 218)
La position actuelle de la souris est: (584, 218)
La position actuelle de la souris est: (584, 219)
La position actuelle de la souris est: (584, 219)
La position actuelle de la souris est: (584, 220)
La position actuelle de la souris est: (584, 220)
```

Fig. 3.4. Résultat de l'exécution du programme code précédent

Comment autrement obtenir la position du pointeur de la souris en utilisant le module intégré «ctype»

Déterminons à quelles coordonnées se trouvent le pointeur de la souris après un intervalle de temps bien déterminé. Nous allons utiliser le module «time» pour fixer le temps et le module «ctypes» (*qui permet de communiquer ou d'interagir avec les codes du langage «C», d'appeler des fonctions dans les fichiers à extension «dll» ou alors d'appeler certaines fonctions tierces*) pour trouver ces coordonnées.

```
1  import time
2  import ctypes
3
4  def SourisGetPointeur():
5      tochka = le_format_xy()
6      Interface.GetCursorPos(ctypes.byref(tochka))
7      if int(tochka.L)==0 and int(tochka.H)==0:
8          global la_fin
9          la_fin = True
10     return {"L": tochka.L, "H": tochka.H}
11
12 print("Pour arrêter le programme, placez le pointeur de la souris sur
13                     les coordonnées \nx=0 et y=0 (c'est-à-dire dans le
14                     coin supérieur gauche de l'écran)")
15 while True:
16     la_fin = False
17     Interface = ctypes.windll.user32
18     class le_format_xy(ctypes.Structure):
19         _fields_ = [("L", ctypes.c_uint), ("H", ctypes.c_uint)]
20
21     coord = SourisGetPointeur()
22     print(coord)
23
24     if la_fin:
25         print("La FIN  du PROGRAMME")
26         break
27
28     time.sleep(2)
```

FR III.4. Comment déterminer l'encodage d'un fichier texte

Il est important de savoir quel codage a été utilisé pour créer un texte ou une page web. Cette connaissance peut nous permettre de créer des logiciels qui par rapport au langage utilisé, puissent choisir automatiquement le type de codage pour une affiche correcte des informations à l'écran de son ordinateur.

Voici des exemples de codes qui permettent de déterminer le codage d'un seul ou d'un groupe de fichiers texte dans un répertoire bien déterminé.

Version 1. *Cas où nous avons à analyser un seul fichier*

```
1  from chardet.universaldetector import UniversalDetector
2
3  le_detecteur = UniversalDetector()
4  puth = 'c:\\work\\my_file.txt'
5  le_detecteur.reset()
6
7  with open(puth, 'rb') as ligne:
8      for elt in ligne:
9          le_detecteur.feed(elt)
10         if le_detecteur.done:
11             break
12     le_detecteur.close()
13     le_resultat = le_detecteur.result
14
15     le_code = le_resultat['encoding']
16     le_lang = le_resultat['language']
17     print('Codage = ',le_code, 'et Langage = ',le_lang)
```

Commentaire des codes précédents

#1 Importation de la classe «UniversalDetector» du module «chardet» qui permet de détecter les encodages de caractères.
#3 Création du détecteur.
#4 Spécifie le nom et le chemin d'accès complet du fichier respectant le modèle voulu.
#5 Réinitialise le détecteur à son état d'origine (efface les anciennes données).
#7 Nous obtenons l'objet qui est le fichier en mode lecture (Les données du fichiers s'enregistrent dans «ligne». Elles sont positionnées en liste comme une massive de données).
#8 Nous ouvrons la boucle et effectuons un processus d'itération sur chaque ligne.
#9-10 Analyse la ligne obtenue par la methode «feed».
#12 Fermeture du détecteur après utilisation
#13 Fixe le résultat obtenu dans unevariable nommé «le_resultat». Ce résultat est stocké dans un dictionnaire à 3 éléments.
#15 Retire le nom du codage dans le dictionnaire «le_resultat» en utilisant le mot clé «encoding».
#16 Retire le nom du langage dans le dictionnaire «le_resultat» en utilisant le mot clé «language».
#17 Imprime le codage et la langage obtenu.

Version 2. *Cas où nous avons à analyser un nombre élevé de fichiers avec pour extension « txt »*

```
import glob
from chardet.universaldetector import UniversalDetector

le_detecteur = UniversalDetector()
all_path = glob.glob('c:\\work\\*.txt')
for pathfile in sorted(all_path):
    print('Dans le fichier','«',pathfile,'», nous avons:'.ljust(6), end=' ')
    le_detecteur.reset()
    for ligne in open(pathfile, 'rb'):
        le_detecteur.feed(ligne)
        if le_detecteur.done:
            break
    le_detecteur.close()
    le_resultat = le_detecteur.result
    for cle in le_resultat:
        if cle == 'encoding':
            print('Codage =',le_resultat[cle], end=' et ')
        if cle == 'language':
            if le_resultat[cle]==None:
                print("Langage = N'a pas pu détecter",end='.\n')
            elif len(le_resultat[cle])<=1:
                print("Langage = N'a pas pu détecter",end='.\n')
            else:
                print("le langage = ",le_resultat[cle], end='.\n')
```

Commentaire des codes précédents

#1 Importation du module glob: permet de sélectionner récursivement les fichiers dans un répertoire selon un modèle donné du style Unix
#2 Importation de la classe UniversalDetector du module chardet qui permet de détecter les encodages de caractères
#4 Création du détecteur
#5 Lecture de tous les fichiers qui se trouvent dans le répertoire "work" du disque "C" (dans le système opérationnel Windows) et les stockent dans une liste nommée « all_file » selon le modèle spécifié.
#6 Ouverture de la boucle pour parcourir toute la liste des fichiers enregistrés après le tri.
#7 Imprime le fichier retiré du répertoire en ajoutant des espaces
#8 Réinitialise le détecteur à son état d'origine (efface les anciennes données)
#9 Ouverture d'une nouvelle boucle pour lire dans le fichier choisi toutes les données en mode "rb", ligne par ligne.
#10 et #11 Analyse la ligne obtenue par le détecteur. La méthode feed() permet d'analyser une partie du texte, afin de détecter l'information recherchée. Lorsque le détecteur obtient cette information, la commande "Detector.done" prend alors la valeur "True", et ce qui nous permet de mettre fin à notre analyse.
#13 Fermeture du détecteur après utilisation
#14 Fixe le résultat obtenu dans unevariable nommé "le_resultat. Ce résultat est stocké dans un dictionnaire à 3 éléments.
#15 Ouverture de la boucle pour retirer le type de codage et le langage utilisé dans le fichier actif
 # De la 16 à la 24 ligne, nous faisons imprimer le codage et la langage en utilisant les mots clés du dictionnaire ('encoding' et 'language').

Version 3

```
1  import glob
2  from chardet.universaldetector import UniversalDetector
3
4  le_detecteur = UniversalDetector()
5  puth = 'c:\\work\\*.txt'
6  all_path = glob.glob(puth)
7  for pathfile in all_path:
8      print('Dans le fichier','«',pathfile,'», nous avons:'.ljust(6), end=' ')
9      le_detecteur.reset()
10     with open(pathfile, 'rb') as ligne:
11         for ll in ligne:
12             le_detecteur.feed(ll)
13             if le_detecteur.done:
14                 break
15     le_detecteur.close()
16     le_resultat = le_detecteur.result
17
18     le_code = le_resultat['encoding']
19     le_lang = le_resultat['language']
20     print('Codage = ',le_code, 'et Langage = ',le_lang)
```

Commentaire des codes précédents
#1 Importation du module glob: permet de sélectionner récursivement les fichiers dans un répertoire selon un modèle donné du style Unix. *#2 Importation de la classe UniversalDetector du module chardet qui permet de détecter les encodages de caractères.* *#4 Création du détecteur.* ...

Commentaire des codes précédents

#5 Spécifie le nom et le chemin d'accès complet du fichier respectant le modèle voulu.
#6 Lecture de tous les fichiers qui se trouvent dans le répertoire "work" du disque "C" (dans le système opérationnel Windows) et les stockent dans une liste nommée « all_file » selon le modèle spécifié.
#7 Ouverture de la boucle pour parcourir toute la liste des fichiers enregistrés.
#8 Imprime le fichier retiré du répertoire en ajoutant des espaces.
#9 Réinitialise le détecteur à son état d'origine (efface les anciennes données).
#10 Nous obtenons l'objet qui est le fichier en mode lecture (Les données du fichiers s'enregistrent dans "ligne". Elles sont positionnées en liste comme une massive de données).
#11 Nous ouvrons la boucle et effectuons un processus d'itération sur chaque ligne.
#12-14 Analyse la ligne obtenue par le détecteur
#15 Fermeture du détecteur après utilisation
#14 Fixe le résultat obtenu dans unevariable nommé «le_resultat». Ce résultat est stocké dans un dictionnaire à 3 éléments.
#15 Ouverture de la boucle pour retirer le type de codage et le langage utilisé dans le fichier actif
#16 Fixe le résultat obtenu dans unevariable nommé «le_resultat». Ce résultat est stocké dans un dictionnaire à 3 éléments.
#18 Retire le nom du codage dans le dictionnaire «le_resultat» en utilisant le mot clé «encoding».
#19 Retire le nom du langage dans le dictionnaire «le_resultat» en utilisant le mot clé «language».
#20 Imprime le codage et la langage obtenu.

Exercice 1.

Nous venons de présenter 3 variantes pour déterminer le codage et le langage d'un fichier qui a été créé. Nous vous demandons d'utiliser les trois versions pour établir une quatrième variante. Déterminer cette quatrième variante de l'exercice précédent à l'aide des informations reçues des versions 1, 2 et 2. Perfectionner les codes en utilisant les instructions: try, except, else et/ou finally (cf. Tom 1).

FR III.5. Comment déterminer l'encodage d'une page web

Nous devons savoir que si vous ouvrez un page web et que vous constatez que les caractères normaux que vous attendez ressemblent à des symboles inhabitués, à des caractères bizarres (voir fig. 3.5 : A, B, C, D) qui n'ont rien avoir avec le texte simple (voir fig. 3.5 : E) que vous attendez, cela veut dire que vous avez un problème d'encodage des caractères sur votre page web. Ce problème se rencontre actuellement rarement, car les navigateurs dans lesquels ils se trouvent en permanence, sont devenus pratiquement intelligents, afin d'éviter cette mauvaise affichage des caractères sur les pages web.

FR III.5.1. Représentation de l'encodage dans une page Web

Toute information sous forme du texte est enregistrée dans le document avec de caractères bien précis et appartenant à un codage spécifique. Les pages web, ne font pas exception. Les plus fréquents et utilisés sont:

- KOI8-R (une page de codes de 8 bits - pour coder les lettres de l'alphabet cyrillique (lettres de l'alphabet russe))
- KOI8-U (une page de codes de 8 bits - pour coder les lettres de l'alphabet cyrillique (lettres de l'alphabet ukrainien))
- Windows 1251 (cp1251 - pour le système opérationnel Windows - version russe)
- ISO-8859-1 (parfois appelé latin1 - pour le français)

- ISO-8859-15 (parfois appelé latin9 - variation du précédent)
- UTF-8 (8 représente la longueur minimale en bit que peut avoir un caractère. Sa dimension est comprise entre 1 et 4 octects) ; UTF-16 (la longueur minimale est 16 bits) ; UTF-32 (Il n'y a pas de dimension minimale d'octect, elle est fixe et chaque caractère occupe 4 octets dans la mémoire). Ainsi, unicode permet théoriquement d'encoder toutes les langues, du russe, de l'anglais, du chinois, de l'arabe et bien d'autres sans oublier le français. Il est le standard qui s'utilise actuellement par les langages de programmation.

L'encodage dans un document web est déclaré de la manière suivante: dans toutes les en-têtes HTTP envoyées par le serveur avec la page web, il est nécessaire de les introduire selon une formulation standard.

Dans un document web html, vous pourrez mettre dans les en-têtes les codes suivants:

```
<meta http-equiv="content-type" content="text/html; charset=utf-8">
```

Toute fois, nous devons retenir que l'information donnée par la balise « meta » est secondaire et n'occupe pas le premier plan. C'est l'information fournie par le serveur qui sera d'abord lue, c'est-à-dire que cette information prendra le dessus, si elle existe.

A – charset=koi8-r

B – charset=koi8-u

C – charset=cp1251

D – charset=ISO-8859-1

E – charset=utf-8

Fig. 3.5. Exemples d'affichage du texte russe dans le navigateur en utilisant différents encodages (A, B, C, D et E)

FR III.5.2. Détermination par code de l'encodage d'une page web

Il existe plusieurs manières dans le langage Python de déterminer l'encodage d'une page web. Pour atteindre nos objectifs, il est necessaire d'utiliser la bibliotheque HTTP/1.1 client. Avec elle, la connexion HTTP (HTTPConnection) passe par un certain nombre « d'états », qui définissent la période à laquelle un client peut légalement faire une autre demande ou

récupérer la réponse qui a été donnée à une demande particulière. Nous utilisons egalement le module « request » qui est une bibliothèque extensible pour ouvrir des URL à l'aide de divers protocoles.

Dans ce module, c'est la fonction « urlopen » qui accepte une chaîne contenant une URL ou un objet de requête. Il fonctionne de la manière suivante en ouvrant l'URL et en renvoyant les résultats sous forme d'objet de type fichier; l'objet renvoyé a quelques méthodes supplémentaires que nous pouvons utiliser: OpenerDirector, urlopen, Request.

Ainsi, nous dirons que le module « request » permet d'envoyer une demande complète au serveur.

Avant d'obtenir l'encodage d'une page quelconque, il faudra d'abord vérifier si la dite page existe ou pas. Pour cela, nous utiliserons la fonction « HTTPResponse » qui permet d'obtenir la réponse à une demande faite au serveur. Ainsi, avec les méthodes « status », « reason » et « code », nous pourrons obtenir l'état de la page web, c'est-à-dire, dire si elle existe ou pas en utilisant les données du tableau 3.4.

Tableau 3.4 – Quelques éléments indicatifs lors de la recherche d'une page web dans un site

Le statut (status)	404	301-303, 307	200	405	400	406
Le verdict (reason)	Not Found	Moved temporarily	Ok	Not Allowed	Bad Request	Not acceptable
Le code (code)	404	301-302	200	405	400	406
Signification	Page n'existe pas	Erreurs de redirection	Page trouvée	Page non Autorisée	Mauvaise demande	Pas de réponse acceptables

Variante 1

```
1  import http.client as siteweb
2  import urllib.request
3  from chardet.universaldetector import UniversalDetector
4
5  web_link = "http://dl.papacha.ru/index.php/registratsiya1"
6
7  union = siteweb.HTTPConnection('dl.papacha.ru')
8  union.request("HEAD", web_link)
9  answer = union.getresponse()
10 union.close()
11
12 print("Le status de la page web = ",answer.status)
13 print("L' Accessibilité' = ",answer.reason)
14
15 if answer.code != 404:
16     web_link1 = urllib.request.urlopen(web_link)
17     le_detecteur = UniversalDetector()
18     for line in web_link1.readlines():
19         le_detecteur.feed(line)
20         if le_detecteur.done: break
21     le_detecteur.close()
22     web_link1.close()
23     print('Codage = ',le_detecteur.result['encoding'], 'et Langage = ',
24                               le_detecteur.result['language'])
25 else:
26     print("La page web n'existe pas à cause de: ", answer.reason)
```

Variante 2.

```
1  import urllib.request
2  import chardet
3  web_link = "http://dl.papacha.ru/index.php/registratsiya"
4  dataweb = urllib.request.urlopen(web_link).read()
5
6  chardet.detect(dataweb)
7  print('Codage = ',chardet.detect(dataweb)['encoding'], 'et
8                  Langage = ',chardet.detect(dataweb)['language'])
```

Commentaire des codes précédents
#1 Importation du module urllib.request *pour ouvrir les URL* *#2 Importation du module* chardet *pour détecter l'encodage des caractères* *#3 Détermination de l'adresse à laquelle nous allons essayer de détecter l'encodage* *#4 Obtient le résultat de la requête sur la page indiquée en forme d'objet* *#6 Lire les données optenues pour pouvoir détecter l'encodage des caractères* *#7 Retire le nom du codage et du langage dans le dictionnaire en utilisant les mots clés «encoding» et «language» pour l'afficher sur l'écran.*

Pour conclure, dans le cas où nous voulons détecter l'encodage de plusieurs fichiers séparément, il est nécessaire de réutiliser un seul *UniversalDetector* objet dans le même code. Avec la commande *detector.reset()*, nous devrons réinitialiser le détecteur au début de chaque fichier avant de faire une nouvelle vérification d'encodage avec la commande *detector.feed()* autant de fois que nous aurons de nouveaux fichiers à détecter. A chaque fois, il faudra le fermer en appelant la commande *detector.close()* et enfin retirer le résultat grâce à la commande detector.result; ce résultat sera stocké dans un dictionnaire.

Comme vous l'aviez constaté, nous avons utilisé le module de détection de caractères dans Mozilla. Il fonctionne de la manière suivante: un certain composant reçoit des octets en tant que données entrantes et se base sur les octets de données pour pouvoir "deviner" quel est le jeu de caractères des données utilisées. Ce module ne recherche pas les informations étiquetées (telles que: *<meta http-equiv="content-type" content="text/html; charset=utf-8"*>) dans la page web. Le module “chardet” fait plutôt une devinette, ce qui veut dire que, quelquefois la réponse est imprécise c’est-à-dire peut ne pas être juste. Donc vous êtes libre de l'utiliser ou alors d'apporter d'autres modifications afin de rendre le résultat fiable à 100%.

Suite à la page 278


RU Глава IV. АЛГОРИТМЫ СОРТИРОВКИ

RU Введение

Алгоритм сортировки – это способ организации данных таким образом, чтобы обеспечить коллективную организацию объектов в строго определенном порядке. Порядок может быть возрастающим, убывающим или подчиняться другим критериям. Для достижения этой цели в научном мире было разработано несколько методов. Отметим, что алгоритмы сортировки оцениваются на основе скорости выполнения (времени, затрачиваемого на достижение результата) и эффективности использования памяти (чем меньше ей требуется памяти, тем она эффективнее). Предлагаем вам некоторые из них в виде схем. Прежде чем это сделаем, попробуем перечислить известные алгоритмы в большинстве научных учебников:

- сортировка выбором (см. рис. 4.1);
- сортировка слиянием;
- сортировка Шелла;
- сортировка пузырьком (см. рис. 4.2);
- сортировка вставками (см. рис. 4.3);
- Гномья сортировка;
- сортировка с помощью двоичного дерева;
- сортировка Timsort (сортировки вставками и слиянием);
- сортировка расчёской;
- пирамидальная сортировка (сортировка кучи, англ. Heapsort);
- плавная сортировка (англ. Smoothsort);
- быстрая сортировка (англ. Quicksort);
- интроспективная сортировка (англ. Introsort);
- придурковатая сортировка (англ. Stooge sort);
- сортировка коктейлей (англ. Cocktail), или сортировка шейкеров (англ. Shaker), или двухсторонняя сортировка пузырьков – это разновидность сортировки пузырьков (см. рис. 4.4).

Полезно знать, как работают эти алгоритмы, чтобы лучше их применять. Pandas в Python – это очень богатая библиотека, которая позволяет манипулировать данными различными способами. Среди прочего, он позволяет не только сортировать определенные данные, но и дает программисту возможность выбрать тип алгоритма сортировки, который будет использоваться для повышения эффективности решения имеются задачи.

RU IV.1. Схематическое описание некоторых алгоритмов сортировки

Алгоритм сортировки выбором (или сортировкой извлечением) представляет собой алгоритм упорядочивания путем сравнения (см. рис. 4.1). Этот алгоритм считается простым и неэффективным.

Идея этого метода состоит в том, чтобы создать отсортированную последовательность, прикрепив один элемент к другому в правильном порядке. Необходимо построить конечную последовательность, начиная с левого конца массива. Алгоритм состоит из n последовательных шагов от нуля до (n-1; n – это общее количество элементов в массиве). На *i*-ом шаге, находим и выбираем наименьший из заданных элементов и меняем его с помощью элемента a[i].

Алгоритм пузырьковой сортировки

Алгоритм (см. рис. 4.2) выполняет повторяющиеся процессы в массиве a[i], который можно отсортировать (i – номер элемента массива). Для каждого прохода элементы последовательно сравниваются попарно, и если порядок пары неверен, элементы переставляются. Проходы в массиве повторяются (n-1; где n-количество элементов в массиве) раз или до тех пор, пока следующий

проход не покажет, что обмены больше не нужны, что означает, что массив в конечном итоге отсортирован. При каждом прохождении алгоритма во внутреннем цикле самый большой элемент в следующем массиве помещается на свое место в конце массива рядом с предыдущим «большим элементом», а самый маленький элемент (*min*) перемещается из одной позиции в начало массива («появляется» в нужном положении, как пузырь в воде – отсюда и название алгоритма).

Алгоритм сортировки вставками (см. рис. 4.3) является классическим алгоритмом сортировки.

Как работает алгоритм: на каждом этапе алгоритма мы выбираем один из элементов (всего *n*) из записей (массива a[*i*]) и вставляем его в нужную *i* позицию в уже отсортированном списке до тех пор, пока набор данных не будет исчерпан. Особенность этого алгоритма заключается в том, что он может работать в реальном времени, то есть он может получать список для сортировки по элементам без потери эффективности.

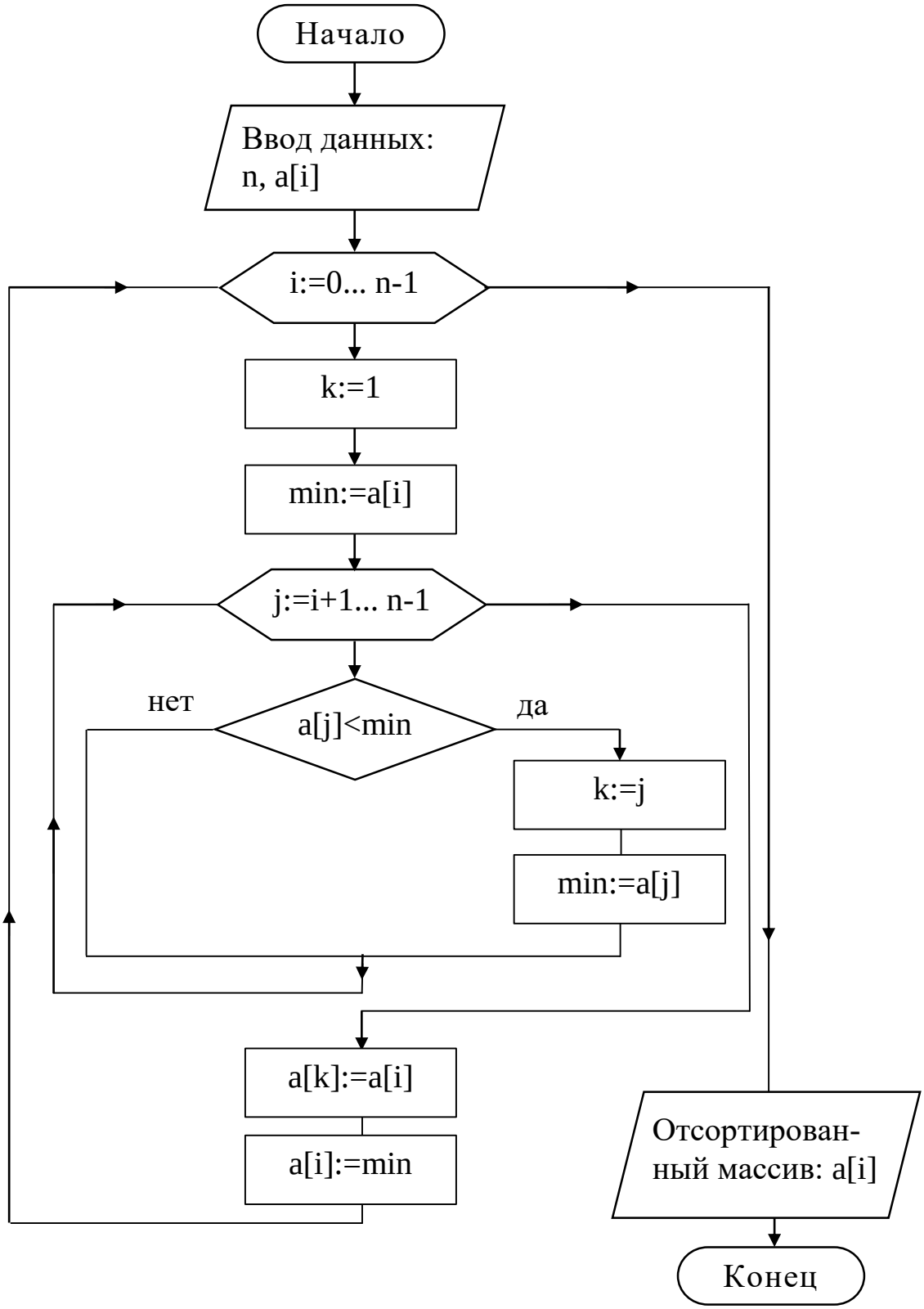


Рис. 4.1. Алгоритм сортировки выбором

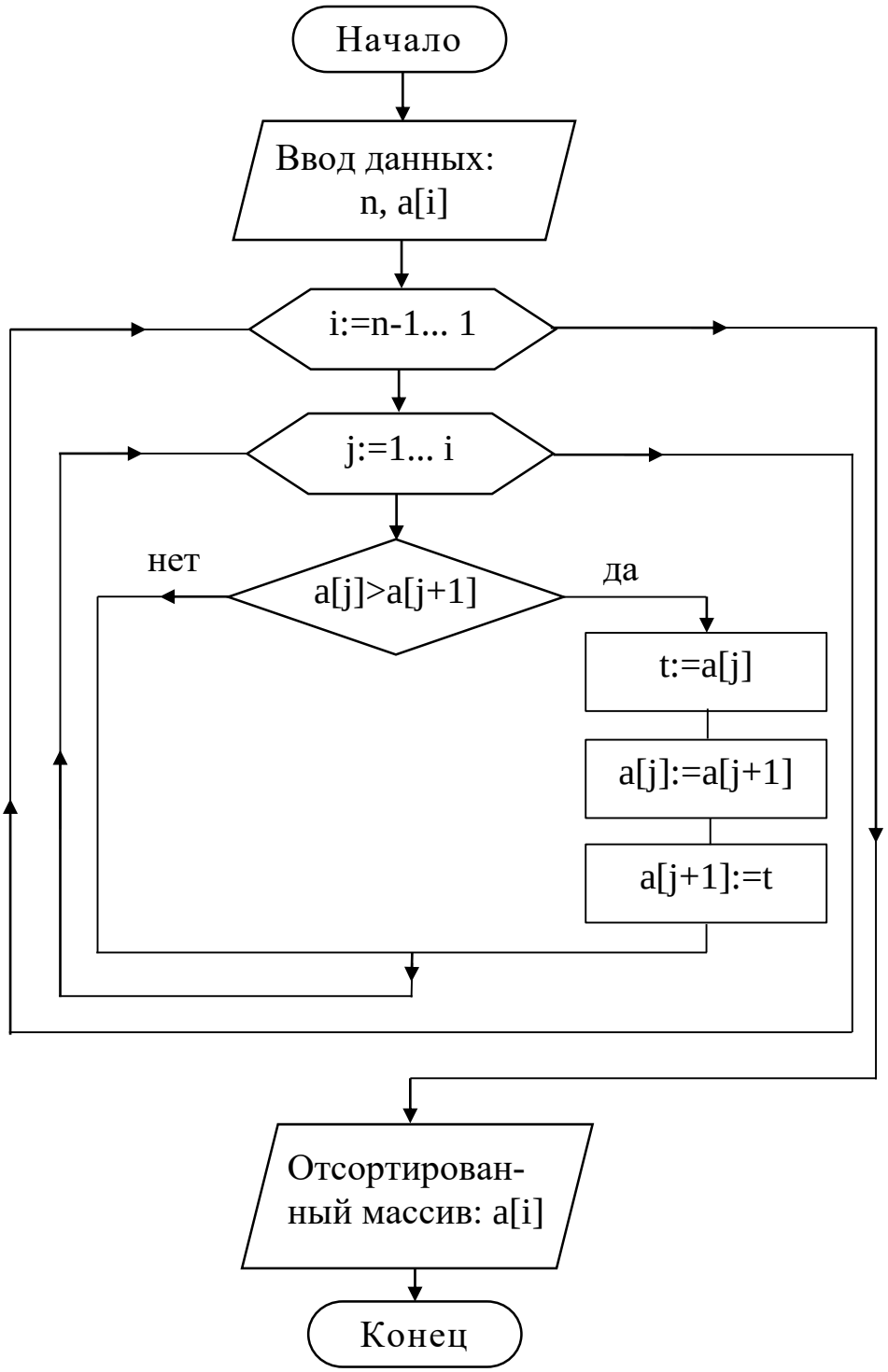


Рис. 4.2. Алгоритм пузырьковой сортировки

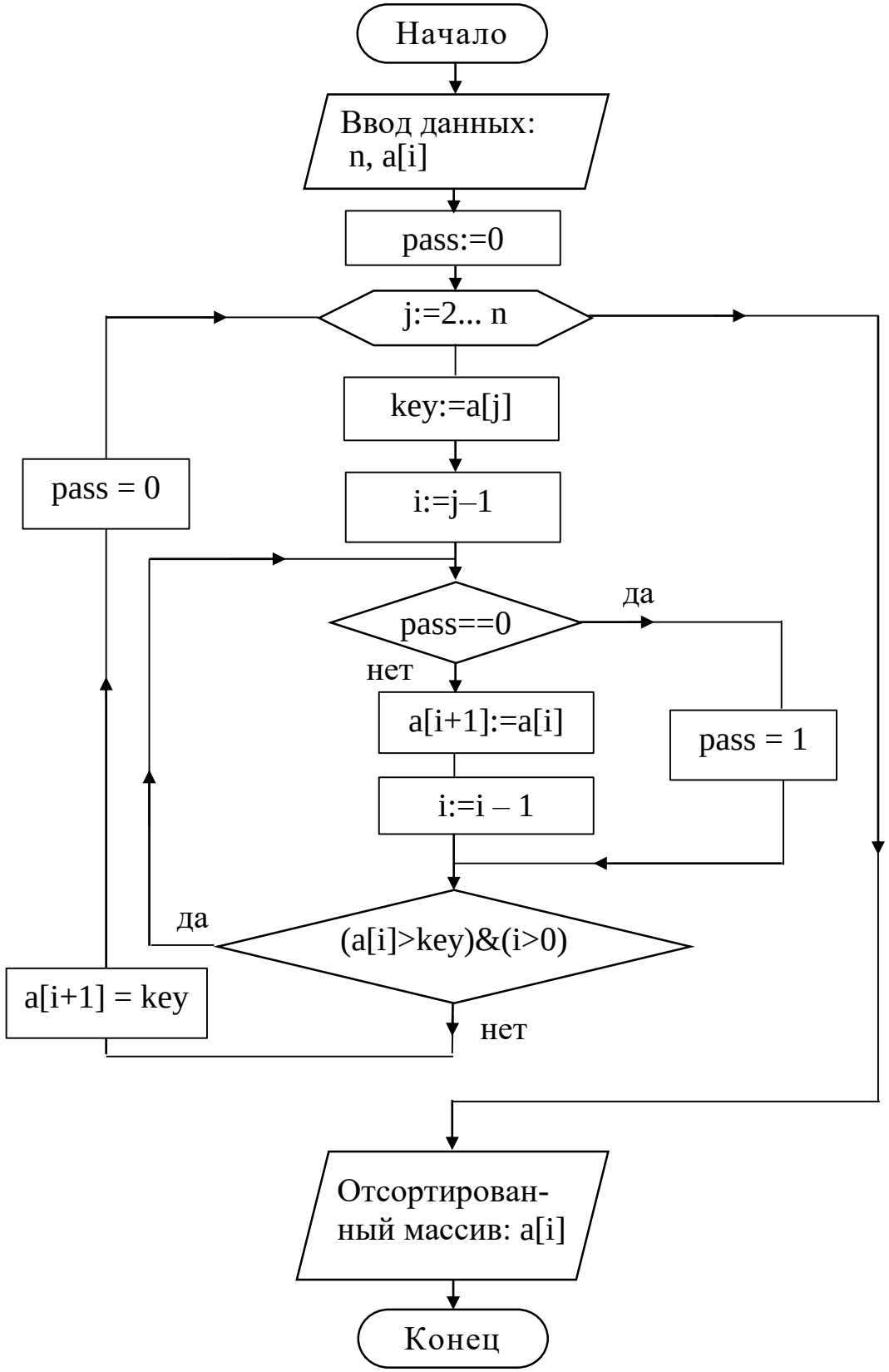


Рис. 4.3. Алгоритм сортировки вставками

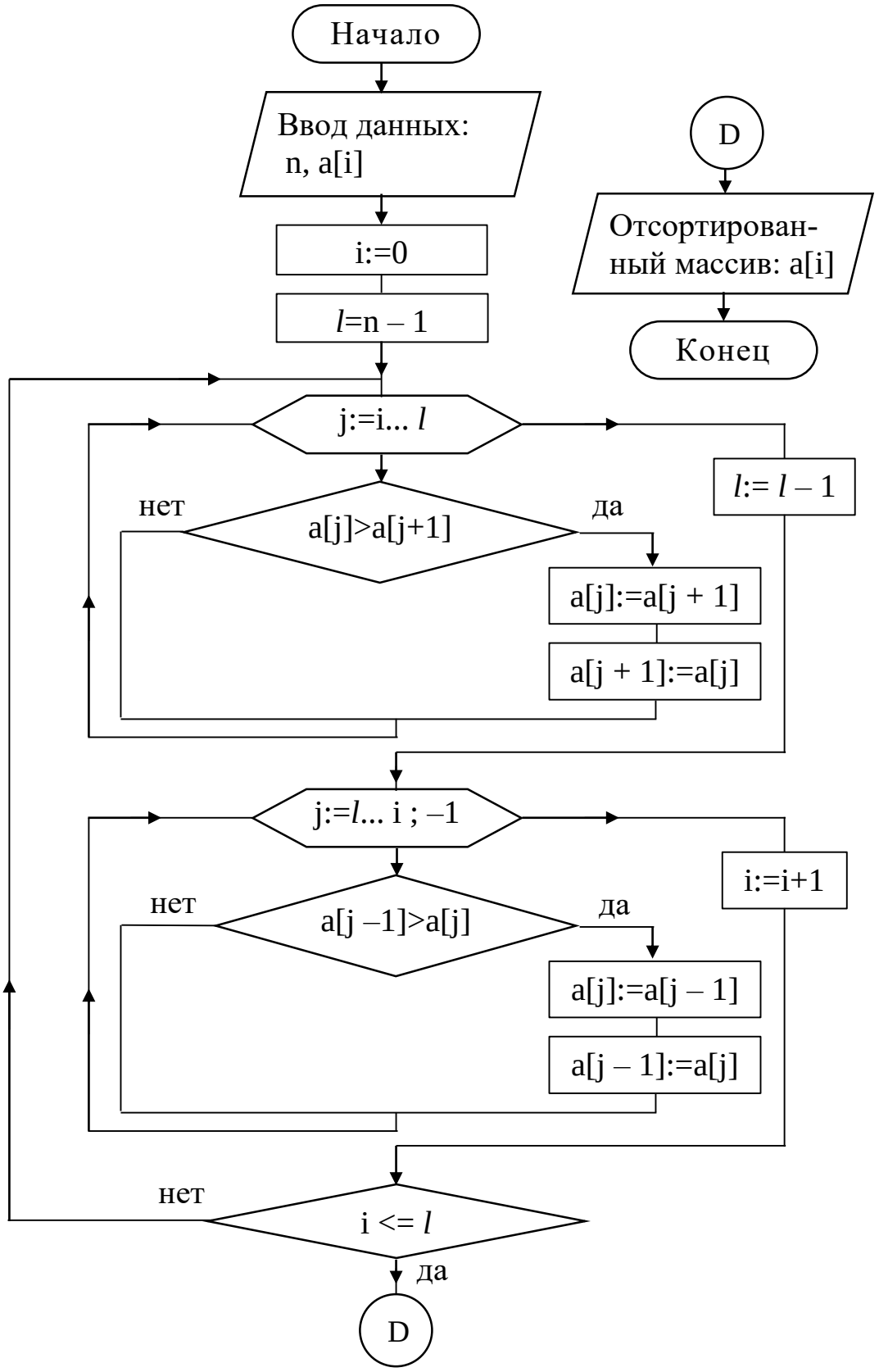


Рис. 4.4. Алгоритм сортировка коктейлей

Алгоритм сортировки коктейлей (англ. cocktail sort) или двунаправленная сортировка пузырьков (см. рис. 4.4) – это алгоритм, который является одновременно алгоритмом сортировки и сортировкой по сравнению. Разница между этим алгоритмом и пузырьковой сортировкой заключается в том, что он выполняет сортировку в каждом направлении при каждом проходе по списку, который требуется отсортировать.

RU IV.2. Сортировка по кучам или сортировка по пирамидам (на англ. heapsort)

Этот алгоритм можно рассматривать как улучшение алгоритма сортировки пузырьков. Можем сказать, что его основным недостатком является медленность в реализации. Как он работает? Каков принцип этого алгоритма?

Он заключается в получении кучи или бинарного дерева из массива и проверке свойств, о которых речь пойдет чуть ниже. Первый элемент – это корень, второй и третий являются двумя потомками первого элемента и т. д. Таким образом, первый элемент имеет в качестве дочерних элементов элементы 2×i и 2×i+1, если индексация выполняется с 1 (2×i+1 и 2×i+2, если индексация выполняется с 0).

Напомним, что двоичное (бинарное) дерево в информатике – это структура данных, представляющая собой иерархию (см. рис. 4.5 и 4.6) в форме перевернутой буквы "V" на французском или английском языке, каждый элемент которой называется узлом. Начальный узел называется корневым, а два других узла являются дочерними элементами.

Сортировка основана на следующих моментах:

а) перестроить массивную;

б) извлекать элементы из кучи в массиве один за другим;

в) на каждом поддереве (узле и 2 дочерних элементах) определите и инициализируйте самый большой элемент и сохраните его в корне кучи или в исходном узле:

если исходный (корневой) узел хранится в индексе i, то левый дочерний элемент будет вычисляться по формуле 2×i + 1, а правый дочерний элемент - по формуле 2×i + 2 (при условии, что индексация начинается с 0).

г) Если левый узел (левый дочерний элемент) больше корня, то фиксируем этот левый элемент как максимальный элемент;

д) если правый узел (правый дочерний элемент) больше, чем самый большой элемент в данный момент, тогда фиксируем этот правый элемент как максимальный элемент;

е) Если самый большой элемент не является корнем, то:

- наблюдаем перестановку элементов в массиве между самым большим текущим элементом и последним элементом в куче;
- уменьшите размер кучи на 1;

ж) рекурсивно преобразовать затронутое поддерево в кучу и начать заново с точки "b".

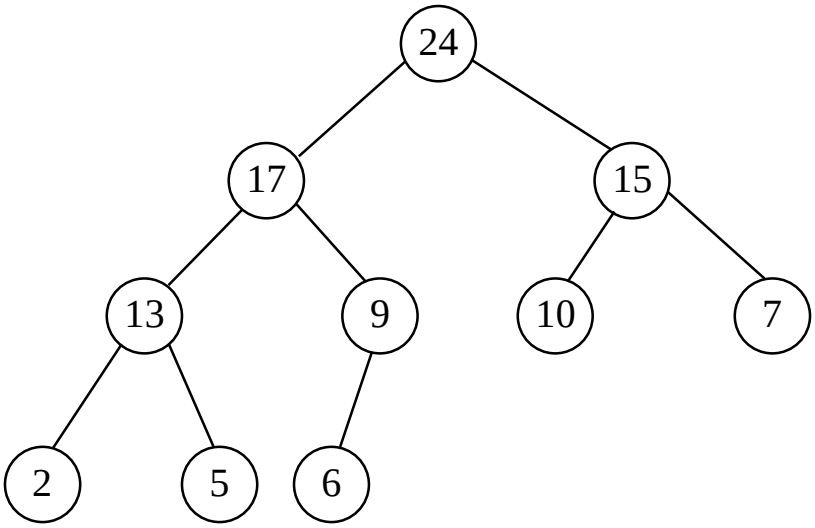


Рис. 4.5. Пример сортировки дерева в бинарном виде

24	17	15	13	9	10	7	10	2	5	6

Рис. 4.6. структура хранения данных дерева сортировки

RU IV.3. Описание алгоритма быстрой сортировки и сортировки слиянием

а) Алгоритм быстрой сортировки (англ. “QuickSort”)

Алгоритм сортировки состоит из следующих шагов:

- выберите элемент, называемый опорной точкой (эталон) в массиве. Этот элемент отсчета может оказаться в массиве. Выбор элемента может повысить эффективность алгоритма, но не обязательно определяет точность алгоритма;
- сравните все остальные элементы с эталоном и переставьте их в массиве таким образом, чтобы можно было разделить массив на три непрерывных сегмента, следующих друг за другом: сначала «элементы, меньшие эталона», после «элементы, равные эталону» и, наконец, «элементы, которые больше эталона»;
- для сегментов с меньшими и большими значениями необходимо рекурсивно выполнить одну и ту же последовательность операций и остановиться, когда длина указанных массивов будет равна единице.

б) Алгоритм сортировки по слиянию (англ. “Merge sort”)

Алгоритм сортировки состоит из соблюдения следующих шагов:

а) Сортируемая масса делится на две части примерно одинакового размера;

б) каждая из полученных частей сортируется отдельно, например – по одному и тому же алгоритму;

в) две упорядоченные массы половинного размера соединяются в одну.

PS: В пунктах, а) и б), рекурсивное деление задания на более мелкие происходит до тех пор, пока размерность массива не достигнет единицы (заметим, что любой массив, который остается с одним элементом, можно считать упорядоченным).

г) объединение двух упорядоченных массивов в один.

В итоге:

- на первом этапе вам нужно разделить список на части из одного элемента;
- на втором этапе элементы сравниваются два на два по соседству (каждый элемент сравнивается со своим соседом);
- на третьем этапе их необходимо отсортировать и объединить после сравнения;
- и в конце концов на последнем этапе все элементы сортируются и объединяются вместе.

RU IV.4. Вопросы и задачи

1) Какие типы алгоритма сортировки знаете?

2) Какие алгоритмы сортировки применяются в библиотеке Pandas Python?

3) Какой алгоритм сортировки используется по умолчанию?

4) Основные отличие между алгоритмами *QuickSort* и *Merge?*

5) Напишите программы, позволяющих сортировать одномерный массив размером n (n-это количество элементов в массиве) в порядке убывания и возрастания с использованием языка Python. Запрещено использовать стандартные встроенные функции сортировки. Используйте следующие алгоритмы для решения задачи сортировки этого массива:

- сортировка по пирамидам;
- сортировка QuickSort;
- сортировка пузырьком;
- сортировки выбором;
- сортировки вставками;
- сортировка коктейлей.

6) Напишите программы, позволяющих сортировать двухмерный массив размером n×m (*n* – количество элементов в строке и *m* – количество элементов в столбце) в порядке убывания и возрастания с использованием языка Python (Сортировка выполняется слева направо по ходу движения вниз). Запрещено использовать стандартные встроенные функции сортировки. Используйте следующие алгоритмы для решения задачи сортировки этого массива:

- сортировка по пирамидам;
- сортировка QuickSort;
- сортировка пузырьком;
- сортировки выбором;
- сортировки вставками;
- сортировка коктейлей.

7) Напишите программы, позволяющих сортировать двухмерный массив размером n×m (*n* – количество элементов в строке и *m* – количество элементов в столбце) в порядке убывания и возрастания с использованием языка Python (Сортировка выполняется справо налево по ходу движения вниз). Запрещено использовать стандартные встроенные функции сортировки. Используйте следующие алгоритмы для решения задачи сортировки этого массива:

- сортировка по пирамидам;
- сортировка QuickSort;
- сортировка пузырьком;
- сортировки выбором;
- сортировки вставками;
- сортировка коктейлей.

Продолжение на странице 291

EN Chapter IV. SORTING ALGORITHMS

EN Introduction

A sorting algorithm is a way of organizing data in such a way as to ensure the collective organization of objects in a strictly defined order. The order can be ascending, descending, or subject to other criteria. To achieve this goal, several methods have been developed in the scientific world. Note that sorting algorithms are evaluated based on the speed of execution (time spent on achieving the result) and the efficiency of memory use (the less memory it requires, the more efficient it is). We offer you some of them in the form of diagrams. Before we do this, let's try to list the known algorithms in most scientific textbooks:

- selection sorting (see Figure 4.1);
- merge sorting;
- Shell sorting (Shellsort or Shell's method);
- bubble sort (see Figure 4.2);
- sorting by inserts (see Figure 4.3);
- Gnome sort;
- tree sort;
- Timsort sorting (insertion and merge sorting);
- comb sorting;
- pyramid sorting (heap sorting, English Heapsort);
- Smooth sorting (English Smoothsort);
- quick sort (English Quicksort);
- introspective sorting (English Introsort);
- stupid sorting (English Stooge sort);
- Cocktail sorting, or shaker sorting, or double–sided bubble sorting is a kind of bubble sorting (see Figure 4.4).

It is useful to know how these algorithms work in order to apply them better. Pandas in Python is a very rich library that allows you to manipulate data in various ways. Among other things, it allows not only to sort certain data, but also gives the programmer the opportunity to choose the type of

sorting algorithm that will be used to improve the efficiency of solving a given problem.

EN IV.1. Schematic description of some sorting algorithms

Sorting algorithm by selection (Selection sort or sorting by extraction) is an algorithm for ordering by comparison (see Figure 4.1). This algorithm is considered simple and inefficient.

The idea of this method is to create a sorted sequence by attaching one element to another in the correct order. It is necessary to construct a finite sequence starting from the left end of the array. The algorithm consists of n consecutive steps from zero to (n-1; n is the total number of elements in the array). At the i^{-th} step, we find and select the smallest of the given elements and change it using the element a[i].

Bubble sorting algorithm

The algorithm (see Figure 4.2) performs repetitive processes in the array a[*i*], which can be sorted (*i* is the number of the array element). For each pass, the elements are sequentially compared in pairs, and if the order of the pair is incorrect, the elements are rearranged. The passes in the array are repeated (n-1; where n is the number of elements in the array) once or until the next pass shows that the exchanges are no longer needed, which means that the array is eventually sorted. Each time the algorithm passes through the inner loop, the largest element in the next array is placed in its place at the end of the array next to the previous «big element», and the smallest element (min) moves from one position to the beginning of the array («appears» in the desired position, like a bubble in water – hence the name of the algorithm).

The insertion sorting algorithm (see Figure 4.3) is a classical sorting algorithm.

How the algorithm works: at each stage of the algorithm, we select one of the elements (total *n*) from the records (array a[*i*]) and insert it into the desired *i* position in the already sorted list until the data set is exhausted. The peculiarity of this algorithm is that it can work in real time, that is, it can get a list to sort by elements without loss of efficiency.

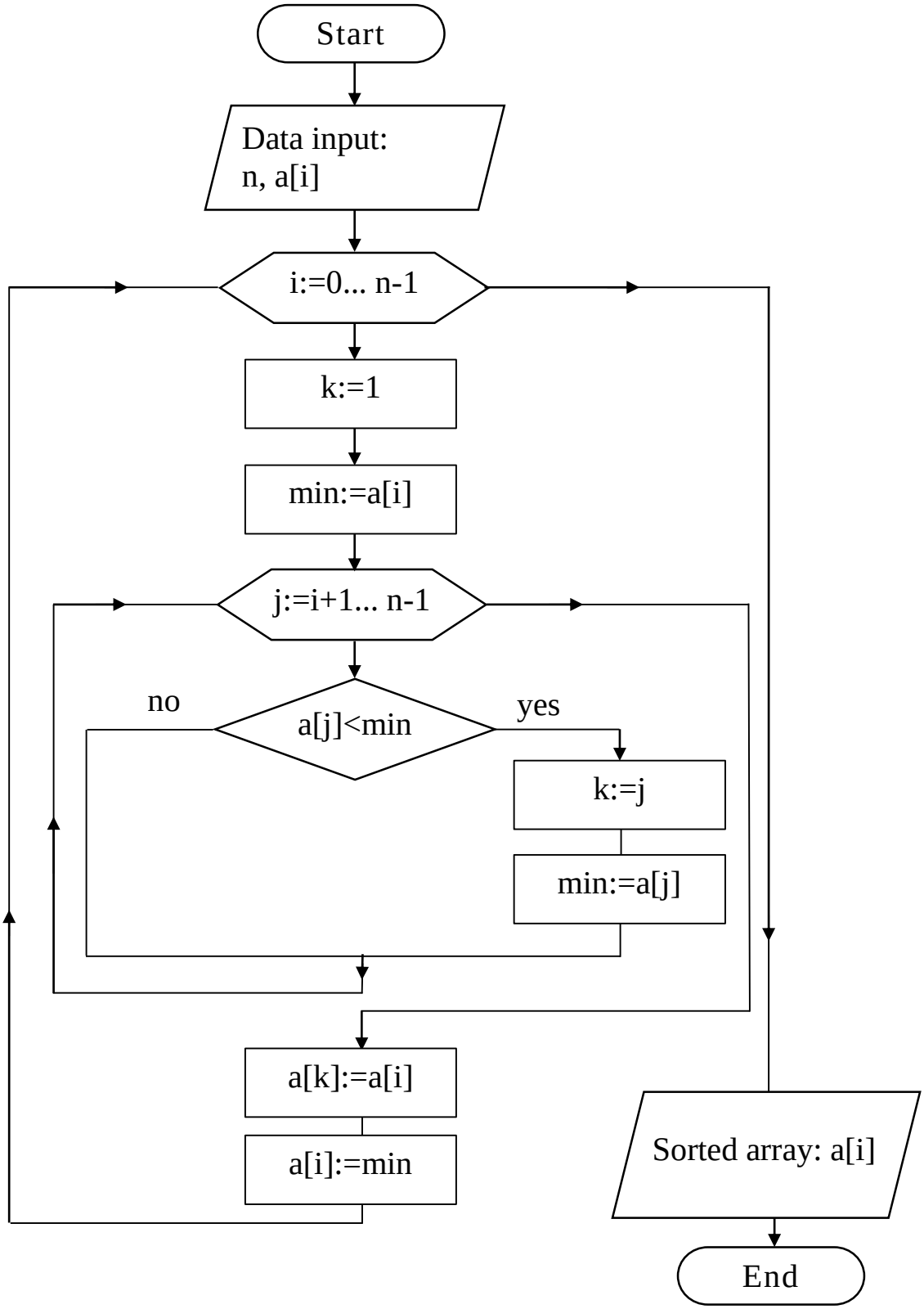


Figure 4.1. Selection sorting algorithm

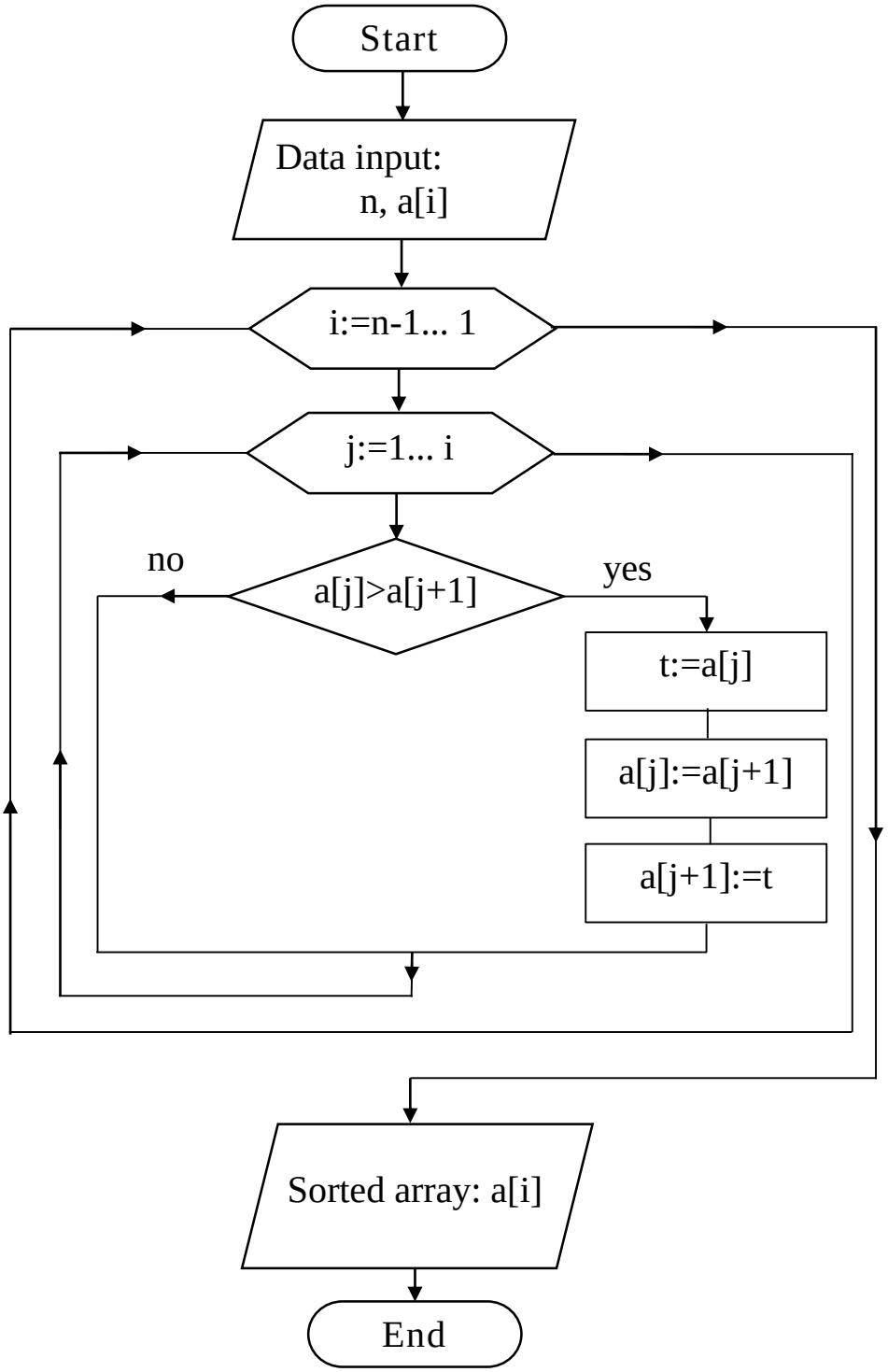


Figure 4.2. Bubble sorting algorithm

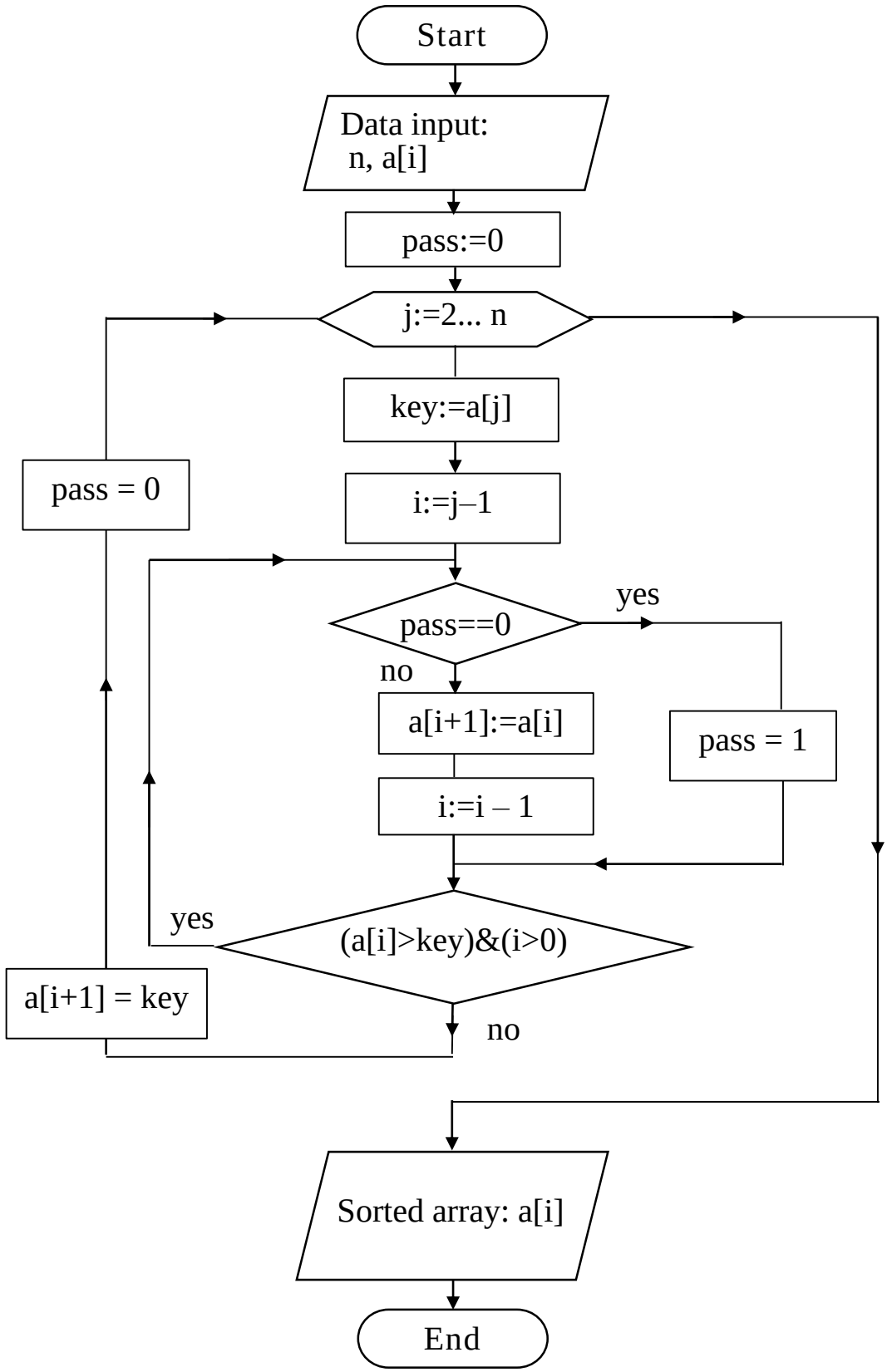


Figure 4.3. Insertion sorting algorithm

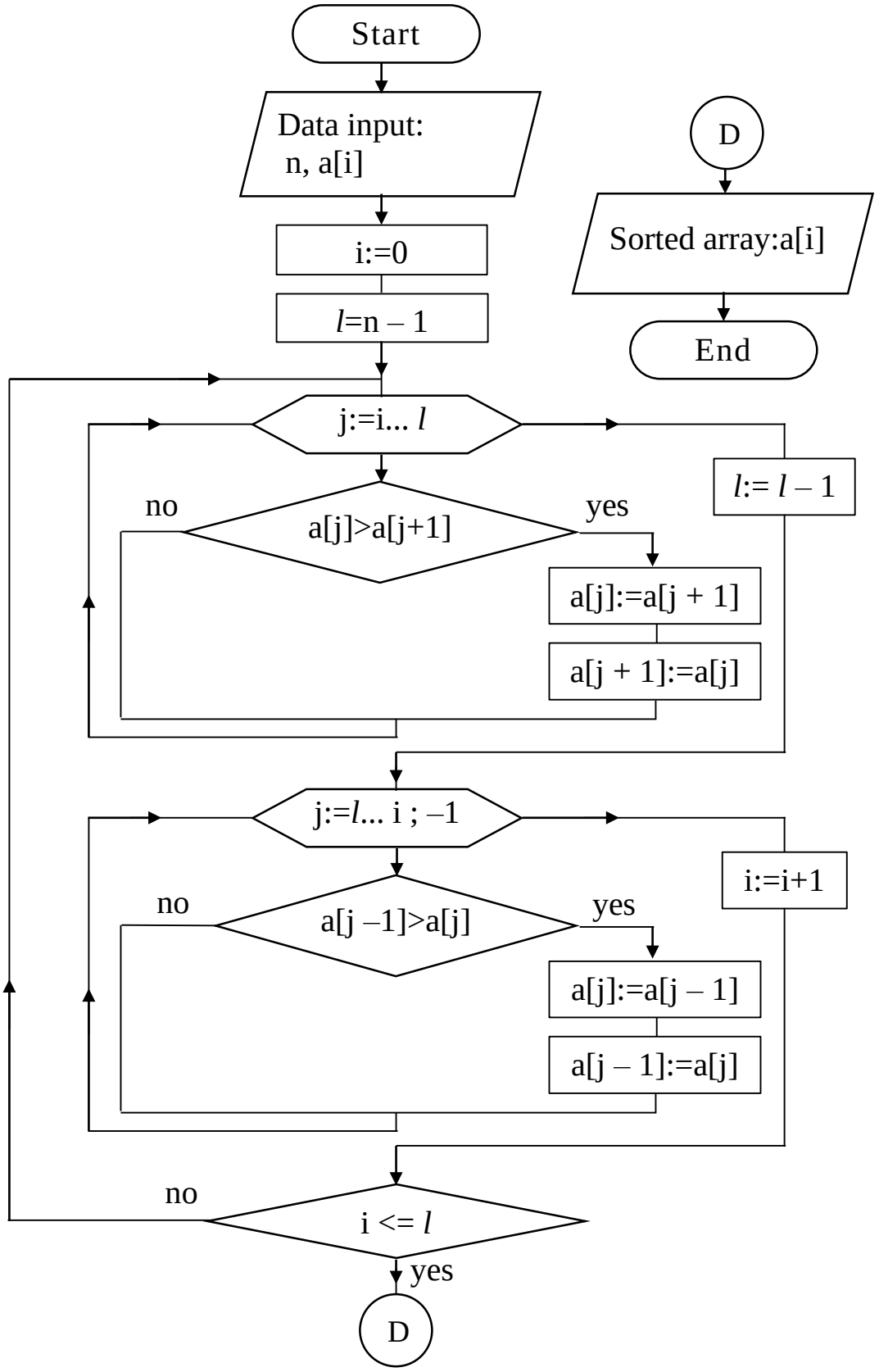


Figure 4.4. Cocktail sort algorithm

Cocktail shaker sort also known as **bidirectional bubble sort** or **cocktail sort** also known as **shaker sort** (see Figure 4.4) is an algorithm that is both a sorting algorithm and a comparison sorting. The difference between this algorithm and bubble sorting is that it performs sorting in each direction with each pass through the list that needs to be sorted.

EN IV.2. Sorting by heaps (heapsort) or sorting by pyramids

This algorithm can be considered as an improvement of the bubble sorting algorithm. We can say that its main disadvantage is slowness in implementation. How does it work? What is the principle of this algorithm?

It consists in obtaining a heap or binary tree from an array and checking the properties, which will be discussed below. The first element is the root, the second and third are the two descendants of the first element, etc. Thus, the first element has as child elements 2×i and 2×i+1 if indexing is performed from 1 (2×i+1 and 2×i+2 if indexing is performed from 0).

Recall that a binary (binary) tree in computer science is a data structure that represents a hierarchy (see Figures 4.5 and 4.6) in the form of an inverted letter "V" in French or English, each element of which is called a node. The initial node is called the root node, and the other two nodes are children.

Sorting is based on the following points:

a) rebuild a massive;

b) extract elements from the heap in the array one by one;

c) on each subtree (node and 2 child elements), define and initialize the largest element and store it in the root of the heap or in the source node:

if the source (root) node is stored in index i, then the left child element will be calculated by the formula 2×i + 1, and the right child element - by the formula is 2×i + 2 (provided that indexing starts from 0).

d) If the left node (the left child element) is larger than the root, then we fix this left element as the maximum element;

e) if the right node (the right child element) is larger than the largest element at the moment, then fix this right element as the maximum element;

f) e) If the largest element is not the root, then:

- we observe the permutation of elements in the array between the largest current element and the last element in the heap;
- reduce the heap size by 1;

g) recursively transform the affected subtree into a heap and start over from point "b".

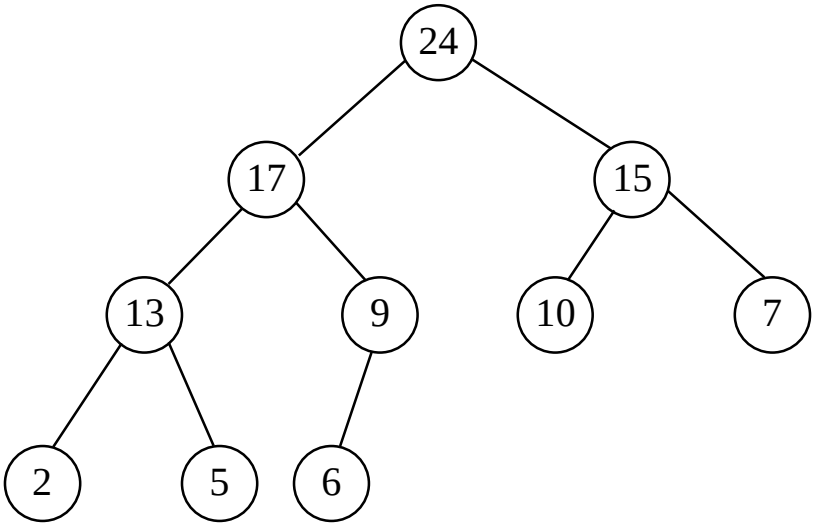


Figure 4.5. Example of sorting a tree in binary form

24	17	15	13	9	10	7	10	2	5	6

Figure 4.6. Data storage structure of the sorting tree

EN IV.3. Description of the algorithm for quick sorting and merge sorting

a) Quick sort algorithm (“QuickSort")

The sorting algorithm consists of the following steps:

- select an element called a reference point in the array. This reference element may end up in an array. Selecting an element can improve the

efficiency of the algorithm, but does not necessarily determine the accuracy of the algorithm;

- compare all the other elements with the standard and rearrange them in the array so that you can divide the array into three continuous segments following each other: first, «elements smaller than the reference», after «elements equal to the reference» and finally, «elements that are larger than the reference»;
- for segments with smaller and larger values, it is necessary to recursively perform the same sequence of operations and stop when the length of the specified arrays is equal to one.

b) Merge sorting algorithm (Merge sort)

The sorting algorithm consists of following the following steps:

a) The sorted mass is divided into two parts of approximately the same size;

b) each of the received parts is sorted separately, for example, according to the same algorithm;

c) two ordered half-sized masses are combined into one.

PS: In points a) and b), the recursive division of the task into smaller ones occurs until the dimension of the array reaches one (note that any array that remains with one element can be considered ordered).

d) combining two ordered arrays into one.

Finally:

- at the first stage, you need to divide the list into parts from one element;
- in the second stage, the elements are compared two by two in the neighborhood (each element is compared with its neighbor);
- at the third stage, they need to be sorted and combined after comparison;
- and in the end, at the last stage, all the elements are sorted and combined together.

EN IV.4. Questions and exercises

1) What types of sorting algorithm do you know?

2) What sorting algorithms are used in the Pandas Python library?

3) Which sorting algorithm is used by default?

4) What are the main differences between QuickSort and Merge algorithms?

5) Write programs that allow you to sort a one-dimensional array of size ***n*** (n is the number of elements in the array) in descending and ascending order using Python. It is forbidden to use standard built-in sorting functions. Use the following different algorithms to solve the posed exercise:

- sorting by pyramids;
- QuickSort sorting;
- bubble sorting;
- sorting by choice;
- sorting by inserts;
- sorting cocktails.

6) Write programs that allow you to sort a two–dimensional array of size n× m (n is the number of elements in a row and m is the number of elements in a column) in descending and ascending order using Python (Sorting is performed from left to right as you move down). It is forbidden to use standard built-in sorting functions. Use the following different algorithms to solve the posed exercise:

- sorting by pyramids;
- QuickSort sorting;
- bubble sorting;
- sorting by choice;
- sorting by inserts;
- sorting cocktails.

7) Write programs that allow you to sort a two–dimensional array of size n×m (n is the number of elements in a row and m is the number of elements in a column) in descending and ascending order using Python

(Sorting is done from right to left as you move down). It is forbidden to use standard built-in sorting functions. Use the following different algorithms to solve the posed exercise:

- sorting by pyramids;
- QuickSort sorting;
- bubble sorting;
- sorting by choice;
- sorting by inserts;
- sorting cocktails.

Continued on page 352

FR Chapitre IV. ALGORITHMES DE TRI

FR Introduction

Un algorithme de tri est une façon d'organiser les données de manière à permettre un arrangement collectif d'objets dans un ordre bien déterminé. L'ordre peut être croissant, décroissant ou obéir à d'autres critères. Pour atteindre ce but, plusieurs méthodes sont proposées dans le monde scientifique. Notons que les algorithmes de tri sont évalués en fonction de la vitesse d'exécution (le temps mis pour atteindre le résultat) et de l'efficacité de l'utilisation de la mémoire (moins elle a besoin d'une grande mémoire, plus il est efficace). Nous allons vous proposer certains d'entre eux en forme de schémas. Avant de le faire, essayons d'énumérer les algorithmes connus dans la plus part des manuels scientifiques:

- le tri par sélection (voir figure 4.1) ;
- le tri rapide ou tri pivot ;
- le tri fusion,ou tri dichotomique ;
- le tri par tas est un algorithme de tri par comparaisons ;
- le tri par insertion (voir figure 4.3) ;
- Introsort ou introspective (C'est une variante du tri rapide) ;
- le tri par sélection (ou tri par extraction) ;
- Timsort est un algorithme de tri hybride dérivé du tri fusion et du tri par insertion ;
- le tri de Shell ou Shell sort ;
- le tri à bulles ou tri par propagation (voir figure 4.2);
- le tri arborescent qui utilise la structure d'arbre binaire de recherche ;
- Smoothsort ;
- le tri cocktail (cocktail sort), ou tri shaker (shaker sort) ou tri à bulles bidirectionnel (bidirectional bubble sort) est une variante du tri à bulles (voir figure 4.4) ;
- le comb sort ou tri à peigne ou tri de Dobosiewicz ;
- le tri pair-impair par transposition.

Il est utile de savoir comment fonctionne ces algorithmes pour mieux les mettre en application. Pandas dans Python est une bibliothèque très riche qui permet de manipuler les données de différentes manières. Entre autres, elle permet non pas seulement de faire des tris de certaines données, mais aussi de donner la possibilité au programmiste de choisir le type d'algorithme à utiliser afin d'augmenter l'efficacité d'une résolution d'un exercice quelconque.

FR IV.1. Description schématique de certains algorithmes tri

L'algorithme tri par sélection (ou tri par extraction) est un algorithme de tri par comparaison (voir figure 4.1). Cet algorithme est considéré comme simple et inefficace.

L'idée de cette méthode est de créer une séquence triée en y attachant un élément par un autre dans le bon ordre. Il est nécessaire de construire une séquence finie, en commençant à l'extrémité gauche du tableau. L'algorithme se compose de n étapes consécutives allant de zéro à (n-1 ; n – étant le nombre total des éléments dans le tableau). À l'étape i, nous trouvons et sélectionnons le plus petit des éléments donnés et le changeons avec l'élément a[i].

Algorithme tri à bulles ou tri par propagation

L'algorithme (voir figure 4.2) procède en des processus répétés sur un tableau qui peut être trié. Pour chaque passage, les éléments sont comparés successivement par paires et, si l'ordre de la paire est incorrect, les éléments sont réarrangés. Les passages sur le tableau sont répétés (n-1 ; où ***n*** est le nombre d'éléments dans le tableau) fois ou jusqu'à ce que le passage suivant révèle que les échanges ne sont plus nécessaires, ce qui veut dire ou signifie que le tableau est finalement trié. À chaque passage de l'algorithme dans la boucle interne, le plus grand élément du tableau suivant est placé à sa place à la fin du tableau à côté du «plus grand élément» précédent, et le plus petit élément se déplace d'une position au début du tableau («apparaît» à la

position souhaitée, comme une bulle dans l'eau — d'où le nom de l'algorithme).

L'algorithme tri par insertion (voir figure 4.3) est un algorithme de tri classique.

Fonctionnement de l'algorithme: à chaque étape de l'algorithme, nous sélectionnons l'un des éléments parmi les entrées et l'insérons à la position

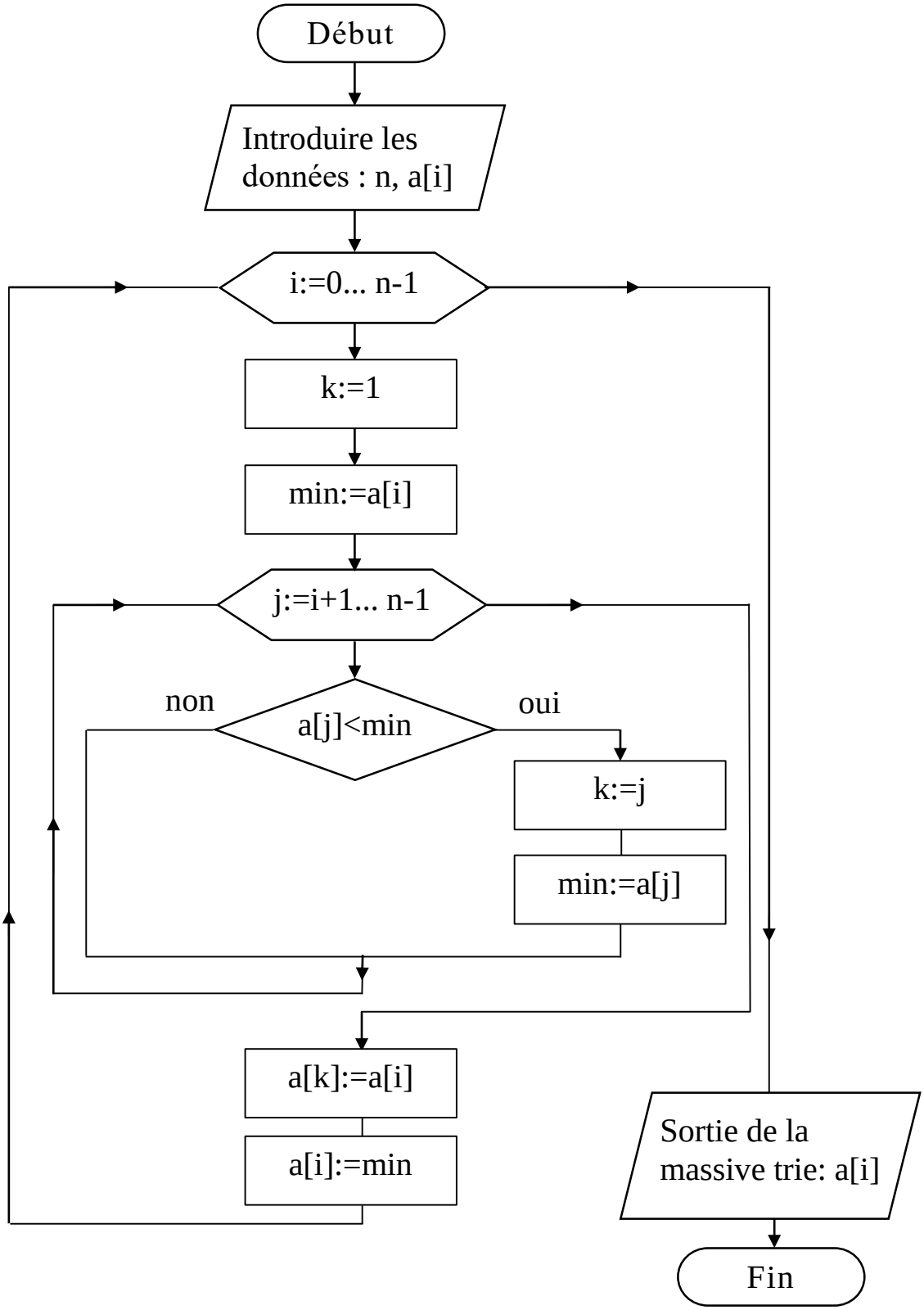


Fig. 4.1. Algorithme tri par sélection

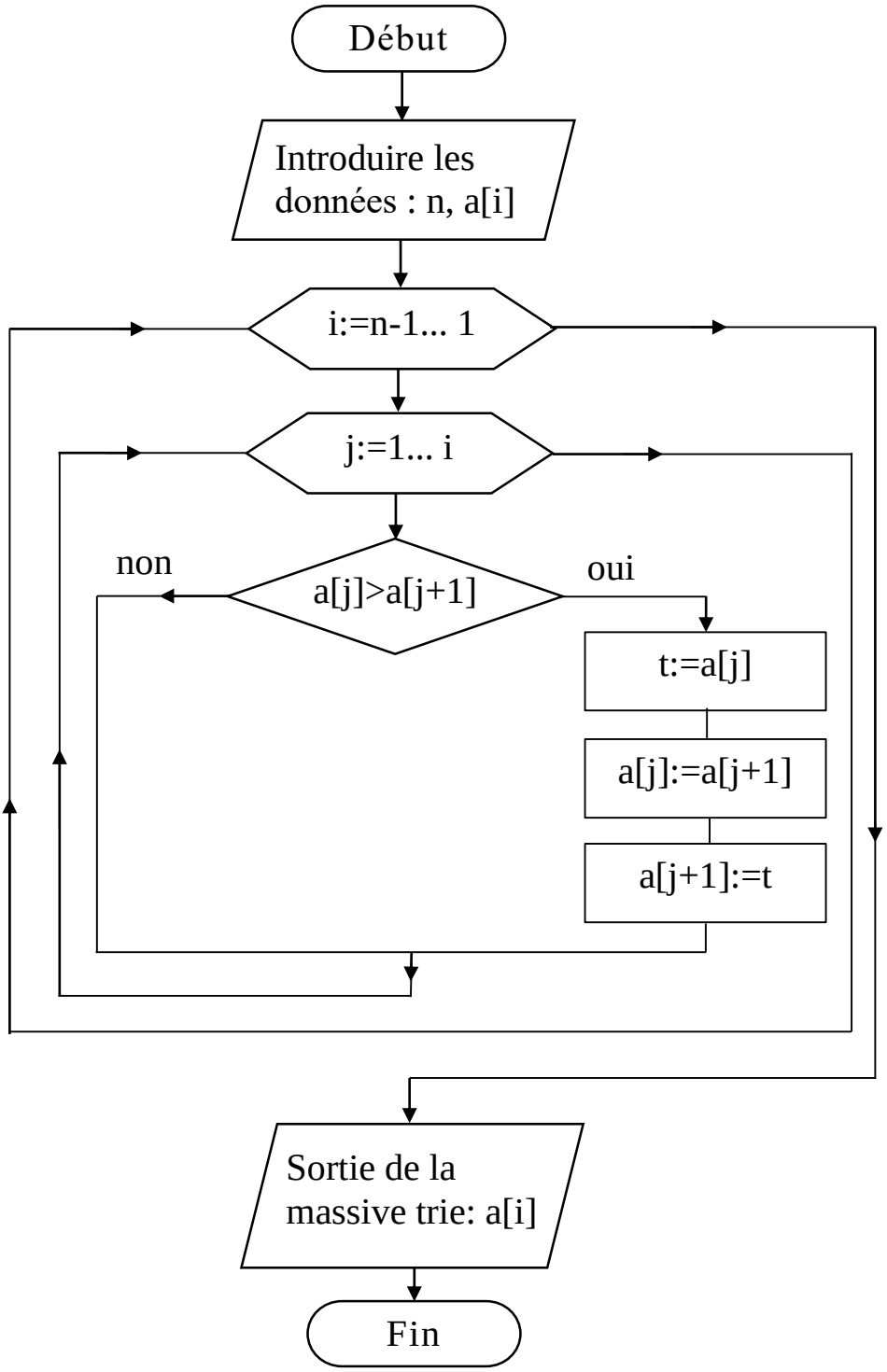


Fig. 4.2. Algorithme tri à bulles

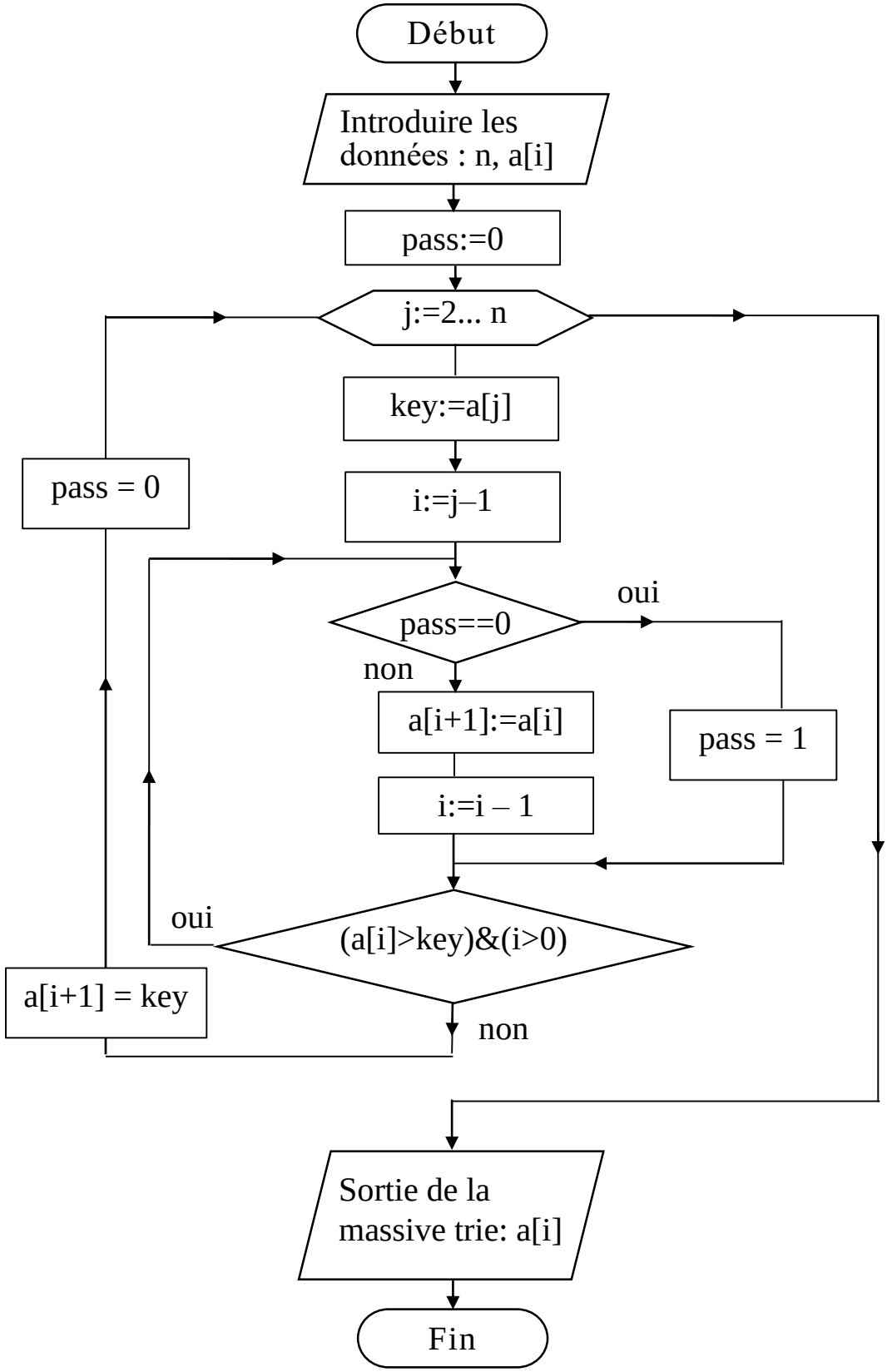


Fig. 4.3. Algorithme tri par insertion

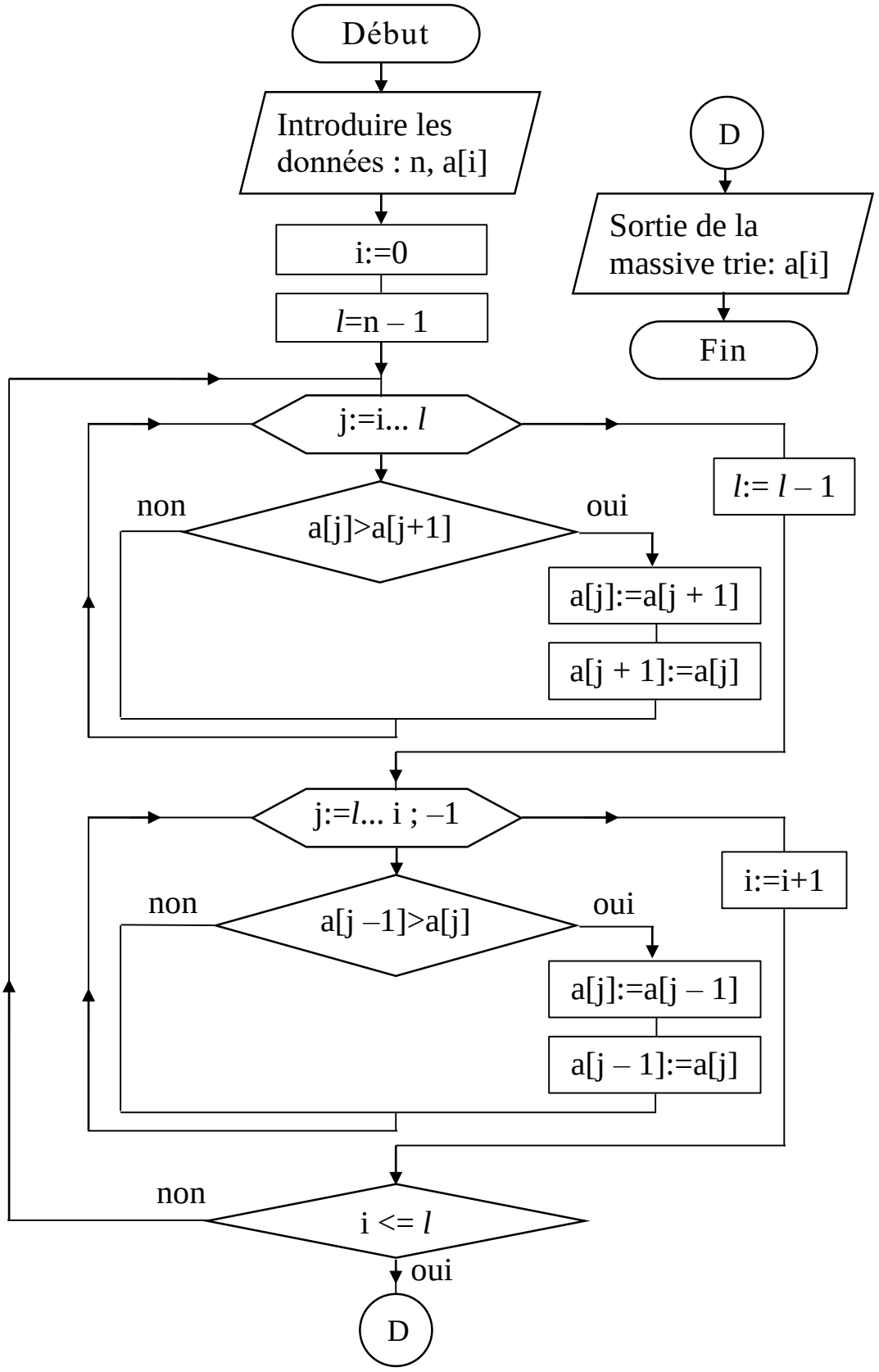


Fig. 4.4. Algorithme tri cocktail

souhaitée dans la liste déjà triée, jusqu'à ce que l'ensemble de données soit épuisé. Cet algorithme a la particularité de pouvoir fonctionner en direct, c'est-à-dire qu'il peut recevoir la liste à trier, élément par élément sans perdre en efficacité.

Algorithme tri cocktail (cocktail sort), ou **tri shaker** (**shaker sort**) ou **tri à bulles bidirectionnel** (voir figure 4.4) est un algorithme qui est à la fois un algorithme de tri et un tri par comparaison. La différence entre cet algorithme et le tri à bulles est qu'il exécute un tri dans chaque direction à chaque passe le long de la liste à trier.

FR IV.2 Le tri par tas ou tri Pyramidal (en anglais : heapsort)

Ce algorithme peut être considéré comme l'amélioration de l'algorithme tri par bulle. Nous pouvons dire que son inconvénient majeur est sa lenteur dans la réalisation. Comment fonctionne t-il? Quel est le principe de cet algorithme?

Il consiste à obtenir un tas ou un arbre binaire venant du tableau et vérifiant les propriétés qui sont relatées un peu plus bas. Le premier élément est la racine, le deuxième et le troisième sont les deux descendants du premier élément, etc. Ainsi le ***i***-ième élément a pour fils les éléments $2\times i$ et $2\times i+1$ si l'indexation se fait à partir de 1 ($2\times i+1$ et $2\times i+2$ si l'indexation se fait à partir de 0).

Rappelons qu'un arbre binaire en informatique est une structure de données qui se représente sous la forme d'une hiérarchie (voir schéma 4.5 et 4.6) en forme de lettre renversée "V" français ou anglais dont chaque élément est appelé nœud. Le nœud initial est appelé racine et le deux autres nœuds sont des éléments fils.

Le tri est fondé sur les points suivants:

a) réarranger la massive ;

b) extraire les éléments du tas dans la massive un par un ;

c) sur chaque sous-arbre (nœud et les 2 queues ou éléments fils), déterminer et initialiser le plus grand élément et le stocker à la racine du tas ou au nœud initial:

si le nœud initial (racine) est stocké à l'index ***i***, l'élément fils de gauche sera calculé par la formule $2 \times i + 1$ et l'élément fils de droit par la formule $2 \times i + 2$ (en supposant que l'indexation commence à 0).

d) Si le nœud de gauche (élément fils de gauche) est supérieur à la racine, alors nous fixons cet élément de gauche comme élément maximal;

e) Si le nœud de droite (élément fils de droite) est plus grand que le plus grand élément actuellement, alors nous fixons cet élément de droite comme élément maximal;

f) Si le plus grand élément n'est pas une racine, alors :

- nous assistons à une permutation de place des éléments dans la massive, entre le plus grand élément actuel et le dernier élément du tas;
- réduire la taille du tas de 1

g) convertir récursivement en tas le sous-arbre affecté et recommencer à partir du point «b».

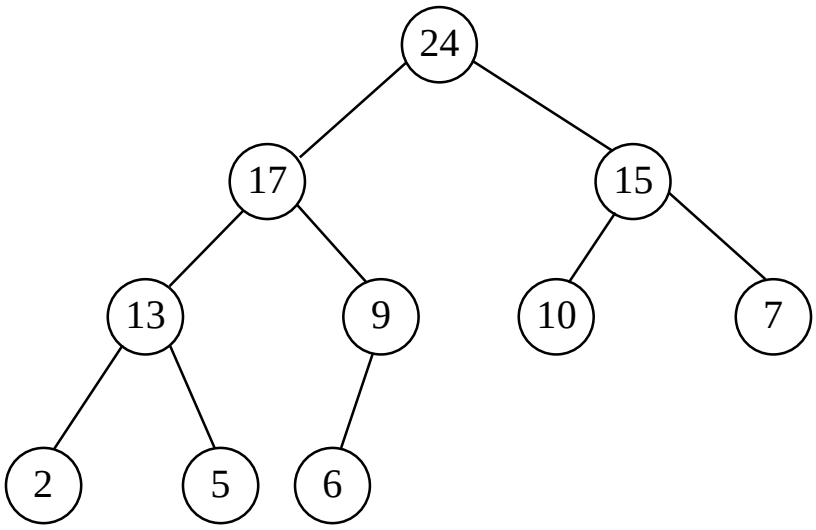


Fig. 4.5. Exemple d'arbre de tri en forme binaire

24	17	15	13	9	10	7	10	2	5	6

Fig. 4.6. Structure de stockage des données de l'arbre de tri

FR IV.3. Description de l'algorithme tri rapide et tri par fusion

a) Algorithme tri rapide (en anglais "QuickSort")

L'algorithme du tri consiste à respecter les étapes suivantes:

- sélectionner un élément appelé référence (pivot) dans la massive. Ce pivot peut se retrouver dans la massive. Le choix de l'élément peut améliorer l'efficacité de l'algorithme, mais ne détermine pas obligatoirement l'exactitude de l'algorithme ;
- comparer tous les autres éléments avec la référence et les réorganiser dans la massive de manière à pouvoir pouvoir diviser la massive en trois segments continus qui se succèdent: d'abord «les éléments qui sont plus petits que la référence», après «les éléments qui sont égales à la référence» et enfin «les éléments qui sont plus grands que la référence» ;
- pour les segments qui ont les valeurs plus petites et plus grandes, il faut effectuer récursivement la même séquence d'opérations et s'arrêter lorsque la longueur des dites massives sera égale à un.

b) Algorithme tri par fusion (en anglais "Merge sort")

L'algorithme du tri consiste à respecter les étapes suivantes:

a) La massive triable est divisée en deux parties à peu près de la même taille;

b) chacune des parties résultantes est triée séparément, par exemple – par le même algorithme;

c) deux massives ordonnées de demi-taille se connectent en une seule.

NB: Dans le point a) et b), la division récursive de l'exercice en plus petites se produit jusqu'à ce que la dimension de la massive atteigne un (notons que toute massive qui reste avec un élément peut être considérée comme ordonnée).

d) Joindre deux massives ordonnées en une seule.

En résumé:

- dans la première phase, il faut diviser la liste en morceaux d'un élément;
- dans la deuxième phase, les éléments sont comparés deux à deux par voisinage (chaque élément est comparé avec son voisin);
- dans la troisième phase, il faut les trier et combiner après comparaison;
- et en fin dans la dernière étape, tous les éléments sont triés et fusionnés ensemble.

FR IV.4. Questions et exercices

1) Quels sont les types d'algorithme tri que vous connaissez?

2) Quels types algorithmes tris sont utilisés et appliqués dans la bibliothèque Pandas de Python?

3) Quel algorithme tri par défaut est utilisé dans Pandas?

4) Quelle est la différence entre les algorithmes QuickSort et Merge?

5) Écrire des programmes qui permettent de faire le tri d'un tableau unidimensionnel de taille *n* (*n* est le nombre d'éléments dans le tableau) par ordre décroissant et croissant en utilisant le langage Python. Il est interdit d'utiliser les fonctions intégrées de tri standard. Utiliser les algorithmes suivants pour résoudre le tri de ce tableau:

- trier par pyramide ;
- tri QuickSort ;
- tri par bulle ;
- sélection de tri ;
- tri par insertions ;
- tri des cocktails.

6) Écrire des programmes qui permettent de faire le tri d'un tableau bidimensionnel de taille n×m (n est le nombre d'éléments dans une ligne et m est le nombre d'éléments dans une colonne) par ordre décroissant et croissant en utilisant le langage Python (Le tri se fait de la gauche vers la droite en descendant). Il est interdit d'utiliser les fonctions standard intégrées de tri. Utiliser les algorithmes suivants pour résoudre le tri de ce tableau:

- trier par pyramide ;
- tri QuickSort ;
- tri par bulle ;
- sélection de tri ;
- tri par insertions ;
- tri des cocktails.

7) Écrire des programmes qui permettent de faire le tri d'un tableau bidimensionnel de taille *n*×*m* (*n* est le nombre d'éléments dans une ligne et *m* est le nombre d'éléments dans une colonne) par ordre décroissant et

croissant en utilisant le langage Python (Le tri se fait de la droite vers la gauche en descendant). Il est interdit d'utiliser les fonctions standard intégrées de tri. Utiliser les algorithmes suivants pour résoudre le tri de ce tableau:

- trier par pyramide ;
- tri QuickSort ;
- tri par bulle ;
- sélection de tri ;
- tri par insertions ;
- tri des cocktails.

Suite à la page 407


RU Глава V. PANDAS – ИНСТРУМЕНТ АНАЛИЗА ДАННЫХ

Эта библиотека позволяет интегрировать функциональность пакетов NumPy и matplotlib, предоставляя пользователю удобный инструмент для анализа и визуализации различных данных. Таблицы данных, которыми манипулируют Pandas, называются DataFrame.

Pandas – это библиотека python, которая позволяет выполнять следующие роли:

- позволяет легко манипулировать анализируемыми данными;
- позволяет манипулировать таблицами данных с метками на столбцах и строках; легко манипулирует неупорядоченными данными в упорядоченной форме (обратите внимание, что столбцы могут вставляться и удаляться из структур данных во время обработки);
- позволяет строить графики из библиотеки *m* «matplotlib»;
- также используется для обработки массивов размером 1, 2 или n со встроенной индексацией;
- предоставляет широкий выбор источников данных: текстовые файлы, файлы типа Excel, файлы с разделенными запятыми (CSV), файлы в формате HDF5 (Hierarchical Data Forma – Иерархическая форма данных), базы данных SQL и другие.

Библиотека требует установки в вашей среде Python с помощью команды: « pip install pandas ». Если у вас возникли проблемы во время установки, рекомендуется ознакомиться со следующей веб-страницей для получения дополнительной информации:

URL : https://pandas.pydata.org/docs/getting_started/install.html

Структура данных в Pandas представлена в 2 разных формах:

- как «Series», то есть в виде строки (одномерный массив (вектор строки или столбца));

- под двумерным массивом, т. е. в виде матрицы (строк и столбцов);

RU V.1. Pandas в «Series»

Чтобы использовать эту форму представления данных «Series», необходимо знать, что ее синтаксис выглядит следующим образом:

*Pandas.Series(**data**=None, **index**=None, **dtype**: Dtype | None=None, **name**=None, **copy**: bool=False, **fastpath**: bool=False)*

где ***data*** – являются последовательными данными в виде списка;

index – это метка, используемая для нумерации каждого элемента или значения в массиве. Количество индексов равно количеству элементов, расположенных в массиве, поэтому они должны иметь одинаковую длину с обеих сторон. Если значения Индекса не указаны, то по умолчанию они будут принимать значения от 0 до *n* соответственно. (n-1 – это общее количество введенных значений);

dtype – тип данных, требуемый: int, int8, int64, float, float64, object (или str), bool, complex, etc.;

name – Имя, тип строки(str); это название для «Series» необязателен;

copy (логический тип, по умолчанию нет) – позволяет копировать входные данные. Для данных типа “словарь” – «dictionnaire» значение по умолчанию «None» ведет себя как «copy=True».

Пример 1:

Предположим, у нас был лист бумаги, на котором мы в течение недели отмечали значения температуры в градусах Цельсия (C) во Владимире в определенное время. Давайте обозначим числами 0–6 разные дни недели, начиная с воскресенья. Полученные температуры различны и приведены в следующей таблице 5.1:

Таблица 5.1 – Данные о температурах в градусах Цельсия

0	1	2	3	4	5	6
10C	5C	8C	14C	2C	9C	7C

Эти данные могут быть введены в компьютер с помощью функций Pandas через «Series».

Числа от 0 до 6 будут считаться индексами, а температуры – одномерными массивами (Series). Итак, вот как будет выглядеть код из ***списка***:

```
1  import pandas as pa
2
3  d_s = ["10C", "5C", "8C", "14C", "2C", "9C", "7C"]
4  data_s = pa.Series(data=d_s)
5  print(data_s)
```

Результат после выполнения кода

```
0    10C
1     5C
2     8C
3    14C
4     2C
5     9C
6     7C
dtype: object
```

Мы наблюдаем автоматическое появление номеров в каждой строке от 0 до 6, которые представляют номера строк и одновременно разные дни недели. Эти числа мы можем заменить их первыми двумя буквами дней недели; то есть Во →0; По → 1; Вт→2; Ср→3; Че→4; Пя→5 и Су→6. Давайте посмотрим, как изменить эти

числа по префиксам дней недели с помощью параметра «index». Для этого необходимо внести следующие изменения в код:

```
1  import pandas as pa
2
3  d_s = ["10C", "5C", "8C", "14C", "2C", "9C", "7C"]
4  ind = ["Во", "По", "Вт", "Ср", "Че", "Пя", "Су"]
5  data_s = pa.Series(data=d_s, index=ind)
   print(data_s)
```

Результат после выполнения кода

```
Во     10C
По      5C
Вт      8C
Ср     14C
Че      2C
Пя      9C
Су      7C
dtype: object
```

Попробуем найти среднее значение температуры за неделю с помощью этих данных. Для этого необходимо будет использовать функцию «mean».

Проблема заключается в том, что значения должны быть не типа «*объект*», а типа «*int*», например. Перед преобразованием значений с помощью функции «*astype*», необходимо сначала удалить символ «C» из данных с помощью функции «*replace*». Вот как будет представлен код:

```
1  import pandas as pa
2
3  d_s = ["10C", "5C", "8C", "14C", "2C", "9C", "7C"]
4  ind = ["Во", "По", "Вт", "Ср", "Че", "Пя", "Су"]
5  data_s = pa.Series(data=d_s, index=ind)
6  trans = data_s.str.replace("C","").astype(int).mean()
7  print("Данные:\n %s"%data_s)
8  print("Средняя температура = %.3f C"%trans)
```

Результат после выполнения кода

```
Данные:
Во     10C
По      5C
Вт      8C
Ср     14C
Че      2C
Пя      9C
Су      7C
dtype: object
Средняя температура = 7.857 C
```

Пример 2:

Давайте попробуем изменить коды с теми же последовательными («Series») данными из «словаря» – «***dictionary***» с указанным индексом (дни недели), который мы заявили в предыдущем примере.

```
1  import pandas as pa
2
3  d_s2 = {"Во": "10С", "По":"5С", "Вт":"8С", "Ср":"14С",
4                    "Чт":"2С", "Пя":"9С", "Су":"7С"}
5  data_s2 = pa.Series(data=d_s2)
6  print("Данные:\n %s"%data_s2)
```

Результат после выполнения кода

```
Данные:
Во      10С
По       5С
Вт       8С
Ср     14С
Че       2С
Пя       9С
Су       7С
dtype: object
```

Мы видим, что ключи словаря соответствуют значениям индекса данных, поэтому, если добавим новые значения индекса с помощью параметра «index», то эффекта не будет.

Пример 3:

Давайте снова изменим коды с теми же последовательными («Series») данными первого примера из одномерного массивного типа «***ndarray***» с четко указанным индексом (дни недели). Нам нужно использовать библиотеку «***NumPy***», то есть импортировать ее.

```
1  import pandas as pa
2  import numpy as np
3
4  d_s3 = np.array(["10C", "5C", "8C", "14C", "2C", "9C", "7C"])
5  ind = ["Во", "По", "Вт", "Ср", "Че", "Пя", "Су"]
6  data_s3 = pa.Series(data=d_s3, index=ind)
7  print("Данные:\n %s"%data_s3)
```

Результат после выполнения кода

```
Данные:
Во     10C
По      5C
Вт      8C
Ср     14C
Че      2C
Пя      9C
Су      7C
dtype: object
```

С помощью «Series» можем выполнять множество манипуляций с данными. Давайте прорешаем несколько примеров с определенными атрибутами:

С помощью атрибута «axes» мы можем обращаться к метке индексов и делать с ними то, что хотим, например, менять тип с помощью «astype» или при необходимости поставить «float».

```
1  ex1 = data_s3.axes[0].astype("float")
2  # или
3  ex1 = data_s3.axes.astype("float")
```

С помощью атрибута «size» можно определить количество элементов в массиве, а с помощью атрибута «ndim» - тип массива по его размерности.

```
1  ex1 = data_s3.shape
2  # и
3  ex1 = data_s3.ndim
```

С помощью атрибута «values» можно выводить данные без какой-либо дополнительной информации.

```
1  ex1 = data_s3.values
```

Остальные атрибуты (см. таблицу 5.2).

Таблица 5.2 – Некоторые полезные атрибуты типа «Series» в Pandas

T	Позволяет вернуть транспонирование массива, который здесь по определению равен самому себе, потому что имеем дело с массивом одного измерения
array	Для расширения массивов данных (например, с помощью *NumPy*)
at	Использовать уникальное значение массива, используя метку
dtype	Возвращает используемый тип данных в массиве: int, int8, int32, int64, float, float64, object (или str), bool, complex или другие.
dtypes	Возвращает используемый тип данных в массиве: int, int8, int32, int64, float, float64, object (или str), bool, complex или другие.
hasnans	Возвращает True, если есть хотя бы одна ячейка со значением «NaN»
iat	Получите или установите значение ячейки, просто указав номер строки или столбца

Окончание табл. 5.2

iloc	Индексация исключительно на основе расположения значений (целых чисел) для выборки по позиции. Он позволяет выбрать значение, принадлежащее определенной строке или столбцу в наборе данных
is_monotonic	Возвращает значение True, если количество данных в массиве увеличивается
is_monotonic_decreasing	Возвращает значение True, если данные в массиве уменьшаются
is_monotonic_increasing	Та же функция, что и « *is_monotonic »*
is_unique	Возвращает значение True, если в массиве есть только одно значение
loc	Позволяет получить доступ к группе строк и столбцов по меткам или логическому значению таблицы
name	Возвращает имя «Series»
nbytes	Возвращает количество байтов данных, используемых в массиве

Итак, какие возможности мы можем использовать с типом «Series»? Pandas дают нам несколько методов или функций для манипулирования данными в одномерном массиве. Полный список этих функций можно получить по следующей ссылке URL: [https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.Series.html], среди этих функций выберем некоторые, чтобы продемонстрировать (используя программные коды), как их использовать. Но перед этим определим новые исходные данные в следующей таблице 5.3 для манипуляции, где первая строка является индексом.

Таблица 5.3 – Данных для обработки

0	1	2	3	4	5	6	7	8	9
110	57.4	88.1	-46	213	-75.6	204	-86	103	49.3

Как получить минимальное значение данных среди положительных значений? Чтобы решить эту задачу, будем использовать две функции:

*«**mask()**»* – для выбора данных из условия и

*«**min()**»* – для нахождения этого минимального значения.

Таким образом, получим следующий код:

```
1  import pandas as pa
2
3  d_se = [110, 57.4, 88.1, -46, 213, -75.6, 204, -86, 103, 49.3]
4  data_se = pa.Series(data=d_se)
5  new_data = data_se.mask(data_se<0).min()
6  print("Мини. положительное значение: %.2f"%new_data)
```

Результат после выполнения кода

```
Мини. положительное значение: 49.30
```

*Еще одна дополнительная функция или другая возможность применения метода **mask()**?*

Si nous changeons le code de la 5-ième ligne par ceci:

```
5  new_data = data_se.mask(data_se<0, 50)
6  print("Данные: \n%s"% new_data)
```

В результате получим замену всех отрицательных значений цифрой 50.

Результат после выполнения кода
Данные: 0 110.0 1 57.4 2 88.1 3 50.0 4 213.0 5 50.0 6 204.0 7 50.0 8 103.0 9 49.3 dtype: float64

Как вернуть определенное количество ***n*** *элементов в порядке убывания или возрастания. В порядке убывания, будем использовать функцию «****nlargest()****», а в порядке возрастания – функцию «****nsmallest()****».*

Синтаксис функции *nlargest()*:

Séries.*nlargest* (*n* = 5 , *keep* = 'first')

Le parametre «***keep***» – peut prendre les valeurs suivantes : {'*first*', '*last*', '*all*'}, par défaut nous avons la valeur '*first*'.

Описание параметров:

n целое число: по умолчанию равно **5** – оно представляет количество значений, отсортированных в порядке убывания.

Параметр «***keep***» – может принимать следующие значения: {'*first*', '*last*', '*all*'}, по умолчанию принимает значение ‘first’.

Когда в списке повторяются определенные значения, можно ориентироваться на выбор из следующих значений:

«***first***»: возвращает первые ***n*** элементов в соответствии с условием в порядке появления.

«***last***»: возвращает последние ***n*** элементов в соответствии с условием в обратном порядке появления.

«***all***»: сохраняет все элементы, которые могут удовлетворить условию, что может привести к получению результата размером больше ***n***.

Пример 4:

Необходимо изменить значения массива ***d_se[i]*** (где ***i*** – индекс массива), так, чтобы визуально можно было понять разницу между параметрами функции ***nlargest()*** из результата; брать *n=4* и изменить различные значения параметров ***keep***. Для решения имеем следующий код:

```
1  import pandas as pa
2
3  d_se = [110, 57.4, 88.1, 110, 213, 55, 215, 104, -86, 201]
4  data_se = pa.Series(data=d_se)
5  new_data = data_se.nlargest(n=4, keep="first")
6  print("Данные: \n%s"%new_data)
```

Результат после выполнения кода
Данные: 6 215.0 4 213.0 9 201.0 0 110.0 dtype: float64

Если мы изменим код строки 5 следующим образом:

```
5  new_data = data_se.nlargest(n=4, keep="last")
```

Результат после выполнения кода
Данные: 6 215.0 4 213.0 9 201.0 3 110.0 dtype: float64

Если снова изменим код в строке 5 следующим образом:

```
5  new_data = data_se.nlargest(n=4, keep="all")
```

Результат после выполнения кода

```
Данные:
6    215.0
4    213.0
9    201.0
0    110.0
3    110.0
dtype: float64
```

Синтаксис функции *nsmallest()*:

```
Series.nsmallest(n=5, keep='first')
```

Описание параметров

Здесь имеем то же описание параметров, что и предыдущая функция *nlargest()*.

Пример 5:

```
1  import pandas as pa
2
3  d_se = [110, 57.4, 88.1, 110, 213, 55, 215, 104, -86, 201]
4  data_se = pa.Series(data=d_se)
5  new_data = data_se.nsmallest(n=4, keep="first")
6  print("Данные: \n%s"%new_data)
```

Результат после выполнения кода

```
Данные:
8   -86.0
5    55.0
1    57.4
2    88.1
dtype: float64
```

Если мы снова изменим код в строке 5,

```
5  new_data = data_se.nsmallest(keep="all")
```

то, мы получим 5 строк, соответствующих условию функции «*nsmallest*», поскольку ***n*** по умолчанию равно **5**:

Результат после выполнения кода

```
Данные:
8   -86.0
5    55.0
1    57.4
2    88.1
7   104.0
dtype: float64
```

*Как по-другому выполнить сортировку данных в «Series» ? Для этого «Pandas» дает возможность использовать функцию «**sort_index**» для сортировки массивных индексов и «**sort_values**» для сортировки значений.*

Синтаксис функции *sort_index ()*:

```
Series.sort_index(axis=0, level=None, ascending=True, inplace=False,
kind='quicksort', na_position='last', sort_remaining=True,
ignore_index=False, key=None)
```

Функция возвращает новый массив (одномерный), отсортированный по его индексу, в случае, если аргумент *«inplace»* имеет значение «*False*», в противном случае (если «*True*») он обновляет исходный массив и возвращает значение «None».

Описание параметров

axis – принимает значение 0, так как имеется тип « Series »;

level – это необязательно; если он не равен «None», то он сортирует значения по указанному уровню (или уровням) индекса;

ascending – по умолчанию он принимает значение «True», поэтому сортировка будет выполняться в порядке возрастания; если значение «False», то сортировка выполняется в порядке убывания;

inplace – если он имеет значение «True», операция выполняется на месте;

kind: {‘quicksort’, ‘mergesort’, ‘heapsort’, ‘stable’}, по умолчанию «quicksort»; с помощью этого параметра, имеем возможность выбрать тип алгоритма сортировки. См. также тему сортировки для получения дополнительной информации (Глава IV). Отметим, что «*mergesort*» и «*stable*» являются единственными стабильными алгоритмами;

na_position: {‘first’, ‘last’}, по умолчанию «last»; если у нас есть «first», тогда он ставит значение «NaN» в начале, а если «last», он ставит значение «NaN» в конце;

sort_remaining: по умолчанию значение «True»; он работает в том случае, если этот параметр имеет значение «True», а индексы находятся на нескольких уровнях, поэтому сортировка будет выполняться и на других уровнях индекса;

ignore_index – по умолчанию «False»; если значение «True», результирующая ось будет помечена от 0 до n – 1;

key – если это не значение «None», то необходимо использовать функцию *key* к значениям индекса перед сортировкой.

PS. NaN – Not a Number - Не число

Пример 6:

Предположим, что имеется корзина из 10 разных фруктов с более или менее отличающимися друг от друга ценами: "личи"=67 рублей, "банан"=70 рублей, "авокадо"=156,5 рублей, "кумкват"=105 рублей,

"ананас"=170 рублей, "апельсин"=120 рублей, "помело" =105 рублей, "лимон"=144 рубля, "Мандарин"=135,5 рубля, "грейпфрут"=99 рублей. Используйте функцию «sort_index» для сортировки имен фруктов в порядке возрастания и убывания. Отображение результата на экране с различными ценами.

```
1  import pandas as pa
2
3  d_se = [67, 70, 156.5, 105, 170, 120, 105, 144, 135.5, 99]
4  ind = ["Личи", "Банан", "Авокадо", "Кумкват", "Ананас",
5                  "Апельсин", "Помело", "Лимон",
6                  "Мандарин", "Грейпфрут"]
7  data_se = pa.Series(data=d_se, index=ind)
8  new_data = data_se.sort_index(axis=0, level=None,
9                  ascending=True, kind='stable')
10 print("Названия фруктов, отсортированных в порядке
11                 возрастания: \n%s"%new_data)
```

Результат после выполнения кода

```
Названия фруктов, отсортированных в порядке возрастания:
Авокадо       156.5
Ананас        170.0
Апельсин      120.0
Банан          70.0
Грейпфрут      99.0
Кумкват       105.0
Лимон         144.0
Личи           67.0
Мандарин      135.5
Помело        105.0
dtype: float64
```

Чтобы выполнить сортировку в порядке убывания, необходимо изменить 5-ю и 7-ю строки следующим образом (ascending=False):

```
5  new_data = data_se.sort_index(axis=0, level=None,
6                                 ascending=False, kind='stable')
7  print("Названия фруктов, отсортированных в порядке
8  убывания: \n%s"%new_data)
```

Результат после выполнения кода

```
Названия фруктов, отсортированных в порядке убывания:
Помело       105.0
Мандарин     135.5
Личи          67.0
Лимон        144.0
Кумкват      105.0
Грейпфрут     99.0
Банан         70.0
Апельсин     120.0
Ананас       170.0
Авокадо      156.5
dtype: float64
```

Синтаксис функции sort_values ():

```
Series.sort_values(axis=0, ascending=True, inplace=False,
kind='quicksort', na_position='last', ignore_index=False, key=None)
```

Позволяет сортировать данные (значения) типа «Series» (списка) в порядке возрастания или убывания в соответствии с критериями, определяемыми следующими параметрами: *axis*, *ascending*, *inplace*, *na_position*, *ignore_index*, *key* и особенно параметром *kind*, который позволяет направлять алгоритм или метод сортировки: метод «*quicksort*» – “быстрая сортировка”, «*mergesort*» – “сортировка по слиянию” или «heapsort» – “сортировка по пирамиде или по куче”.

Алгоритм, используемый по умолчанию, - это “быстрая сортировка” то есть «*quicksort*».

Описание параметров

Здесь мы имеем то же описание параметров, что и предыдущая функция *sort_index()*.

Исходя из данных предыдущего примера 6, попробуем по-разному сортировать цены на фрукты в порядке возрастания и убывания.

```
1  import pandas as pa
2
3  x = [67, 70, 156.5, 105, 170, 120, 105, 144, 135.5, 99]
4  ind = ["Личи", "Банан", "Авокадо", "Кумкват", "Ананас",
5                  "Апельсин", "Помело", "Лимон",
6                  "Мандарин", "Грейпфрут"]
7  data_se = pa.Series(data=x, index=ind)
8  new_data = data_se.sort_values()
9  print("Цены на фрукты, отсортированные в порядке
10             возрастания:\n%s"%new_data)
```

Результат после выполнения кода

```
Цены на фрукты, отсортированные в порядке возрастания:
Личи          67.0
Банан         70.0
Грейпфрут     99.0
Кумкват      105.0
Помело       105.0
Апельсин     120.0
Мандарин     135.5
Лимон        144.0
Авокадо      156.5
Ананас       170.0
dtype: float64
```

В строке № 8 кода мы не поместили никаких параметров, это означает, что функция *sort_values()* использует значения по умолчанию, где *ascending=True* и *kind= "quicksort"*. Чтобы получить значения в порядке убывания, необходимо заменить строки 8, 9 и 10 этими кодами:

```
8  new_data = data_se.sort_values(ascending=False)
9  print("Цены на фрукты, отсортированные в порядке
10                                  убывания: \n%s"%new_data)
```

Результат после выполнения кода
Цены на фрукты, отсортированные в порядке убывания: Ананас 170.0 Авокадо 156.5 Лимон 144.0 Мандарин 135.5 Апельсин 120.0 Кумкват 105.0 Помело 105.0 Грейпфрут 99.0 Банан 70.0 Личи 67.0 dtype: float64

RU V.2. Передача данных из одномерного массива (Series) в файл разных форматов

С помощью нескольких функций мы можем преобразовать данные из одномерного массива (из структуры «Series») в другие форматы данных или даже перенести эти данные в файлы типа, например, csv; excel и многие другие. Подробнее об этих функциях см. в таблице 5.4.

Таблица 5.4 – Некоторые функции для «Series»

1	to_csv([path_or_buf, sep, na_rep, ...])	Позволяет записать объект в файл типа *csv*
	*Series.to_csv(**path_or_buf**=None, **sep**=',', na_rep='', float_format=None, columns=None, header=True, index=True, index_label=None, mode='w', encoding=None, compression='infer', quoting=None, quotechar='"', line_terminator=None, chunksize=None, date_format=None, doublequote=True, escapechar=None, decimal='.', errors='strict', storage_options=None) ...*	

Продолжение табл. 5.4

1	***path_or_buf*** *– objet de type fichier,par défaut la valeur est None, cette valeur est de type String, (implémentation du système d'exploitation: os.PathLike[str]), ou objet de type fichier implémentant une fonction write().*	
2	to_excel(excel_writer[, sheet_name, na_rep, ...])	Позволяет записать объект в лист Excel
	Series.to_excel(excel_writer, sheet_name='Sheet1', na_rep='', float_format=None, columns=None, header=True, index=True, index_label=None, startrow=0, startcol=0, engine=None, merge_cells=True, encoding=None, inf_rep='inf', verbose=True, freeze_panes=None, storage_options=None)	
3	to_clipboard([excel, sep])	Позволяет скопировать объект во временное хранилище (называемое буфером обмена) системы
	*Series.to_clipboard(excel=True, sep=None, **kwargs)* *По умолчанию sep='\t' является разделителем полей* ***kwargs – эти параметры будут переданы в структуру DataFrame.to_csv*	
4	to_dict([into])	Позволяет преобразовать данные из Series {метка→ value}в объект типа словаря (dictionary).
5	to_list() ou tolist()	Возвращает список значений
6	to_string([buf, na_rep, float_format, ...])	Отображает строковое представление из «Series»
7	to_frame([name])	Позволяет преобразовать из «Series» в «DataFrame»
8	to_hdf(path_or_buf, key[, mode, complevel, ...])	Позволяет записывать данные в файл типа HDF5(Hierarchical Data Format: Иерархический Формат Данных – максимальный размер файла HDF: 2 Гб) с помощью HDFStore
	Series.to_hdf(path_or_buf, key, mode='a', complevel=None, complib=None, append=False, format=None, index=True, min_itemsize=None, nan_rep=None, dropna=None, data_columns=None, errors='strict', encoding='UTF-8')	

Окончание табл. 5.4

9	to_json([path_or_buf, orient, date_format, ...])	Преобразуйте объект в строку JSON – которая является сокращением от «JavaScript Object Notation», независимого от любого языка программирования формата для обмена данными в Интернете.
	Series.to_json(path_or_buf=None, orient=None, date_format=None, double_precision=10, force_ascii=True, date_unit='ms', default_handler=None, lines=False, compression='infer', index=True, indent=None, storage_options=None)	
10	to_latex([buf, columns, col_space, header, ...])	Позволяет отображать объект в массиве Latex (язык и система компоновки документов), в длинной таблице или во вложенной таблице
11	to_sql(name, con[, schema, if_exists, ...])	Позволяет записывать записи, хранящиеся в базе данных SQL
	Series.to_sql(name, con, schema=None, if_exists='fail', index=True, index_label=None, chunksize=None, dtype=None, method=None)	
12	to_markdown([buf, mode, index, storage_options])	Позволяет печатать данные из «Series» в формате «Markdown» – (легкий язык разметки)
13	to_numpy([dtype, copy, na_value])	Позволяет превращаться в массивы типа «NumPy»
14	to_xarray()	Возвращает объект массива из объекта Pandas.
15	to_period([freq, copy])	Используется для преобразования данных из этого объекта «Series» в индекс с определенной частотой.
16	to_pickle(path[, compression, protocol, ...])	Позволяет разрешить сериализацию объектов в файле.
17	to_timestamp([freq, how, copy])	Применяется для преобразования индекса периода в индекс даты и времени(DatetimeIndex)

RU V.3. Другие методы с использованием «Series»

В принципе существует множество функций, которые можно использовать для расширения функциональности обработки данных в «Series». В таблице 5.5 имеются около ста методов.

Таблица 5.5 – Другие функции «Series»

1	abs()
2	add(other[, level, fill_value, axis])
3	add_prefix(prefix)
4	add_suffix(suffix)
5	agg([func, axis])
6	aggregate([func, axis])
7	align(other[, join, axis, level, copy, ...])
8	all([axis, bool_only, skipna, level])
9	any([axis, bool_only, skipna, level])
10	concat (objs, axis = 0, join='outer',...)
11	apply(func[, convert_dtype, args])
12	argmax([axis, skipna])
13	argmin([axis, skipna])
14	argsort([axis, kind, order])
15	asfreq(freq[, method, how, normalize, ...])
16	asof(where[, subset])
17	astype(dtype[, copy, errors])
18	at_time(time[, asof, axis])
19	autocorr([lag])
20	backfill([axis, inplace, limit, downcast])
21	between(left, right[, inclusive])
22	between_time(start_time, end_time[, ...])
23	bfill([axis, inplace, limit, downcast])
24	bool()
25	cat
26	clip([lower, upper, axis, inplace])
27	combine(other, func[, fill_value])
28	combine_first(other)
29	compare(other[, align_axis, keep_shape, ...])

Продолжение табл. 5.5

30	convert_dtypes([infer_objects, ...])
31	copy([deep])
32	corr(other[, method, min_periods])
33	count([level])
34	cov(other[, min_periods, ddof])
35	cummax([axis, skipna])
36	cummin([axis, skipna])
37	cumprod([axis, skipna])
38	cumsum([axis, skipna])
39	describe([percentiles, include, exclude, ...])
40	diff([periods])
41	div(other[, level, fill_value, axis])
42	divide(other[, level, fill_value, axis])
43	divmod(other[, level, fill_value, axis])
44	dot(other)
45	drop([labels, axis, index, columns, level, ...])
46	drop_duplicates([keep, inplace])
47	droplevel(level[, axis])
48	dropna([axis, inplace, how])
49	dt
50	duplicated([keep])
51	eq(other[, level, fill_value, axis])
52	equals(other)
53	ewm([com, span, halflife, alpha, ...])
54	expanding([min_periods, center, axis, method])
55	explode([ignore_index])
56	factorize([sort, na_sentinel])
57	ffill([axis, inplace, limit, downcast])
58	fillna([value, method, axis, inplace, ...])
59	filter([items, like, regex, axis])
60	first(offset)
61	first_valid_index()
62	floordiv(other[, level, fill_value, axis])
63	ge(other[, level, fill_value, axis])
64	get(key[, default])

Продолжение табл. 5.5

65	groupby([by, axis, level, as_index, sort, ...])
66	gt(other[, level, fill_value, axis])
67	head([n])
68	hist([by, ax, grid, xlabelsize, xrot, ...])
69	idxmax([axis, skipna])
70	idxmin([axis, skipna])
71	infer_objects()
72	info([verbose, buf, max_cols, memory_usage, ...])
73	interpolate([method, axis, limit, inplace, ...])
74	isin(values)
75	isna()
76	isnull()
77	item()
78	items()
79	iteritems()
80	keys()
81	kurt([axis, skipna, level, numeric_only])
82	kurtosis([axis, skipna, level, numeric_only])
83	last(offset)
84	last_valid_index()
85	le(other[, level, fill_value, axis])
86	lt(other[, level, fill_value, axis])
87	mad([axis, skipna, level])
88	map(arg[, na_action])
89	mask(cond[, other, inplace, axis, level, ...])
90	max([axis, skipna, level, numeric_only])
91	mean([axis, skipna, level, numeric_only])
92	median([axis, skipna, level, numeric_only])
93	memory_usage([index, deep])
94	min([axis, skipna, level, numeric_only])
95	mod(other[, level, fill_value, axis])
96	mode([dropna])
97	mul(other[, level, fill_value, axis])
98	multiply(other[, level, fill_value, axis])
99	ne(other[, level, fill_value, axis])

Продолжение табл. 5.5

100	nlargest([n, keep])
101	notna()
102	notnull()
103	nsmallest([n, keep])
104	nunique([dropna])
105	pad([axis, inplace, limit, downcast])
106	pct_change([periods, fill_method, limit, freq])
107	pipe(func, *args, **kwargs)
108	plot
109	pop(item)
110	pow(other[, level, fill_value, axis])
111	prod([axis, skipna, level, numeric_only, ...])
112	product([axis, skipna, level, numeric_only, ...])
113	quantile([q, interpolation])
114	radd(other[, level, fill_value, axis])
115	rank([axis, method, numeric_only, ...])
116	ravel([order])
117	rdiv(other[, level, fill_value, axis])
118	rdivmod(other[, level, fill_value, axis])
119	reindex(*args, **kwargs)
120	reindex_like(other[, method, copy, limit, ...])
121	rename([index, axis, copy, inplace, level, ...])
122	rename_axis([mapper, index, columns, axis, ...])
123	reorder_levels(order)
124	repeat(repeats[, axis])
125	replace([to_replace, value, inplace, limit, ...])
126	resample(rule[, axis, closed, label, ...])
127	reset_index([level, drop, name, inplace])
128	rfloordiv(other[, level, fill_value, axis])
129	rmod(other[, level, fill_value, axis])
130	rmul(other[, level, fill_value, axis])
131	rolling(window[, min_periods, center, ...])
132	round([decimals])
133	rpow(other[, level, fill_value, axis])
134	rsub(other[, level, fill_value, axis])

Окончание табл. 5.5

135	rtruediv(other[, level, fill_value, axis])		
136	sample([n, frac, replace, weights, ...])		
137	searchsorted(value[, side, sorter])		
138	sem([axis, skipna, level, ddof, numeric_only])		
139	set_axis(labels[, axis, inplace])		
140	set_flags(*[, copy, allows_duplicate_labels])		
141	shift([periods, freq, axis, fill_value])		
142	skew([axis, skipna, level, numeric_only])		
143	shift(...)		
144	sort_index([axis, level, ascending, ...])		
145	sort_values([axis, ascending, inplace, ...])		
146	sparse	147	str
148	squeeze([axis])		
149	std([axis, skipna, level, ddof, numeric_only])		
150	sub(other[, level, fill_value, axis])		
151	subtract(other[, level, fill_value, axis])		
152	sum([axis, skipna, level, numeric_only, ...])		
153	swapaxes(axis1, axis2[, copy])		
154	swaplevel([i, j, copy])	155	tail([n])
156	take(indices[, axis, is_copy])		
157	transform(func[, axis])		
158	transpose(*args, **kwargs)		
159	truediv(other[, level, fill_value, axis])		
160	truncate([before, after, axis, copy])		
161	tz_convert(tz[, axis, level, copy])		
162	tz_localize(tz[, axis, level, copy, ...])		
163	unique()		
164	unstack([level, fill_value])		
165	update(other)		
166	value_counts([normalize, sort, ascending, ...])		
167	var([axis, skipna, level, ddof, numeric_only])		
168	view([dtype])		
169	where(cond[, other, inplace, axis, level, ...])		
170	xs(key[, axis, level, drop_level])		

RU V.4. Pandas в двумерном массиве – DataFrame

RU V.4.1. Хранение и вставка данных

Как хранить данные в таблице с помощью DataFrame? DataFrame имеет двумерную структуру данных (см. рисунок 5.1), которая может хранить данные разных типов: символы, целые числа, значения с плавающей запятой, текстовые значения и т. д. Также следует помнить, что каждый столбец DataFrame является структурой «Series» (вертикальной последовательностью значений (данных)).

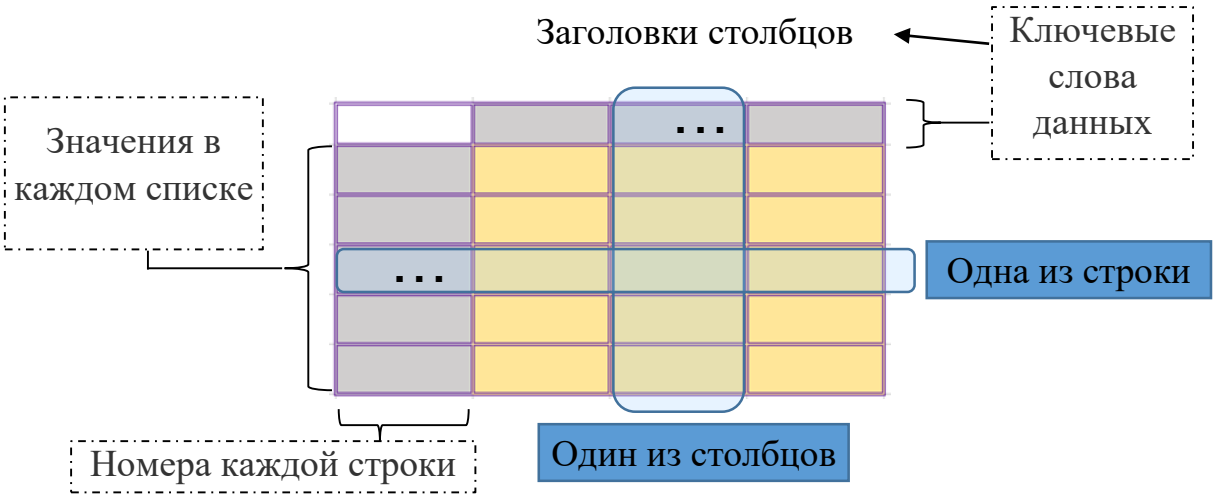


Рис. 5.1. Структура таблицы «DataFrame»

Пример 7

Предположим, у нас есть 4 студента, отсутствующих в каком-либо классе. Необходимо представить данные этих студентов в «DataFrame», зная, что нам известно их полные имена (Иванов Иван, Петров Дима, Сидоров Саша и Панова Ирина); даты их рождения (10.02.2000, 23.11.2001, 12.05.1999 и 27.04.2000) и номера их удостоверений личности (233, 632,595 и 810). Не трудно заметить, что эти данные имеют разные типы (текстовые значения – символы, даты

и целые числа). Задача состоит в том, чтобы хранить эти данные студентов с помощью Pandas. Как это сделать таким образом, чтобы получить результат, показанный на рисунке 5.2?

	ФИО	Даты рождения	Номера удостоверений личности
0	Иванов Иван	10.02.2000	233
1	Петров Дима	23.11.2001	632
2	Сидоров Саша	12.05.1999	595
3	Панова Ирина	27.04.2000	810

Рис. 5.2. DataFrame из примера №1 (результат)

Синтаксис «DataFrame» в Pandas выглядит следующим образом:

DataFrame(data=None, index: Axes | None=None, columns: Axes | None=None, dtype: Dtype | None=None, copy: bool | None=None),

где ***data*** – может содержать: данные в форме “словаря” – «dictionary», либо серийные данные («Series»), либо массивные данные, либо постоянные данные, либо классы данных, либо объекты типа списка;

index – это метка, используемая для полученного изображения. По умолчанию он варьируется от 0 до числа строк, которые нужно вывести;

columns – имена меток столбцов для разных данных;

dtype – принудительный тип данных (по умолчанию None, если он существует, разрешен только один «dtype»): int, int8, int64, object и т. д.

copy (логический тип, по умолчанию нет – «None») – позволяет копировать данные из входных данных. Создает копию массива данных, если параметр copy имеет значение *True*. А если *copy = False*, то ничего не происходит.

Решение: используем словарь для построения DataFrame:

```
import pandas as pa

data_p = pa.DataFrame(
    {
        "ФИО": [
            "Иванов Иван",
            "Петров Дима",
            "Сидоров Саша",
            "Панова Ирина"
        ],
        "Даты рождения": [
            "10.02.2000",
            "23.11.2001",
            "12.05.1999",
            "27.04.2000"
            ],
        "Номера удостоверений личности": [
            233,
            632,
            595,
            810
            ]
    }
)

print(data_p)
```

Результат после выполнения кода			
	ФИО	Даты рождения	Номера удостоверений личности
0	Иванов Иван	10.02.2000	233
1	Петров Дима	23.11.2001	632
2	Сидоров Саша	12.05.1999	595
3	Панова Ирина	27.04.2000	810

Комментарий и внесение некоторых изменений
Можно наблюдать автоматическое появление номеров в каждой строке от 0 до 3, учитывая, что имеются фактически 4 строки данных после выполнения предшествующего кода. Рассмотрим, как изменить эти номера на другие, например, на метки “а”, “б”, “в” и “г” с помощью параметра «index».

 *Как изменить эти номера?*

```
1  import pandas as pa
2
3  d = {
4        "ФИО":     ["Иванов Иван", "Петров Дима", "Сидоров
5                    Саша", "Панова Ирина"],
6        "Даты рождения":   ["10.02.2000", "23.11.2001",
7                    "12.05.1999", "27.04.2000"],
8     "Номера удостоверений личности": [233, 632,595, 810]
9      }
10
11 data_p2 = pa.DataFrame(data=d, index= ["а","б","в","г"])
12 print(data_p2)
```

Результат после выполнения кода

	ФИО	Даты рождения	Номера удостоверений личности
а	Иванов Иван	10.02.2000	233
б	Петров Дима	23.11.2001	632
в	Сидоров Саша	12.05.1999	595
г	Панова Ирина	27.04.2000	810

Замечания, которые мы можем сделать:

Полученная таблица состоит из 4 столбцов с меткой на каждом столбце «ФИО», «Даты рождения», «Номера удостоверений личности».

В столбце «ФИО» есть строковые значения; в столбце «Даты рождения» есть строковые значения, которые можно преобразовать в тип «Datetime» т.е. дата и время; в столбце «Номера удостоверений личности» есть значения типа «int» – целые числа.

В таблицах типа «DataFrame» один и тот же столбец может иметь значения разных типов.

Мы также обнаруживаем, что каждый столбец является «Series».

RU V.4.2 Возможные операции с четко определенным столбцом

Чтобы обратиться к определенному столбцу, можо использовать следующую команду:

```
Название таблицы [“Название этикетки”]
```

Изменения, которые необходимо внести в предыдущий код:

```
11 data_p2 = pa.DataFrame(data=d)
12 col_1 = data_p2["Номера удостоверений личности"]
13 print(col_1)
```

Результат после выполнения кода	
0	233
1	632
2	595
3	810

Чтобы обратиться к элементу в данном столбце, можем использовать следующую команду:

Пример:

```
11 data_p2 = pa.DataFrame(data=d)
12 col_1 = data_p2["Номера удостоверений личности"]
13 elt = col_1[n]
```

col_1 – это переменная, которая будет получать все элементы в столбце “*Номера удостоверений личности*”;

data_p2 – массив DataFrame;

n – номер элемента в «Series» (столбце).

Чтобы полностью напечатать столбец

```
12 col_1 = data_p2["Номера удостоверений личности"]
13 print(col_1[:])
   # или ещё
13 print(col_1)
```

Для печати первых двух элементов столбца

```
13 col_1 = data_p2["Номера удостоверений личности"]
14 print(col1[:2])
15
```

Для печати всех элементов столбца, которые стоят после первого

```
13 col_1 = data_p2["Номера удостоверений личности"]
14 print(col_1[1:])
15
```

Для печати всех элементов столбца, которые стоят после второго

```
13  col_1 = data_p2["Номера удостоверений личности"]
14  print(col_1[2:])
15
```

Чтобы найти и распечатать минимальное значение столбца

```
13  vmin = data_p2["Номера удостоверений личности"].min()
14  print(vmin)
15
```

Чтобы найти и распечатать максимальное значение столбца

```
13  vmax =data_p2["Номера удостоверений личности"].max()
14  print(vmax)
15
```

Для сортировки заданного столбца в порядке возрастания в таблице «DataFrame»

Чтобы отсортировать этот «DataFrame» по одному столбцу, необходимо указать имя столбца с помощью параметра «by».

С помощью параметра «ascending», упорядочиваем порядок сортировки по возрастанию (со значением «True») или по убыванию (со значением «False»). По умолчанию функция «sort_value» сортирует по возрастанию.

```
13  tri1 =data_p2.sort_values(by="Номера удостоверений личности",
                              ascending=True)
14  print(tri1)
```

Результат после выполнения кода

	ФИО	Даты рождения	Номера удостоверений личности
0	Иванов Иван	10.02.2000	233
2	Сидоров Саша	12.05.1999	595
1	Петров Дима	23.11.2001	632
3	Панова Ирина	27.04.2000	810

С параметром «inplace» – сортировка будет выполняться в массиве «DataFrame» на месте. Новая переменная не сможет получить результат этой сортировки. Результат останется в старом массиве, для этого параметру «inplace» необходимо присвоить значение «True».

```
13 data_p2.sort_values(by="ФИО", ascending=True, inplace=True)
   print(data_p2)
14
```

Результат после выполнения кода

	ФИО	Даты рождения	Номера удостоверений личности
0	Иванов Иван	10.02.2000	233
3	Панова Ирина	27.04.2000	810
1	Петров Дима	23.11.2001	632
2	Сидоров Саша	12.05.1999	595

Для сортировки заданного столбца в порядке возрастания в таблице «DataFrame».

Чтобы выполнить сортировку по нескольким столбцам таблицы «DataFrame», всегда используя метод «sort_values», нужно указать имена столбцов, которые будем использовать для сортировки, через запятую. После этого с параметром «ascending» также потребуется указать порядок сортировки столбцов, разделенных запятыми. Итак, давайте попробуем выполнить эту сортировку по столбцам «ФИО» и «Номера удостоверений личности».

```
13  tri2 = data_p2.sort_values(["ФИО","Номера удостоверений
                личности"], ascending=[True,True])
14  print(tri2)
```

Результат после выполнения кода

	ФИО	Даты рождения	Номера удостоверений личности
0	Иванов Иван	10.02.2000	233
3	Панова Ирина	27.04.2000	810
1	Петров Дима	23.11.2001	632
2	Сидоров Саша	12.05.1999	595

Этот результат создает впечатление, что последний столбец не был отсортирован. Чтобы убедиться, что все работает нормально, давайте попробуем внести некоторые изменения в исходные данные «DataFrame». Давайте сменим «Сидоров Саша» на новый «Петров Дима» с датой рождения 12.05.1999 года и номером удостоверения личности, равным 595. После этого изменения выполнение предыдущего кода даст следующий результат:

Результат после выполнения кода

	ФИО	Даты рождения	Номера удостоверений личности
0	Иванов Иван	10.02.2000	233
3	Панова Ирина	27.04.2000	810
2	Петров Дима	12.05.1999	595
1	Петров Дима	23.11.2001	632

RU V.4.3 Некоторые операции и взаимодействия между Excel и Pandas

Поскольку Excel, OpenOffice calc и многие другие электронные таблицы используются многими пользователями компьютеров, важно знать, как интегрировать данные из Pandas или манипулировать ими.

Воспользуемся данными по следующей ссылке, чтобы узнать, как выполнять разные операции. Этот файл содержит некоторые данные (см. рис. 5.3) в формате csv.

url = https://raw.githubusercontent.com/pandas-dev/pandas/main/pandas/tests/io/data/csv/tips.csv

```
total_bill,tip,sex,smoker,day,time,size
16.99,1.01,Female,No,Sun,Dinner,2
10.34,1.66,Male,No,Sun,Dinner,3
21.01,3.5,Male,No,Sun,Dinner,3
23.68,3.31,Male,No,Sun,Dinner,2
24.59,3.61,Female,No,Sun,Dinner,4
25.29,4.71,Male,No,Sun,Dinner,4
8.77,2.0,Male,No,Sun,Dinner,2
26.88,3.12,Male,No,Sun,Dinner,4
15.04,1.96,Male,No,Sun,Dinner,2
14.78,3.23,Male,No,Sun,Dinner,2
10.27,1.71,Male,No,Sun,Dinner,2
35.26,5.0,Female,No,Sun,Dinner,4
15.42,1.57,Male,No,Sun,Dinner,2
18.43,3.0,Male,No,Sun,Dinner,4
```

Рис. 5.3. Структура данных в URL файла csv

Как прочитать данные из файла, который находится в URL-адресе

```
1  import pandas as pa
2
3  url = "https://raw.githubusercontent.com/pandas-
4                    dev/pandas/main/pandas/tests/io/data/csv/tips.csv"
5  data = pa.read_csv(url)
6  infos = data.head(6)
7  print(infos)
```

Результат после выполнения кода

```
   total_bill   tip     sex smoker day   time  size
0       16.99  1.01  Female     No Sun Dinner     2
1       10.34  1.66    Male     No Sun Dinner     3
2       21.01  3.50    Male     No Sun Dinner     3
3       23.68  3.31    Male     No Sun Dinner     2
4       24.59  3.61  Female     No Sun Dinner     4
5       25.29  4.71    Male     No Sun Dinner     4
```

Если вместо строки № 6 мы поставим:

```
6  infos = data.tail(5)
```

получим последние 5 строк файла.

Résultat après exécution du code

```
     total_bill   tip     sex smoker  day   time  size
239       29.03  5.92    Male     No  Sat Dinner     3
240       27.18  2.00  Female    Yes  Sat Dinner     2
241       22.67  2.00    Male    Yes  Sat Dinner     2
242       17.82  1.75    Male     No  Sat Dinner     2
243       18.78  3.00  Female     No Thur Dinner     2
```

Особенностью Pandas является то, что он позволяет напрямую выполнять операции над целыми столбцами в отличии от других электронных таблиц. Рассмотрим следующий пример, чтобы понять, как это работает.

Предположим, что мы хотим изменить данные в столбце [sex] файла « *tips.csv* » на символы « F » и « M ». « F » и должны заменить слова « Female » и « M » словом « Male ». Поскольку у нас есть только два слова, можем использовать метод « *where* », связанный с модулем «NumPy». В то время как в Excel будем использовать условие «если – то» (см. рис. 5.4).

```
1  import pandas as pa
2  import numpy as nu
3
4  url = "https://raw.githubusercontent.com/pandas-
5                    dev/pandas/main/pandas/tests/io/data/csv/tips.csv"
6  data = pa.read_csv(url)
7  data["sex"] = nu.where(data["sex"] = = "Female", "F", "M")
8  infos = data.head(5)
9  print(infos)
```

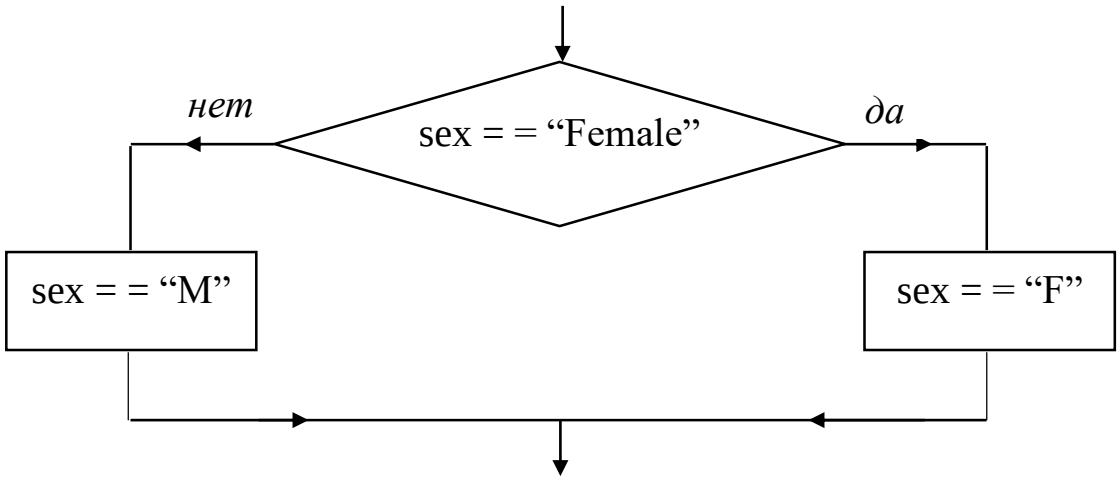


Рис. 5.4. Иллюстрация функции «если – то»

Результат после выполнения кода

	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	F	No	Sun	Dinner	2
1	10.34	1.66	M	No	Sun	Dinner	3
2	21.01	3.50	M	No	Sun	Dinner	3
3	23.68	3.31	M	No	Sun	Dinner	2
4	24.59	3.61	F	No	Sun	Dinner	4
5	25.29	4.71	M	No	Sun	Dinner	4

Можно удалить столбец с помощью функции « drop », переименовать имя столбца с помощью функции « rename », найти максимальное значение « max », минимальное « min », ... столбца. Таким образом, существует множество функций, которые можно применить к столбцам файла. С помощью функции « columns », можем перечислить имена разных столбцов в «*DataFrame*».

Коды могут выглядеть примерно, как эти строки:

```
new_value = data["name_colonne"].fonction()
```

```
data = data.drop("column_name ", axis=1)
```

```
infos = data.rename(columns={"old_name": "new_name"}).tail(6)
```

```
infos = data["column_name "].str.find("what_to_find").tail(6)
```

Чтобы обрабатывать строками, можно использовать существующие методы, которые были описаны в предыдущем книге – Том I или в других источниках.

 Возможно ли в Python с помощью библиотеки Pandas сохранить данные в Excel? Конечно, да. Для этого достаточно просто использовать функцию ***to_excel()*** с ее параметрами. Синтаксис этой функции выглядит следующим образом:

DataFrame.to_excel(*excel_writer*, *sheet_name*='Sheet1', *na_rep*='', *float_format*=None, *columns*=None, *header*=True, *index*=True, *index_label*=None, *startrow*=0, *startcol*=0, *engine*=None, *merge_cells*=True, *encoding*=None, *inf_rep*='inf', *verbose*=True, *freeze_panes*=None, *storage_options*=None),

где

excel_writer: путь к файлу или наличие объекта **ExcelWriter**;

sheet_name: имя листа, который будет содержать данные (по умолчанию ‘Sheet1’);

na_rep: представление отсутствующих данных;

float_format: строковый формат для чисел с плавающей запятой;

columns: столбец (столбцы), который(е) будет(ут) сохранен(ы) вместе с данными;

header: напишите имена столбцов. Если задан список строк, предполагается, что они являются псевдонимами для имен столбцов.

index: включить (значение True) или исключить (False) имена (индексы) строк;

index_label: название (метка) столбца для строки индексов;

startrow: координата x первой строки ячейки сверху влево для записи данных;

startcol: координата y первого столбца ячейки сверху вниз для записи данных;

engine: определить движок для применения: “openpyxl” или “xlsxwriter”;

merge_cells: записывает многоиндексные и иерархические строки в виде объединенных ячеек;

encoding: кодировка результирующего файла Excel; {« openpyxl », }

inf_rep: представление для бесконечности (в Excel нет собственного представления для бесконечности);
verbose: отображает дополнительную информацию в журналах ошибок;
freeze_panes: указывает самую нижнюю строку и самый правый столбец, которые необходимо “заморозить”;
storage_options: дополнительные параметры, которые имеют смысл для конкретного соединения с хранилищем, например «host», «port», «username», «password» и т. д.

 Синтаксис «**ExcelWriter**»

ExcelWriter(*path*: FilePathOrBuffer | ExcelWriter, *engine*=None, *date_format*=None, *datetime_format*=None, *mode*: str="w", *storage_options*: StorageOptions=None, *if_sheet_exists*: str | None=None, *engine_kwargs*: dict | None=None)

где
path: путь к файлу типа *xls, xlsx* или *ods*;
engine: движок, используемый для записи;
date_format: позволяет форматировать строку для дат, записанных в файлы Excel ;
(Например, 'ГГГГ-ММ-ДД') ;
datetime_format: permet de formater la chaîne de format pour les objets datetime écrits dans des fichiers Excel ;
datetime_format: используется для форматирования строки для объектов *datetime*, записанных в файлы Excel ;
(Например, 'ГГГГ-ММ-ДД ЧЧ:ММ:СС') ;
mode: режим использования файла (запись или добавление -{'w', 'a' или 'r+'}, по умолчанию 'w');
storage_options: дополнительные опции, которые имеют смысл для конкретного подключения к хранилищу, например «host», «port», «username», «password» и т. д.

if_sheet_exists: позволяет направлять запись на уже существующий лист excel (только в режиме добавления) - {'*error*', '*new*', '*replace*', '*overlay*'}, по умолчанию значение равно “error”;

error: выдает ошибку ValueError;

new: создать новый лист с именем, определяемым автоматически;

replace: удаляет содержимое листа перед записью на него нового;

overlay : запись содержимого в существующий лист без удаления старого содержимого;

engine_kwargs: аргументы ключевых слов для передачи в движок engine.

xlsxwriter (режим записи);

openpyxl (в режиме записи и добавления);

odswriter.

 Пример 8 кода для записи данных в Excel с помощью функции ***to_excel()***. **Случай, когда файл (*file_excel1.xlsx*) не существует**.

```
1  import os
2  import pandas as pa
3  from pandas.io.excel import ExcelWriter
4
5  if __name__ == "__main__":
6      path_file = os.path.dirname(__file__) +
7                          r"\FolderData\file_excel1.xlsx"
8      data_p1 = pa.DataFrame(
9          [['Иван', 34],
10          ['Ирина', 25],
11          ['Рома', 31]],
12         columns=['имя', 'возраст'])
13     with ExcelWriter(path_file,  mode="w") as exemplaire_write:
14         data_p1.to_excel(exemplaire_write, sheet_name="Название
15                          листа_1",  index_label="№", startrow=0,
16                          startcol=0)
```

Результат после выполнения кода

	A	B	C
1	№	имя	возраст
2	0	Иван	34
3	1	Ирина	25
4	2	Рома	31
5			

Название листа_1

Пример 9 кода для записи данных в Excel с помощью функции *to_excel()*. **Случай, когда файл (*file_excel1.xlsx*) существует**, и мы хотим добавить второй лист с другими данными в тот же файл.

Для этого необходимо создать объект **ExcelWriter** с именем файла, о котором идет речь, после указания листа в файле для сохранения.

```
1  import os
2  import pandas as pa
3  from pandas.io.excel import ExcelWriter
4
5  if __name__ == "__main__":
6      path_file = os.path.dirname(__file__) +
7                      r"\FolderData\file_excel1.xlsx"
8      data_p2 = pa.DataFrame(
9          [['Январь', 31],
10          ['Февраль', 28],
11          ['Март', 31]],
12         columns=['Месяц', 'количество дней'])
13     with ExcelWriter(path_file,  mode="a", engine="openpyxl") as
14                     exemplaire_write:
15         data_p2.to_excel(exemplaire_write, sheet_name="Название
16                     листа_2",  index_label="№", startrow=0,
17                     startcol=0)
```

Результат после выполнения кода

	A	B	C	D	E
1	№	Месяц	количество дней		
2	0	Январь	31		
3	1	Февраль	28		
4	2	Март	31		
5					

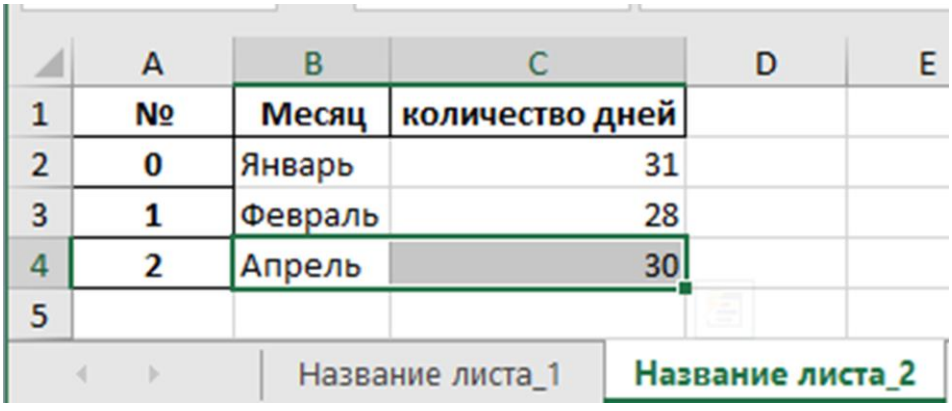
Название листа_1 | Название листа_2

Пример 10 кода для записи данных в Excel с помощью функции *to_excel()*. **Как обновить файл**, изменив только одну

информацию? Попробуем изменить «Март» на «Апрель» с количеством дней, равным 30.

```
1  import os
2  import pandas as pa
3  from pandas.io.excel import ExcelWriter
4
5  if __name__ == "__main__":
6      path_file = os.path.dirname(__file__) +
7                  r"\FolderData\file_excel1.xlsx"
8      data_p2 = pa.DataFrame([['Апрель', 30]], columns=None)
9
10     with ExcelWriter(path_file,  mode="a", engine="openpyxl",
11     if_sheet_exists="overlay") as exemplaire_write:
12         data_p2.to_excel(exemplaire_write, sheet_name="Название
13                 листа_2", index=False, header=False, startrow=3,
14                 startcol=1)
```

Результат после выполнения кода



	A	B	C	D	E
1	№	Месяц	количество дней		
2	0	Январь	31		
3	1	Февраль	28		
4	2	Апрель	30		
5					

Название листа_1 | Название листа_2

Основным параметром для обновления является «***if_sheet_exists***» со значением «***overlay***». С помощью этого параметра можно сохранить старые данные, стереть эти старые данные на одном листе или

автоматически создать новый лист для размещения новых данных на нем.

Архивировать файл. Также можно упаковать файл Excel в zip-архив. Для этого необходимо делать следующие действия:

- указать путь, по которому будет храниться сжатый файл;
- вводить данные (например, в DataFrame), которые будут храниться в файле Excel;
- поместить данные в файл Excel, который будет упакован;
- с помощью метода « ZipFile » необходимо сжать и упаковать файл.

```
1  import os, zipfile
2  import pandas as pa
3
4  if __name__ == "__main__":
5      path_file1 = os.path.dirname(__file__) +
6                          r"\FolderData\file_zip1.zip"
7      data_p3 = pa.DataFrame(
8          [['Январь', 31],
9           ['Февраль', 28],
10          ['Март', 31]],
11         columns=['Месяц', 'количество дней'])
12
13 with zipfile.ZipFile(path_file1, "w") as zpf:
14     with zpf.open("file_name_excel.xlsx", "w") as data_temp:
15         with pa.ExcelWriter(data_temp) as exemplaire_writer:
16             data_p3.to_excel(exemplaire_writer)
```

Результат после выполнения кода
Имеется файл (*file_name_excel.xlsx*) с данными, сжатыми в «*zip*» под названием: *file_zip1.zip*

Создайте документ Excel со следующими данными (см. рис. 5.5):

	A	B	C	D
1	№	Фамилия	Имя	Оценка
2	1	Петров	Давид	3,24
3	2	Иванов	Иван	15,02
4	3	Сидоров	Виктор	10,1
5	4	Лискок	Андрей	9,12
6	5	Ефимова	Настя	11,5
7	6	Кузнецов	Кирилл	5,76
8	7	Калинина	Татьяна	18,2
9	8	Зайцева	Лена	14,11
10	9	Шутов	Максим	4,97
11	10	Гусева	Юлия	7,45
12	11	Земцова	Алина	8,26
13				

Лист1 Лист2

	A	B	C	D
1	№	Дата	Количество	Религия
2	1	20.02.2014	45	Католики
3	2	11.08.2003	123	Православные
4	3	07.10.2011	23	Протестанты
5	4	25.05.2008	56	Мусульмане
6	5	12.06.2010	201	Католики
7	6	17.01.2020	67	Православные
8	7	02.12.2013	110	Православные
9	8	23.03.2015	45	Мусульмане
10	9	17.07.2003	120	Протестанты
11	10	29.10.2018	79	Православные
12	11	05.09.2009	88	Язычники
13				

Лист1 Лист2

Рис. 5.5. Файл Excel «*data_excel1.xlsx*» на 2 листах

Метод «***read_excel***» может считывать локальные системные файлы со следующими расширениями: *xls, xlsx, xlsm, xls, odt, ods* и *odt* или же еще может считывать эти файлы с URL-адреса. Синтаксис функции ***pandas.read_excel***.

*pandas.**read_excel**(io, sheet_name=0, header=0, names=None, index_col=None, usecols=None, squeeze=None, dtype=None, engine=None, converters=None, true_values=None, false_values=None, skiprows=None, nrows=None, na_values=None, keep_default_na=True, na_filter=True, verbose=False, parse_dates=False, date_parser=None, thousands=None, decimal='.', comment=None, skipfooter=0, convert_float=None, mangle_dupe_cols=True, storage_options=None)*

io – имя (путь) файла для чтения;

sheet_name: он используется для указания имен(и) листов для чтения. Значение по умолчанию равно 0 (первый лист); со значением 1: второй лист для использования; "Sheet1": лист для использования с именем "Sheet1"; если значение равно «None», то будут прочитаны все электронные листы.

header – он используется для меток столбцов;

names – он содержит список используемых меток столбцов;

index_col – столбец для использования в качестве меток строк;

dtype – тип данных для столбцов;

usecols – список букв или диапазонов столбцов Excel (например, "A: F" или "A, D, F: J"; другие примеры: usecols='B:D' или usecols=[0,2] или usecols=['Nom', 'Оценка'];

engine – различные движки для манипуляций: “xlrd”, “openpyxl”, “odf”, “pyxlsb”;

“xlrd " поддерживает старые файлы Excel с расширением (.xls).

“openpyxl” поддерживает новые форматы файлов Excel.

“odf” поддерживает форматы файлов OpenDocument (.odf, .ods,. odt).

“pyxlsb” поддерживает двоичные файлы Excel.

converters – позволяет преобразовывать значения в определенных столбцах;

skiprows – позволяет исключить одну или несколько строк; например: skiprows=[1] исключает вторую строку; skiprows=[2,3] – устраняет третью и четвертую строки; skiprows=2 – устраняет первые 2 строки.

Как прочитать данные с листа № 2 файла Excel «*data_excel1.xlsx*» на рисунке 5.5? Выводить первые 3 строки на экране.

```
1  import os
2  import pandas as pa
3
4  path_file = os.path.dirname(__file__) +
5                                  r"\FolderData\data_excel1.xlsx"
6  data_p = pa.read_excel(path_file, sheet_name='Лист2')
7
8  print(data_p.head(3))
```

Результат после выполнения кода

	№	Дата	Количество	Религия
0	1	2014-02-20	45	Католики
1	2	2003-08-11	123	Православные
2	3	2011-10-07	23	Протестанты

Точно так же можно считывать данные с другого листа и выполнять операции.

Можем использовать специальную функцию лямбда (***lambda***) в этих операциях, чтобы облегчить некоторые вычисления. Например, если хотим исключить значительное количество строк, то для этого будет менее удобно перечислять все эти строки в квадратных скобках, чем использовать следующую простую формулу. Шестая строка программы будет выглядеть примерно так:

```
6  data_p = pa.read_excel(path_file, sheet_name='Лист1',
7                         skiprows=lambda y: y<3)
```

Вместо цифры 3 может быть другая цифра большего значения. Эта функция удаляет первые 3 строки.

Для получения дополнительной информации о возможностях «DataFrame», можно перейти по ссылке на следующем веб-сайте Pandas: [*https://pandas.pydata.org/pandas-docs/stable/reference/frame.html*]. Речь идет: о преобразовании данных, статистических расчетах, манипулировании этикетками, обработке данных и многом другом.

RU V.5. Упражнения на Pandas

Задача 1. *Напишите программу коды которой позволяют определить максимальное значение среди отрицательных значений массивных данных из таблицы 5.3 (используйте структуру Pandas Series).*

Задача 2*. Предположим, что имеется корзина из 10 разных фруктов с более или менее отличающимися друг от друга ценами: "личи"=67 рублей, "банан"=70 рублей, "авокадо"=156,5 рублей, "кумкват"=105 рублей, "ананас"=170 рублей, "апельсин"=120 рублей, "помело" =105 рублей, "лимон"=144 рубля, "Мандарин"=135,5 рубля, "грейпфрут"=99 рублей. Используйте функцию «sort_index» для сортировки имен фруктов в порядке возрастания и убывания. Отображение результата на экране с различными ценами. Примените параметр «inplace = True» в функцию «sort_values()», какой результат или данные (о структуре Series в Pandas) будут иметь переменные «new_data» и «data_se»?*

Задача 3*. Создайте новый модуль сортировки алгоритмов по выбору и используйте его для упорядочения данных в списке (используйте структуру Pandas Series).*

Задача 4*. Создайте новый модуль пузырькового алгоритма сортировки и используйте его для упорядочения данных в списке (используйте структуру Pandas Series).*

***Задача 5**. Создайте новый модуль сортировки алгоритмов путем вставки и используйте его для упорядочения данных в списке (используйте структуру Pandas Series).*

***Задача 6.** Создайте новый модуль алгоритмов коктейлей или двунаправленного пузырькового алгоритма сортировки и используйте его для упорядочения данных в списке (используйте структуру Pandas Series).*

***Задача 7 (с ответом).** Создайте программные коды, которые позволят сортировать данные в структуре типа «Series» в порядке возрастания и убывания; сохраните результаты в файле формата csv, который можно будет открыть с помощью программного обеспечения Excel (каждая группа данных будет помещена в четко определенный столбец).*

***Задача 8 (с ответом)**. Напишите программные коды, которые позволят читать и записывать данные в файл типа csv (используйте структуру типа «Series» Pandas).*

***Задача 9 (с ответом)**. Создайте программные коды, которые позволят добавить новый столбец в URL-файл, указанный ниже. Информация должна соответствовать следующей формуле (или условию): если значение ячейки столбца « size » больше 2, то значение новой ячейки столбца будет иметь значение « Хорошо », в противном случае оно будет иметь значение « Плохо ». Отобразите результат последних 6 строк на экране.*

url = “https://raw.githubusercontent.com/pandas-dev/pandas/main/pandas/tests/io/data/csv/tips.csv”

Задача 10.

1) Создайте DataFrame, который отражает следующие данные (см. таб. 5.6) двумя разными способами (из словаря «dictionary» и из вектора NumPy), и потом, выберите один из способов для продолжения решения задачи:

Таблица 5.6 – Данные о продукте

№	Наименование товара	Единица измерения	Цена покупки	Цена продажи	Остаток
1	*Рис*	*кг.*	110	125	250
2	*Масло*	*л.*	500	575	120
3	*Сахар*	*кг.*	70	125	300
4	*Мыло*	*шт.*	180	195	120
5	*Вода*	*л.*	60	85	100
6	*Апельсин*	*кг.*	115	150	150
7	*Манго*	*кг.*	165	180	160
8	*Кастрюля*	*шт.*	3500	3805	200
9	Стакан	*шт.*	250	300	350
10	Лимон	*кг.*	90	105	50

2) В предыдущей таблице данных необходимо добавить два столбца с именами «Дата покупки» и «Скидка» соответственно в следующих позициях: сразу после второго столбца и непосредственно перед последним столбцом.

3) Заполните оба столбца более или менее реальными случайными данными.

RU V.6. Ответы на задания в Pandas

Ответ к задаче 7

```
1  import os, csv
2  import pandas as pa
3
4  x = [67, 70, 156.5, 105, 170, 120, 105, 144, 135.5, 99]
5  ind = ["Личи", "Банан", "Авокадо", "Кумкват", "Ананас",
6                    "Апельсин", "Помело", "Лимон", "Мандарин",
7                    "Грейпфрут"]
8
9  data_se = pa.Series(data=x, index=ind)
10
11 os.makedirs('FolderSort', exist_ok=True)
12 new_data = data_se.sort_values(ascending=False)
13 new_data.to_csv('FolderSort/out1.csv', sep=';')
14 new_data = data_se.sort_values(ascending=True, inplace=True)
15 data_se.to_csv('FolderSort/out2_1.csv', sep=';')
16
17 with open ("FolderSort/out1.csv", 'r', newline='') as csv1file:
18    with open ("FolderSort/out2_1.csv", 'a+', newline='') as csv2file:
19       read_csv = csv.reader(csv1file, delimiter=',')
20       for row in read_csv:
21          writer = csv.writer(csv2file)
```

Комментарий

Символ ';' позволяет Excel открыть каждую строку данных в каждом столбце.

« *newline*=" » – позволяет запретить пробел между строками данных.

Создаются два файла: первый (*out1.csv*) имеет результат убывающей сортировки данных, а второй (*out2_1.csv*) для начала имеет

результат возрастающей сортировки данных, а затем добавляется результат убывающей сортировки данных.

Ответ к задаче 8

Запись данных в файл типа csv.

```
1  import csv, os
2
3  os.makedirs('FolderData', exist_ok=True)
4
5  with open ("FolderData/texte_exo8.csv", 'w+', newline=' ',
6                      encoding="Windows-1251") as csv_file8:
7     write_to_csv = csv.writer(csv_file8, delimiter=";")
8     write_to_csv.writerow(["Номер", "Имя", "Код страны",
9                      "Телефон", "Страна"])
10    write_to_csv.writerow(["1", "Иванов", "7", "9103456744",
11                     "Россия"])
12    write_to_csv.writerow(["2", "Манье", "237", "6551579",
13                     "Камерун"])
14    write_to_csv.writerow(["3", "Титто", "39", "3354477333",
15                     "Италия"])
16    write_to_csv.writerow(["4", "Андреенко", "375", "291113322",
17                     "Белоруссия"])
18    write_to_csv.writerow(["5", "Хайли", "86", "1065440011",
19                     "Китай"])
```

Если откроем созданный файл (*texte_exo8.csv*) простым текстовым редактором, структура данных будет похожа на рисунок 5.6, а при открытии с помощью программного обеспечения Excel или OpenOffice calc мы получим рисунки 5.7 и 5.8 соответственно.

```
Номер;Имя;Код страны;Телефон;Страна
1;Иванов;7;9103456744;Россия
2;Манье;237;6551579;Камерун
3;Титто;39;3354477333;Италия
4;Андреенко;375;291113322;Белоруссия
5;Хайли;86;1065440011;Китай
```

Рис. 5.6. Структура файла в формате csv

	A	B	C	D	E
1	Номер	Имя	Код страны	Телефон	Страна
2	1	Иванов	7	9103456744	Россия
3	2	Манье	237	6551579	Камерун
4	3	Титто	39	3354477333	Италия
5	4	Андреенко	375	291113322	Белоруссия
6	5	Хайли	86	1065440011	Китай

Рис. 5.7. Структура файла в формате csv из OpenOffice calc

	A	B	C	D	E
1	Номер	Имя	Код страны	Телефон	Страна
2	1	Иванов	7	9103456744	Россия
3	2	Манье	237	6551579	Камерун
4	3	Титто	39	3354477333	Италия
5	4	Андреенко	375	291113322	Белоруссия
6	5	Хайли	86	1065440011	Китай

Рис. 5.8. Структура файла в формате csv из Excel

Читать и печатать данных из файла типа csv

```
1  import csv
2
3  with open ("FolderData/texte_exo8.csv", newline='',
4                                  encoding="Windows-1251") as csv_file_r:
5      read_csv = csv.DictReader(csv_file_r, delimiter=';')
6      print("Номер", "Имя", "Код страны","Телефон", "Страна")
7      for lign in read_csv:
8          print(lign["Номер"]," ",lign["Имя"]," ",lign["Код страны"]," ",
9                      lign["Телефон"]," ", lign["Страна"])
```

Результат после выполнения кода

Номер	Имя	Код страны	Телефон	Страна
1	Иванов	7	9103456744	Россия
2	Манье	237	6551579	Камерун
3	Титто	39	3354477333	Италия
4	Андреенко	375	291113322	Белоруссия
5	Хайли	86	1065440011	Китай

Ответ к задаче 9

```
1  import pandas as pa
2  import numpy as nu
3
4  url = "https://raw.githubusercontent.com/pandas-
5                      dev/pandas/main/pandas/tests/io/data/csv/tips.csv"
6  data = pa.read_csv(url)
7  data["Новый ст"] = nu.where(data["size"] > 2, "Хорошо",
8                      "Плохо")
9  infos = data.tail(6)
```

Результат после выполнения кода

	total_bill	tip	sex	smoker	day	time	size	Новый ст
238	35.83	4.67	Female	No	Sat	Dinner	3	Хорошо
239	29.03	5.92	Male	No	Sat	Dinner	3	Хорошо
240	27.18	2.00	Female	Yes	Sat	Dinner	2	Плохо
241	22.67	2.00	Male	Yes	Sat	Dinner	2	Плохо
242	17.82	1.75	Male	No	Sat	Dinner	2	Плохо
243	18.78	3.00	Female	No	Thur	Dinner	2	Плохо

RU V.7. Вопросы

1) Как получить информацию о данных и зачем использовать Pandas Python?

2) Какая польза от Pandas, когда у нас есть библиотека NumPy?

3) Каковы основные структуры данных Pandas?

4) В чем разница между Series и DataFrame?

5) Каковы возможности создания структуры Series в Pandas?

6) Каковы возможности создания DataFrame в Pandas?

7) Какие функции уникальны для Series?

8) Какие функции уникальны для DataFrame?

9) Какие функции одновременно характеризуют Series и DataFrame?

10) В чем разница между следующими двумя методами .loc() и .iloc()

11) Каковы различные методы сортировки данных?

12) Какие алгоритмы используют Pandas для сортировки?

13) Как читать файлы с помощью Pandas?

14) Как можно эффективно работать с большими данными?

15) Как ввести данные в файл Excel

16) Можно ли открыть файл в формате csv из простого текстового редактора, не рискуя испортить его структуру, сохранив его снова?

17) Можно ли открыть файл в формате ***xlsx (xls)*** из простого текстового редактора, не рискуя испортить его структуру, сохранив его снова?

Продолжение на странице 465

EN Chapter V. PANDAS – A DATA ANALYSIS TOOL

This library allows you to integrate the functionality of the NumPy and matplotlib packages, providing the user with a convenient tool for analyzing and visualizing various data. The data tables manipulated by Pandas are called data frames, in other words, the data table in Pandas is called a data frame file (DataFrame).

Pandas is a python library that allows you to perform the following roles:

- this makes it easy to manipulate the analyzed data;
- allows manipulation of data tables with labels on columns and rows; easily manipulates unordered data in an ordered form (note that columns can be inserted and removed from data structures during processing);
- allows you to build graphs from the m "matplotlib" library;
- is also used for processing arrays of size 1, 2 or n with built-in indexing;
- provides a wide range of data sources: text files, Excel files, comma–separated (CSV) files, HDF5 files (Hierarchical Data Forma - Hierarchical Data Form), SQL databases and others.

The library requires installation in your Python environment using the command: "pip install pandas". If you have any problems during the installation, it is recommended to consult the following web page for more information:

URL : https://pandas.pydata.org/docs/getting_started/install.html

The data structure in Pandas is presented in 2 different forms:

- as a "Series", that is, in the form of a string (a one-dimensional array (a vector of a row or column));

- under a two-dimensional array, i.e. in the form of a matrix (rows and columns);

EN 5.1. Pandas in «Series»

To use this form of data representation "Series", you need to know that its syntax looks like this:

*Pandas.Series(**data**=None, **index**=None, **dtype**: Dtype | None=None, **name**=None, **copy**: bool=False, **fastpath**: bool=False)*

where data is sequential data in the form of a list;

index is a label used to number each element or value in an array. The number of indexes is equal to the number of elements located in the array, so they must have the same length on both sides. If the Index values are not specified, then by default they will take values from 0 to n, respectively. (n-1 is the total number of values entered);

dtype is the data type required: int, int8, int64, float, float64, object (or str), bool, complex, etc

name is string type(str); this name is optional for "Series".

copy (boolean type, not by default) – allows you to copy input data. For data of the "dictionary" type, the default value "None" behaves like "copy=True".

Example 1:

Suppose we had a piece of paper on which we listed various temperatures in degrees Celsius (C) in Vladimir at a certain time during the week. Let's denote the numbers 0-6 as different days of the week, starting from Sunday. The temperatures obtained are different and are shown in the following Table 5.1:

Table 5.1 – Temperature data in degrees Celsius

0	1	2	3	4	5	6
10C	5C	8C	14C	2C	9C	7C

This data can be entered into the computer using the Pandas functions via "Series".

Numbers from 0 to 6 will be considered indexes, and temperatures will be considered one–dimensional arrays (Series). So, here's what the code from the ***list*** will look like:

```
1  import pandas as pa
2
3  d_s = ["10C", "5C", "8C", "14C", "2C", "9C", "7C"]
4  data_s = pa.Series(data=d_s)
5  print(data_s)
```

The result after executing the program codes

```
0    10C
1     5C
2     8C
3    14C
4     2C
5     9C
6     7C
dtype: object
```

We observe the automatic appearance of numbers in each row from 0 to 6, which represent line numbers and simultaneously different days of the week. Instead of these numbers, we can replace them with the first two letters of the days of the week; that is, Su →0; Mo → 1; Tu→2; We→3; Th→4; Fr→5 and Sa→6. Let's see how to change these numbers by these prefixes of the days of the week using the "index" parameter. To do this, you need to make the following changes to the code :

```
1  import pandas as pa
2
3  d_s = ["10C", "5C", "8C", "14C", "2C", "9C", "7C"]
4  ind = ["Su", "Mo", "Tu", "We", "Th", "Fr", "Sa"]
5  data_s = pa.Series(data=d_s, index=ind)
   print(data_s)
```

The result after executing the program codes.

```
Su     10C
Mo      5C
Tu      8C
We     14C
Th      2C
Fr      9C
Sa      7C
dtype: object
```

Let's try to find the average temperature for the week using these data. To do this, you will need to use the "mean" function.

The problem is that the values should not be of the "object" type, but of the "int" type, for example. Before converting values using the "astype" function, you must first remove the "C" character from the data using the "replace" function. Here's how the code will be presented:

```
1  import pandas as pa
2
3  d_s = ["10C", "5C", "8C", "14C", "2C", "9C", "7C"]
4  ind = ["Su", "Mo", "Tu", "We", "Th", "Fr", "Sa"]
5  data_s = pa.Series(data=d_s, index=ind)
6  trans = data_s.str.replace("C","").astype(int).mean()
7  print("Data:\n %s"%data_s)
8  print("Average temperature = %.3f C"%trans)
```

The result after executing the program codes.

```
Data:
Su     10C
Mo      5C
Tu      8C
We     14C
Th      2C
Fr      9C
Sa      7C
dtype: object
Average temperature = 7.857 C
```

Example 2:

Let's try to change the codes with the same sequential ("Series") data from the "dictionary" with the specified index (days of the week), which we announced in the previous example.

```
1  import pandas as pa
2
3  d_s2 = {"Su": "10C", "Mo":"5C", "Tu":"8C", "We":"14C",
4                  "Th":"2C", "Fr":"9C", "Sa":"7C"}
5  data_s2 = pa.Series(data=d_s2)
6  print("Data:\n %s"%data_s2)
```

The result after executing the program codes.

```
Data:
Su     10C
Mo      5C
Tu      8C
We     14C
Th      2C
Fr      9C
Sa      7C
dtype: object
```

We see that the dictionary keys correspond to the data index values, so if we add new index values using the «index» parameter, there will be no effect.

Example 3:

Let's change the codes again with the same sequential («Series») data of the first example from a one-dimensional massive type «***ndarray***» with a clearly specified index (days of the week). We need to use the «***NumPy***» library, that is, import it.

```
1  import pandas as pa
2  import numpy as np
3
4  d_s3 = np.array(["10C", "5C", "8C", "14C", "2C", "9C", "7C"])
5  ind = ["Su", "Mo", "Tu", "We", "Th", "Fr", "Sa"]
6  data_s3 = pa.Series(data=d_s3, index=ind)
7  print("Data:\n %s"%data_s3)
```

The result after executing the program codes.

```
Данные:
Su     10C
Mo      5C
Tu      8C
We     14C
Th      2C
Fr      9C
Sa      7C
dtype: object
```

With the help of "Series" we can perform many manipulations with data. Let's solve a few examples with certain attributes:

Using the "axes" attribute, we can access the index label and do what we want with them, for example, change the type using "astype", for example, if necessary, put "float".

```
1  ex1 = data_s3.axes[0].astype("float")
2  # or
3  ex1 = data_s3.axes.astype("float")
```

Using the "size" attribute, we can determine the number of elements in the array, and using the "ndim" attribute, we can determine the type of array by its dimension.

```
1  ex1 = data_s3.shape
2  # и
3  ex1 = data_s3.ndim
```

Using the "values" attribute, we can output data without any additional information.

```
1  ex1 = data_s3.values
```

Other attributes (see Table 5.2).

Table 5.2 – Some useful attributes of the "Series" type in Pandas

T	Allows you to return the transposition of an array, which here by definition is equal to itself, because we are dealing with an array of one dimension
array	To expand data arrays (for example, using NumPy)
at	Use a unique array value using a label
dtype	Returns the data type used in the array: int, int8, int32, int64, float, float64, object (or string), bool, complex, or others.
dtypes	Returns the data type used in the array: int, int8, int32, int64, float, float64, object (or string), bool, complex, or others.
hasnans	Returns True if there is at least one cell with the value "NaN"
iat	Get or set the cell value by simply specifying the row or column number
iloc	Indexing solely based on the location of values (integers) for sampling by position. It allows you to select a value belonging to a specific row or column in the dataset
is_monotonic	Returns True if the amount of data in the array increases
is_monotonic_decreasing	Returns True if the data in the array is decreasing

End of table 5.2

is_monotonic_increasing	Same function as "is_monotonic"
is_unique	Returns True if there is only one value in the array
loc	Allows you to access a group of rows and columns by labels or a logical value of the table
name	Returns the name "Series"
nbytes	Returns the number of bytes of data used in the array

So, what features can we use with the «Series» type? Pandas give us several methods or functions for manipulating data in a one-dimensional array. You can get a full list of these functions at the following URL link: [https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.Series.html], among these functions, we will select some of them to demonstrate (using program codes) how to use them. But before that, we will define the new source data in the following table № 5.3 for manipulation, where the first row is the index.

Table 5.3 – Data for processing

0	1	2	3	4	5	6	7	8	9
110	57.4	88.1	-46	213	-75.6	204	-86	103	49.3

How to get the minimum data value among the positive values? To solve this problem, we will use two functions:

*«**mask()**»* – to select data from the condition and

*«**min()**»* – to find this minimum value.

Thus, we get the following code:

```
1  import pandas as pa
2
3  d_se = [110, 57.4, 88.1, -46, 213, -75.6, 204, -86, 103, 49.3]
4  data_se = pa.Series(data=d_se)
5  new_data = data_se.mask(data_se<0).min()
6  print("Min. value among the positive: %.2f"%new_data)
```

The result after executing the program codes

```
Min. value among the positive: 49.30
```

Another additional function or another possibility of using the ***mask()*** *method?*

If we change the code of the 5-th line by this:

```
5  new_data = data_se.mask(data_se<0, 50)
6  print("Data: \n%s"% new_data)
```

As a result, we will replace all negative values with the digit 50.

The result after executing the program codes

```
Data:
0    110.0
1     57.4
2     88.1
3     50.0
4    213.0
5     50.0
6    204.0
7     50.0
8    103.0
9     49.3
dtype: float64
```

*How to return a certain number of n elements in descending or ascending order. In descending order, we will use the «**nlargest()**» function, and in ascending order, the «**nsmallest()**» function.*

Syntax of the *nlargest()* function:

Séries.*nlargest* (*n* = 5 , *keep* = 'first')

The parameter «***keep***» – can take the following values: {'*first*', '*last*', '*all*'}, by default we have the value 'first'.

Parameter description:

n integer: the default value is **5** – it represents the number of values sorted in descending order.

The «***keep***» parameter can take the following values: {'first', 'last', 'all'}, by default it takes the value 'first'.

When certain values are repeated in the list, you can focus on choosing from the following values:

«***first***»: returns the first n elements according to the condition in the order of occurrence.

«***last***»: returns the last n elements according to the condition in the reverse order of occurrence.

«***all***»: saves all elements that can satisfy the condition, which may result in a result larger than **n**.

Example 4:

It is necessary to change the values of the array ***d_se[i]*** (where ***i*** is the index of the array), so that you can visually understand the difference between the parameters of the ***nlargest()*** function from the result; take *n=4* and change the various values of the ***keep*** parameters. For the solution, we have the following code:

```
1  import pandas as pa
2
3  d_se = [110, 57.4, 88.1, 110, 213, 55, 215, 104, -86, 201]
4  data_se = pa.Series(data=d_se)
5  new_data = data_se.nlargest(n=4, keep="first")
6  print("Data: \n%s"%new_data)
```

The result after executing the program codes

```
Data:
6    215.0
4    213.0
9    201.0
0    110.0
dtype: float64
```

If we change the code of line 5 as follows:

```
5  new_data = data_se.nlargest(n=4, keep="last")
```

The result after executing the program codes

```
Data:
6    215.0
4    213.0
9    201.0
3    110.0
dtype: float64
```

If we change the code in line 5 again as follows:

```
5  new_data = data_se.nlargest(n=4, keep="all")
```

The result after executing the program codes

```
Data:
6    215.0
4    213.0
9    201.0
0    110.0
3    110.0
dtype: float64
```

Syntax of the *nsmallest()* function:

```
Series.nsmallest(n=5, keep='first')
```

Description of parameters

Here we have the same parameter description as the previous *nlargest()* function.

Example 5:

```
1  import pandas as pa
2
3  d_se = [110, 57.4, 88.1, 110, 213, 55, 215, 104, -86, 201]
4  data_se = pa.Series(data=d_se)
5  new_data = data_se.nsmallest(n=4, keep="first")
6  print("Data: \n%s"%new_data)
```

The result after executing the program codes

```
Data:
8   -86.0
5    55.0
1    57.4
2    88.1
dtype: float64
```

If we change the code in line 5 again,

```
5  new_data = data_se.nsmallest(keep="all")
```

then we will get 5 lines corresponding to the condition of the *"nsmallest"* function, since n is 5 by default:

The result after executing the program codes

```
Data:
8    -86.0
5     55.0
1     57.4
2     88.1
7    104.0
dtype: float64
```

How else to sort the data in «Series» in another way? To do this, "Pandas" gives us the ability to use the ***«sort_index»*** *function to sort massive indexes and* ***«sort_values»*** *to sort values.*

Syntax of the *«**sort_index**»* function:

Series.***sort_index***(*axis*=0, *level*=None, *ascending*=True, *inplace*=False, *kind*='quicksort', *na_position*='last', *sort_remaining*=True, ignore_index=False, key=None)

The function returns a new array (one-dimensional) sorted by its index, if the *«inplace»* argument has the value "False", otherwise (if "True") it updates the original array and returns the value "None".

Description of parameters

axis – takes the value 0, since there is a type "Series";

level – this is optional; if it is not equal to "None", then it sorts the values by the specified level (or levels) of the index;

ascending – by default, it takes the value "True", so sorting will be performed in ascending order; if the value is "False", then sorting is performed in descending order;

inplace – if it has the value "True", the operation is performed in place;

kind: {'quicksort', 'mergesort', 'heapsort', 'stable'}, by default "quicksort"; with this parameter, we can choose the type of sorting algorithm. See also the topic of sorting for more information (Chapter IV). Note that "mergesort" and "stable" are the only stable algorithms;

na_position: {'first', 'last'}, by default "last"; if we have "first", then it puts the value "NaN" at the beginning, and if "last", it puts the value "NaN" at the end;

sort_remaining: the default value is "True"; it works if this parameter is set to "True" and the indexes are at multiple levels, so sorting will be performed at other index levels as well.;

ignore_index – by default "False"; if the value is "True", the resulting axis will be marked from 0 to n – 1;

key – if it is not the value "None", then you need to use the key function to the index values before sorting.

PS. NaN – Not a Number

Example 6:

Suppose there is a basket of 10 different fruits with more or less different prices from each other: "lychee"= 67 rubles, "banana"=70 rubles, "avocado"=156.5 rubles, "kumquat"=105 rubles, "pineapple"=170 rubles, "orange"=120 rubles, "pomelo" =105 rubles, "lemon" =144 rubles, "Tangerine" =135.5 rubles, "grapefruit" = 99 rubles. Use the «sort_index» function to sort the names of fruits in ascending and descending order. Displaying the result on the screen with different prices.

```
1  import pandas as pa
2
3  d_se = [67, 70, 156.5, 105, 170, 120, 105, 144, 135.5, 99]
4  ind = ["Lychee", "Banana", "Avocado", "Kumquat",
5                  "Pineapple", "Orange", "Pomelo", "Lemon",
6                  "Tangerine", "Grapefruit"]
7  data_se = pa.Series(data=d_se, index=ind)
8  new_data = data_se.sort_index(axis=0, level=None,
9                  ascending=True, kind='stable')
10 print("Names of fruits sorted in ascending order:
11                 \n%s"%new_data)
```

The result after executing the program codes

```
Names of fruits sorted in ascending order:
Avocado      156.5
Banana        70.0
Grapefruit    99.0
Kumquat      105.0
Lemon        144.0
Lychee        67.0
Orange       120.0
Pineapple    170.0
Pomelo       105.0
Tangerine    135.5
dtype: float64
```

To sort in descending order, you need to change the 5th and 7th rows as follows (ascending=False):

```
5  new_data = data_se.sort_index(axis=0, level=None,
6                                 ascending=False, kind='stable')
7  print("Names of fruits sorted in descending order:
8  \n%s"%new_data)
```

The result after executing the program codes

```
Names of fruits sorted in descending order
Tangerine     135.5
Pomelo        105.0
Pineapple     170.0
Orange        120.0
Lychee         67.0
Lemon         144.0
Kumquat       105.0
Grapefruit     99.0
Banana         70.0
Avocado       156.5
dtype: float64
```

Syntax of the sort_values() function:

```
Series.sort_values(axis=0, ascending=True, inplace=False,
kind='quicksort', na_position='last', ignore_index=False, key=None)
```

Allows you to sort data (values) of the "Series" type (list) in ascending or descending order according to criteria determined by the following parameters: *axis*, *ascending*, *inplace*, *na_position*, *ignore_index*, *key* and especially the *kind* parameter, which allows you to direct the algorithm or sorting method: the «*quicksort*» method – "quick sort", "mergesort" – "sort by merge" or "heapsort" – "sort by pyramid or by heap". The algorithm used by default is “quick sort” that is, «*quicksort*».

Description of parameters

Here we have the same description of parameters as the previous *sort_index()* function.

Based on the data from the previous example № 1, we will try to sort fruit prices differently in ascending and descending order.

```
1  import pandas as pa
2
3  x = [67, 70, 156.5, 105, 170, 120, 105, 144, 135.5, 99]
4  ind = ["Lychee", "Banana", "Avocado", "Kumquat",
5                    "Pineapple", "Orange", "Pomelo", "Lemon",
6                    "Tangerine", "Grapefruit"]
7  data_se = pa.Series(data=x, index=ind)
8  new_data = data_se.sort_values()
9  print("Fruit prices sorted in ascending order:
10             \n%s"%new_data)
```

The result after executing the program codes

```
Fruit prices sorted in ascending order:
Lychee          67.0
Banana          70.0
Grapefruit      99.0
Kumquat        105.0
Pomelo         105.0
Orange         120.0
Tangerine      135.5
Lemon          144.0
Avocado        156.5
Pineapple      170.0
dtype: float64
```

We didn't put any parameters in line #8 of the code, which means that the *sort_values()* function uses default values, where *ascending=True* and kind= "quicksort". To get the values in descending order, you need to replace lines 8, 9 and 10 with these codes:

```
8  new_data = data_se.sort_values(ascending=False)
9  print("Fruit prices sorted in descending order:
10                          \n%s"%new_data)
```

The result after executing the program codes

```
Fruit prices sorted in descending order:
Pineapple     170.0
Avocado       156.5
Lemon         144.0
Tangerine     135.5
Orange        120.0
Kumquat       105.0
Pomelo        105.0
Grapefruit     99.0
Banana         70.0
Lychee         67.0
dtype: float64
```

EN V.2. Transferring data from a one-dimensional array (Series) to a file of different formats

With the help of several functions, we can convert data from a one-dimensional array (from the «Series» structure) to other data formats or even transfer this data to files of the type, for example, csv; excel and many others. For more information about these functions, see Table 5.4.

Table 5.4 – Some functions for «Series»

1	to_csv([path_or_buf, sep, na_rep, ...])	Allows you to write an object to a *csv* file
	*Series.to_csv(**path_or_buf**=None, **sep**=',', na_rep='', float_format=None, columns=None, header=True, index=True, index_label=None, mode='w', encoding=None, compression='infer', quoting=None, quotechar='"', line_terminator=None, chunksize=None, date_format=None, doublequote=True, escapechar=None, decimal='.', errors='strict', storage_options=None) ...*	

Continued table 5.4

1	***path_or_buf*** – file-type object, by default the value is None, this value is of type String, (OS implementation: *os.PathLike[str]*), or a file-like object implementing a *write()* function.	
2	to_excel(excel_writer[, sheet_name, na_rep, ...])	Allows you to write an object to an Excel sheet
	Series.to_excel(excel_writer, sheet_name='Sheet1', na_rep='', float_format=None, columns=None, header=True, index=True, index_label=None, startrow=0, startcol=0, engine=None, merge_cells=True, encoding=None, inf_rep='inf', verbose=True, freeze_panes=None, storage_options=None)	
3	to_clipboard([excel, sep])	Allows you to copy an object to a temporary storage (called the clipboard) of the system
	*Series.to_clipboard(excel=True, sep=None, **kwargs)* *By default sep='\t' is a field separator* ***kwargs – these parameters will be passed to the structure DataFrame.to_csv*	
4	to_dict([into])	Allows you to convert data from Series {label→ value}to a dictionary type object (dictionary)
5	to_list() ou tolist()	Returns a list of values
6	to_string([buf, na_rep, float_format, ...])	Displays a string representation from "Series"
7	to_frame([name])	Allows you to convert from "Series" to "DataFrame"
8	to_hdf(path_or_buf, key[, mode, complevel, ...])	Allows you to write data to an HDF5 file (Hierarchical Data Format: Hierarchical Data Format – maximum HDF file size: 2 GB) using HDFStore
	Series.to_hdf(path_or_buf, key, mode='a', complevel=None, complib=None, append=False, format=None, index=True, min_itemsize=None, nan_rep=None, dropna=None, data_columns=None, errors='strict', encoding='UTF-8')	

End of table 5.4

9	to_json([path_or_buf, orient, date_format, ...])	Convert the object to a JSON string – which is short for "JavaScript Object Notation", a format independent of any programming language for data exchange on the Internet.
	Series.to_json(path_or_buf=None, orient=None, date_format=None, double_precision=10, force_ascii=True, date_unit='ms', default_handler=None, lines=False, compression='infer', index=True, indent=None, storage_options=None)	
10	to_latex([buf, columns, col_space, header, ...])	Allows you to display an object in a Latex array (the language and layout system of documents), in a long table or in a nested table.
11	to_sql(name, con[, schema, if_exists, ...])	Allows you to write records stored in an SQL database
	Series.to_sql(name, con, schema=None, if_exists='fail', index=True, index_label=None, chunksize=None, dtype=None, method=None)	
12	to_markdown([buf, mode, index, storage_options])	Allows you to print data from "Series" in "Markdown" format – (light markup language)
13	to_numpy([dtype, copy, na_value])	Allows you to turn into arrays of the "NumPy" type
14	to_xarray()	Returns an array object from a Pandas object.
15	to_period([freq, copy])	Used to convert data from this "Series" object to an index with a certain frequency.
16	to_pickle(path[, compression, protocol, ...])	Allows you to allow serialization of objects in the file.
17	to_timestamp([freq, how, copy])	It is used to convert the period index to the date and time index (Datetime Index)

EN V.3. Other methods using «Series»

In principle, there are many functions that can be used to extend the functionality of data processing in the «Series». There are about a hundred methods in table 5.5.

Table 5.5 – Other «Series» functions

1	abs()
2	add(other[, level, fill_value, axis])
3	add_prefix(prefix)
4	add_suffix(suffix)
5	agg([func, axis])
6	aggregate([func, axis])
7	align(other[, join, axis, level, copy, ...])
8	all([axis, bool_only, skipna, level])
9	any([axis, bool_only, skipna, level])
10	concat (objs, axis = 0, join='outer',...)
11	apply(func[, convert_dtype, args])
12	argmax([axis, skipna])
13	argmin([axis, skipna])
14	argsort([axis, kind, order])
15	asfreq(freq[, method, how, normalize, ...])
16	asof(where[, subset])
17	astype(dtype[, copy, errors])
18	at_time(time[, asof, axis])
19	autocorr([lag])
20	backfill([axis, inplace, limit, downcast])
21	between(left, right[, inclusive])
22	between_time(start_time, end_time[, ...])
23	bfill([axis, inplace, limit, downcast])
24	bool()
25	cat
26	clip([lower, upper, axis, inplace])
27	combine(other, func[, fill_value])
28	combine_first(other)
29	compare(other[, align_axis, keep_shape, ...])

Continuation 1 of table 5.5

30	convert_dtypes([infer_objects, ...])
31	copy([deep])
32	corr(other[, method, min_periods])
33	count([level])
34	cov(other[, min_periods, ddof])
35	cummax([axis, skipna])
36	cummin([axis, skipna])
37	cumprod([axis, skipna])
38	cumsum([axis, skipna])
39	describe([percentiles, include, exclude, ...])
40	diff([periods])
41	div(other[, level, fill_value, axis])
42	divide(other[, level, fill_value, axis])
43	divmod(other[, level, fill_value, axis])
44	dot(other)
45	drop([labels, axis, index, columns, level, ...])
46	drop_duplicates([keep, inplace])
47	droplevel(level[, axis])
48	dropna([axis, inplace, how])
49	dt
50	duplicated([keep])
51	eq(other[, level, fill_value, axis])
52	equals(other)
53	ewm([com, span, halflife, alpha, ...])
54	expanding([min_periods, center, axis, method])
55	explode([ignore_index])
56	factorize([sort, na_sentinel])
57	ffill([axis, inplace, limit, downcast])
58	fillna([value, method, axis, inplace, ...])
59	filter([items, like, regex, axis])
60	first(offset)
61	first_valid_index()
62	floordiv(other[, level, fill_value, axis])
63	ge(other[, level, fill_value, axis])
64	get(key[, default])

Continuation 2 of table 5.5

65	groupby([by, axis, level, as_index, sort, ...])
66	gt(other[, level, fill_value, axis])
67	head([n])
68	hist([by, ax, grid, xlabelsize, xrot, ...])
69	idxmax([axis, skipna])
70	idxmin([axis, skipna])
71	infer_objects()
72	info([verbose, buf, max_cols, memory_usage, ...])
73	interpolate([method, axis, limit, inplace, ...])
74	isin(values)
75	isna()
76	isnull()
77	item()
78	items()
79	iteritems()
80	keys()
81	kurt([axis, skipna, level, numeric_only])
82	kurtosis([axis, skipna, level, numeric_only])
83	last(offset)
84	last_valid_index()
85	le(other[, level, fill_value, axis])
86	lt(other[, level, fill_value, axis])
87	mad([axis, skipna, level])
88	map(arg[, na_action])
89	mask(cond[, other, inplace, axis, level, ...])
90	max([axis, skipna, level, numeric_only])
91	mean([axis, skipna, level, numeric_only])
92	median([axis, skipna, level, numeric_only])
93	memory_usage([index, deep])
94	min([axis, skipna, level, numeric_only])
95	mod(other[, level, fill_value, axis])
96	mode([dropna])
97	mul(other[, level, fill_value, axis])
98	multiply(other[, level, fill_value, axis])
99	ne(other[, level, fill_value, axis])

Continuation 3 of table 5.5

100	nlargest([n, keep])
101	notna()
102	notnull()
103	nsmallest([n, keep])
104	nunique([dropna])
105	pad([axis, inplace, limit, downcast])
106	pct_change([periods, fill_method, limit, freq])
107	pipe(func, *args, **kwargs)
108	plot
109	pop(item)
110	pow(other[, level, fill_value, axis])
111	prod([axis, skipna, level, numeric_only, ...])
112	product([axis, skipna, level, numeric_only, ...])
113	quantile([q, interpolation])
114	radd(other[, level, fill_value, axis])
115	rank([axis, method, numeric_only, ...])
116	ravel([order])
117	rdiv(other[, level, fill_value, axis])
118	rdivmod(other[, level, fill_value, axis])
119	reindex(*args, **kwargs)
120	reindex_like(other[, method, copy, limit, ...])
121	rename([index, axis, copy, inplace, level, ...])
122	rename_axis([mapper, index, columns, axis, ...])
123	reorder_levels(order)
124	repeat(repeats[, axis])
125	replace([to_replace, value, inplace, limit, ...])
126	resample(rule[, axis, closed, label, ...])
127	reset_index([level, drop, name, inplace])
128	rfloordiv(other[, level, fill_value, axis])
129	rmod(other[, level, fill_value, axis])
130	rmul(other[, level, fill_value, axis])
131	rolling(window[, min_periods, center, ...])
132	round([decimals])
133	rpow(other[, level, fill_value, axis])
134	rsub(other[, level, fill_value, axis])

End of table 5.5

135	rtruediv(other[, level, fill_value, axis])		
136	sample([n, frac, replace, weights, ...])		
137	searchsorted(value[, side, sorter])		
138	sem([axis, skipna, level, ddof, numeric_only])		
139	set_axis(labels[, axis, inplace])		
140	set_flags(*[, copy, allows_duplicate_labels])		
141	shift([periods, freq, axis, fill_value])		
142	skew([axis, skipna, level, numeric_only])		
143	shift(...)		
144	sort_index([axis, level, ascending, ...])		
145	sort_values([axis, ascending, inplace, ...])		
146	sparse	147	str
148	squeeze([axis])		
149	std([axis, skipna, level, ddof, numeric_only])		
150	sub(other[, level, fill_value, axis])		
151	subtract(other[, level, fill_value, axis])		
152	sum([axis, skipna, level, numeric_only, ...])		
153	swapaxes(axis1, axis2[, copy])		
154	swaplevel([i, j, copy])	155	tail([n])
156	take(indices[, axis, is_copy])		
157	transform(func[, axis])		
158	transpose(*args, **kwargs)		
159	truediv(other[, level, fill_value, axis])		
160	truncate([before, after, axis, copy])		
161	tz_convert(tz[, axis, level, copy])		
162	tz_localize(tz[, axis, level, copy, ...])		
163	unique()		
164	unstack([level, fill_value])		
165	update(other)		
166	value_counts([normalize, sort, ascending, ...])		
167	var([axis, skipna, level, ddof, numeric_only])		
168	view([dtype])		
169	where(cond[, other, inplace, axis, level, ...])		
170	xs(key[, axis, level, drop_level])		

EN V.4. Pandas in a two-dimensional array – DataFrame

EN V.4.1. Storing and inserting data

How to store data in a table using a Data Frame? The DataFrame has a two-dimensional data structure (see Figure 5.1) that can store data of different types: characters, integers, floating point values, text values, etc. It should also be remembered that each DataFrame column is a «Series» structure (a vertical sequence of values (data)).

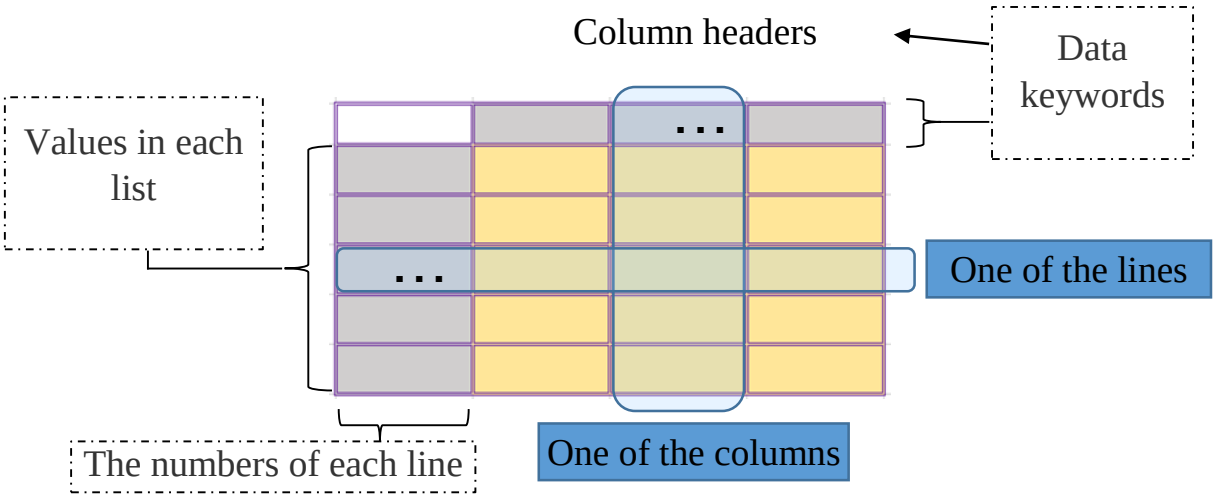


Figure 5.1. Structure of the «DataFrame» table

Example 1

Let's say we have 4 students absent from any class. It is necessary to submit the data of these students to the «DataFrame», knowing that we know their full names (Ivan Ivanov, Dima Petrov, Sasha Sidorov and Irina Panova); their dates of birth (10.02.2000, 23.11.2001, 12.05.1999 and 27.04.2000) and their identity card numbers (233, 632.595 and 810). It is not difficult to notice that these data have different types (text values – symbols,

dates and integers). The challenge is to store this student data using Pandas. How to do it in such a way as to get the result shown in Figure 5.2?

	Name and surnames	Dates of birth	Identity card numbers
0	Ivanov Ivan	10.02.2000	233
1	Petrov Dima	23.11.2001	632
2	Sidorov Sacha	12.05.1999	595
3	Panova Irina	27.04.2000	810

Figure 5.2. «DataFrame» from example №1 (result)

The syntax of «DataFrame» in Pandas looks like this:

DataFrame(data=None, index: Axes | None=None, columns: Axes | None=None, dtype: Dtype | None=None, copy: bool | None=None),

where ***data*** – can contain: data in the form of a "dictionary" - "dictionary", or serial data («Series»), or massive data, or persistent data, or data classes, or list type objects;

index is the label used for the resulting image. By default, it varies from 0 to the number of lines to be output;

columns – names of column labels for different data;

dtype – forced data type (by default None, if it exists, only one "dtype" is allowed): int, int8, int64, object, etc.

copy (logical type, by default none – «None») – allows you to copy data from the input data. Creates a copy of the data array if the copy parameter is set to True. And if *copy = False*, then nothing happens.

Solution: we use a dictionary to build a DataFrame:

```
import pandas as pa

data_p = pa.DataFrame(
    {
        "Name and surnames": [
            "Ivanov Ivan",
            "Petrov Dima",
            "Sidorov Sasha",
            "Panova Irina"
            ],
        "Dates of birth": [
            "10.02.2000",
            "23.11.2001",
            "12.05.1999",
            "27.04.2000"
            ],
        "Identity card numbers": [
            233,
            632,
            595,
            810
            ]
    }
)

print(data_p)
```

The result after executing the program codes

	Name and surnames	Dates of birth	Identity card numbers
0	Ivanov Ivan	10.02.2000	233
1	Petrov Dima	23.11.2001	632
2	Sidorov Sasha	12.05.1999	595
3	Panova Irina	27.04.2000	810

The result after executing the program codes

You can observe the automatic appearance of numbers in each line from 0 to 3, given that there are actually 4 rows of data after executing the previous code. Let's consider how to change these numbers to others, for example, to the labels "a", "b", "c" and "d" using the "index" parameter.

How do I change these numbers?

```
1  import pandas as pa
2
3  d = {
4        "Name and surnames":  ["Ivanov Ivan", "Petrov Dima",
5                               "Sidorov Sasha", "Panova Irina"],
6        "Dates of birth": ["10.02.2000", "23.11.2001", "12.05.1999",
7                           "27.04.2000"],
8        "Identity card numbers": [233, 632,595, 810]
9      }
10
11 data_p2 = pa.DataFrame(data=d, index= ["a","b","c","d"])
12 print(data_p2)
```

The result after executing the program codes

	Name and surnames	Dates of birth	Identity card numbers
a	Ivanov Ivan	10.02.2000	233
b	Petrov Dima	23.11.2001	632
c	Sidorov Sasha	12.05.1999	595
d	Panova Irina	27.04.2000	810

Comments we can make:

The resulting table consists of 4 columns with a label on each column «Name and surnames», «Dates of birth», «Identity card numbers».

In the «Name and surnames» column there are string values; in the «Dates of birth» column there are string values that can be converted to the "Datetime" type, i.e. date and time; in the «Identity card numbers» column there are values of the «int» type – integers.

In tables of the "DataFrame" type, the same column can have values of different types.

We also find that each column is a «Series».

EN V.4.2 Possible operations with a well-defined column

To access a specific column, we can use the following command:

```
Table name [“Label name”]
```

Changes that need to be made to the previous code:

```
11 data_p2 = pa.DataFrame(data=d)
12 col_1 = data_p2["Identity card numbers"]
13 print(col_1)
```

The result after executing the program codes	
0	233
1	632
2	595
3	810

To access an element in this column, we can use the following command:

Example:

```
11  data_p2 = pa.DataFrame(data=d)
12  col_1 = data_p2["Identity card numbers"]
13  elt = col_1[n]
```

col_1 is a variable that will get all the items in the “Identity card numbers” column;

data_p2 is an array of DataFrame;

n is the number of the element in the «Series» (column).

To output or print a column

```
12  col_1 = data_p2["Identity card numbers"]
13  print(col_1[:])
    # or
13  print(col_1)
```

To output the first two elements of the column

```
13  col_1 = data_p2["Identity card numbers"]
14  print(col1[:2])
15
```

To output all the column elements that are after the first one

```
13  col_1 = data_p2["Identity card numbers"]
14  print(col_1[1:])
15
```

To output all the column elements that are after the second

```
13  col_1 = data_p2["Identity card numbers"]
14  print(col_1[2:])
15
```

To find and display the minimum value of a column on the screen

```
13  vmin = data_p2["Identity card numbers"].min()
14  print(vmin)
15
```

To find and display the maximum value of a column on the screen

```
13  vmax =data_p2["Identity card numbers"].max()
14  print(vmax)
15
```

To sort a given column in ascending order in the «DataFrame».

To sort this «DataFrame» by one column, you need to specify the column name using the «by» parameter.

Using the «ascending» parameter, we arrange the sorting order in ascending order (with the value "True") or in descending order (with the value "False"). By default, the «sort_value» function sorts in ascending order.

```
13  tri1 =data_p2.sort_values(by="Identity card numbers",
                              ascending=True)
14  print(tri1)
```

The result after executing the program codes

	Name and surnames	Dates of birth	Identity card numbers
0	Ivanov Ivan	10.02.2000	233
2	Sidorov Sasha	12.05.1999	595
1	Petrov Dima	23.11.2001	632
3	Panova Irina	27.04.2000	810

With the «inplace» parameter, sorting will be performed in the «DataFrame» array in place. The new variable will not be able to get the result of this sorting. The result will remain in the old array, for this the «inplace» parameter must be set to «True».

```
13  data_p2.sort_values(by="Name and surnames", ascending=True,
                        inplace=True)
14  print(data_p2)
```

The result after executing the program codes

	Name and surnames	Dates of birth	Identity card numbers
0	Ivanov Ivan	10.02.2000	233
3	Panova Irina	27.04.2000	810
1	Petrov Dima	23.11.2001	632
2	Sidorov Sasha	12.05.1999	595

To sort a given column in ascending order in the «DataFrame» table.

To sort by several columns of the "DataFrame" table, always using the «sort_values» method, you need to specify the names of the columns that we will use for sorting, separated by commas. After that, with the «ascending» parameter, you will also need to specify the sorting order of columns separated by commas. So, let's try to do this sorting by the columns "Name and surnames" and "Identity card numbers".

```
13 tri2 = data_p2.sort_values(["Name and surnames","Identity card
                numbers"], ascending=[True,True])
14 print(tri2)
```

The result after executing the program codes

	Name and surnames	Dates of birth	Identity card numbers
0	Ivanov Ivan	10.02.2000	233
3	Panova Irina	27.04.2000	810
1	Petrov Dima	23.11.2001	632
2	Sidorov Sasha	12.05.1999	595

This result gives the impression that the last column was not sorted. To make sure everything is working fine, let's try to make some changes to the original «DataFrame» data. Let's change «Sidorov Sasha» to a new «Petrov Dima» with the date of birth on 12.05.1999 and the «Identity card numbers» equal to 595. After this change, executing the previous code will give the following result:

The result after executing the program codes

	Name and surnames	Dates of birth	Identity card numbers
0	Ivanov Ivan	10.02.2000	233
3	Panova Irina	27.04.2000	810
2	Sidorov Sasha	12.05.1999	595
1	Petrov Dima	23.11.2001	632

EN V.4.3 Some operations and interactions between Excel and Pandas

Since Excel, OpenOffice calc and many other spreadsheets are used by many computer users, it is important to know how to integrate or manipulate data from Pandas.

Let's use the data from the following link to find out how to perform different operations. This file contains some data (see Figure 5.3) in *csv* format.

url = https://raw.githubusercontent.com/pandas-dev/pandas/main/pandas/tests/io/data/csv/tips.csv

```
total_bill,tip,sex,smoker,day,time,size
16.99,1.01,Female,No,Sun,Dinner,2
10.34,1.66,Male,No,Sun,Dinner,3
21.01,3.5,Male,No,Sun,Dinner,3
23.68,3.31,Male,No,Sun,Dinner,2
24.59,3.61,Female,No,Sun,Dinner,4
25.29,4.71,Male,No,Sun,Dinner,4
8.77,2.0,Male,No,Sun,Dinner,2
26.88,3.12,Male,No,Sun,Dinner,4
15.04,1.96,Male,No,Sun,Dinner,2
14.78,3.23,Male,No,Sun,Dinner,2
10.27,1.71,Male,No,Sun,Dinner,2
35.26,5.0,Female,No,Sun,Dinner,4
15.42,1.57,Male,No,Sun,Dinner,2
18.43,3.0,Male,No,Sun,Dinner,4
```

Figure 5.3. Data structure in the *csv* file URL

How to read data from a file that is in the URL

```
1  import pandas as pa
2
3  url = "https://raw.githubusercontent.com/pandas-
4                  dev/pandas/main/pandas/tests/io/data/csv/tips.csv"
5  data = pa.read_csv(url)
6  infos = data.head(6)
7  print(infos)
```

The result after executing the program codes

	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
1	10.34	1.66	Male	No	Sun	Dinner	3
2	21.01	3.50	Male	No	Sun	Dinner	3
3	23.68	3.31	Male	No	Sun	Dinner	2
4	24.59	3.61	Female	No	Sun	Dinner	4
5	25.29	4.71	Male	No	Sun	Dinner	4

If instead of line № 6 we put:

```
6  infos = data.tail(5)
```

we will get the last 5 lines of the file.

The result after executing the program codes

	total_bill	tip	sex	smoker	day	time	size
239	29.03	5.92	Male	No	Sat	Dinner	3
240	27.18	2.00	Female	Yes	Sat	Dinner	2
241	22.67	2.00	Male	Yes	Sat	Dinner	2
242	17.82	1.75	Male	No	Sat	Dinner	2
243	18.78	3.00	Female	No	Thur	Dinner	2

A feature of Pandas is that it allows you to directly perform operations on entire columns compared to other spreadsheets. Consider the following example to understand:

Suppose we want to change the data in the [sex] column of the « *tips.csv* » file to the characters « F » and « M ». « F » should replace the words « Female » and « M » with the word «Male». Since we have only two words, we can use the «where» method associated with the «NumPy»

module. While in Excel we will use the «if – then» condition (see Figure 5.4).

```
1  import pandas as pa
2  import numpy as nu
3
4  url = "https://raw.githubusercontent.com/pandas-
5                    dev/pandas/main/pandas/tests/io/data/csv/tips.csv"
6  data = pa.read_csv(url)
7  data["sex"] = nu.where(data["sex"] = = "Female", "F", "M")
8  infos = data.head(5)
9  print(infos)
```

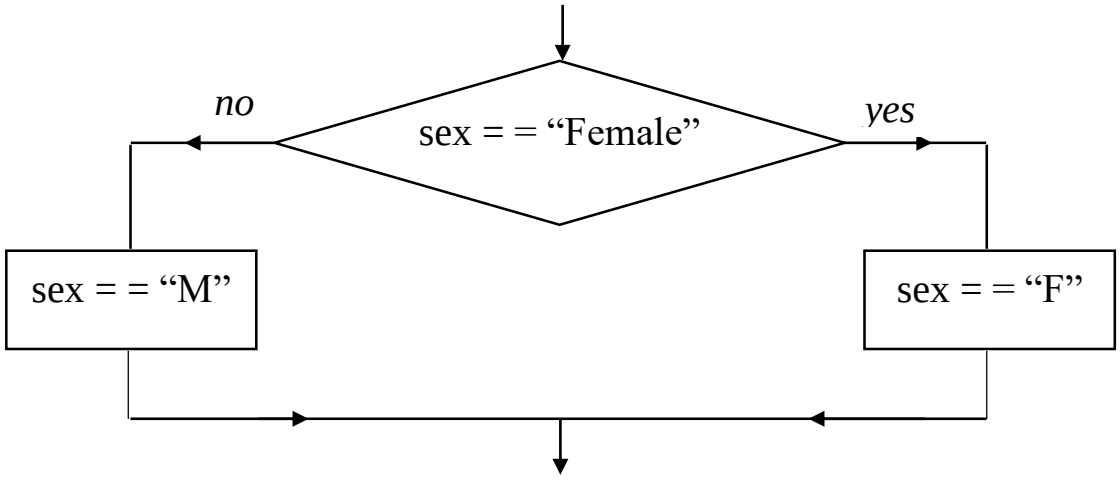


Figure 5.4. Illustration of the «if – then» function

The result after executing the program codes

	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	F	No	Sun	Dinner	2
1	10.34	1.66	M	No	Sun	Dinner	3
2	21.01	3.50	M	No	Sun	Dinner	3
3	23.68	3.31	M	No	Sun	Dinner	2
4	24.59	3.61	F	No	Sun	Dinner	4
5	25.29	4.71	M	No	Sun	Dinner	4

You can delete a column using the « drop » function, rename the column name using the " rename" function, find the maximum value « max », minimum « min », ... of the column. Thus, there are many functions that can be applied to the columns of a file. Using the « columns » function, we can list the names of different columns in the «*DataFrame*».

The codes may look something like these lines:

```
new_value = data["name_colonne"].fonction()
```

```
data = data.drop("column_name ", axis=1)
```

```
infos = data.rename(columns={"old_name": "new_name"}).tail(6)
```

```
infos = data["column_name "].str.find("what_to_find").tail(6)
```

To process strings, we can use existing methods that were described in the previous book -Volume I or in other sources.

Is it possible in Python using the Pandas library to save data in Excel? Of course, yes. To do this, simply use the ***to_excel()*** function with its parameters. The syntax of this function is as follows:

```
DataFrame.to_excel(excel_writer, sheet_name='Sheet1', na_rep='',
float_format=None, columns=None, header=True, index=True,
index_label=None, startrow=0, startcol=0, engine=None,
merge_cells=True, encoding=None, inf_rep='inf', verbose=True,
freeze_panes=None, storage_options=None),
```

where,

excel_writer: the path to the file or the presence of an **ExcelWriter** object;

sheet_name: the name of the sheet that will contain the data (by default '*Sheet1*');

na_rep: representation of missing data;
float_format: string format for floating-point numbers;
columns: the vertical part of the table that will be saved with the data;
header: write the column names. If a list of rows is specified, it is assumed that they are aliases for column names;
index: include (True) or exclude (False) row names (indexes);
index_label: name (label) of the column for the index row;
startrow: x coordinate of the first row of the cell from top to left for writing data;
startcol: the coordinate of the first column of the cell from top to bottom for writing data;
engine: define the engine to use: "*openpyxl*" or "*xlsxwriter*";
merge_cells: writes multi-index and hierarchical rows as merged cells;
encoding: encoding of the resulting Excel file; {"*openpyxl*", }
inf_rep: representation for infinity (Excel does not have its own representation for infinity);
verbose: displays additional information in error logs;
freeze_panes: specifies the lowest row and rightmost column to be "frozen";
storage_options: additional parameters that make sense for a specific storage connection, such as «host», «port», «username», «password», etc.

The «**ExcelWriter**» syntax

ExcelWriter(*path*: FilePathOrBuffer | ExcelWriter, *engine*=None, *date_format*=None, *datetime_format*=None, *mode*: str="w", *storage_options*: StorageOptions=None, *if_sheet_exists*: str | None=None, *engine_kwargs*: dict | None=None)

where,
path: path to a file of type *xls, xls* or *ods*;
engine: engine used for writing;
date_format: allows you to format a string for dates written to Excel files ; (For example, 'YYYY-MM-DD');

datetime_format: used to format a string containing *datetime* objects written to Excel files;

(For example, 'YYYY-MM-DD HH:MM:SS');

mode: file usage mode (write or add -{'w', 'a' or 'r+'}, default 'w');
storage_options: additional options that make sense for a specific storage connection, for example «host», «port», «username», «password», etc.

if_sheet_exists: позволяет направлять запись на уже существующий лист excel (только в режиме добавления) - {'*error*', '*new*', '*replace*', '*overlay*'}, по умолчанию значение равно “error”;

error: выдает ошибку ValueError;

new: создать новый лист с именем, определяемым автоматически;

replace: удаляет содержимое листа перед записью на него нового;

overlay : запись содержимого в существующий лист без удаления старого содержимого;

if_sheet_exists: allows you to direct an entry to an already existing excel sheet (only in the add mode) - {'error', 'new', 'replace', 'overlay'}, by default the value is “error”;

error: throws a ValueError error;

new: create a new sheet with a name determined automatically;

replace: deletes the contents of the sheet before writing a new one to it;

overlay: writing content to an existing sheet without deleting the old content;

engine_kwargs: keyword arguments to pass to engine.

xlsxwriter (recording mode)

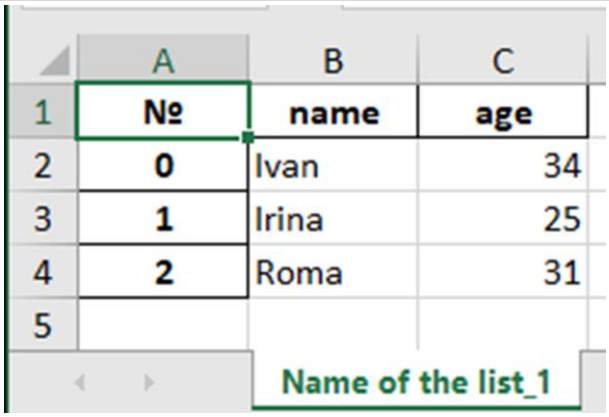
openpyxl (in write and add mode)

orgwriter.

☛ Example 1 of the code for writing data to Excel using the ***to_excel()*** function. **The case when the file (*file_excel1.xlsx*) does not exist**.

```
1  import os
2  import pandas as pa
3  from pandas.io.excel import ExcelWriter
4
5  if __name__ == "__main__":
6      path_file = os.path.dirname(__file__) + r"\file_excel1.xlsx"
7      data_p1 = pa.DataFrame(
8          [['Ivan', 34],
9           ['Irina', 25],
10          ['Roma', 31]],
11         columns=['name', 'age'])
12     with ExcelWriter(path_file,  mode="w") as exemplaire_write:
13         data_p1.to_excel(exemplaire_write, sheet_name="Name of the
14                          list_1",  index_label="№", startrow=0, startcol=0)
15
16
```

The result after executing the program codes



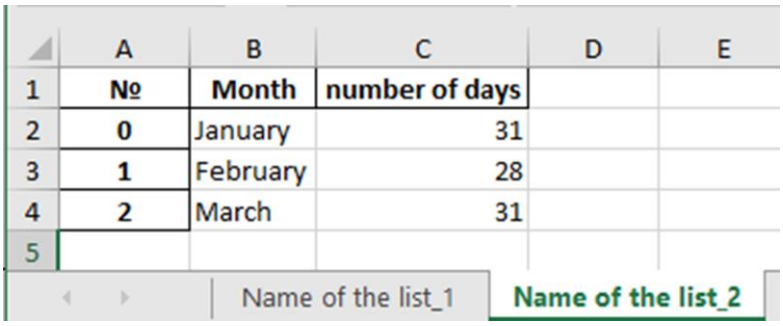
	A	B	C
1	№	name	age
2	0	Ivan	34
3	1	Irina	25
4	2	Roma	31
5			

Name of the list_1

Example 2 of the code for writing data to Excel using the *to_excel()* function. **The case when the file (*file_excel1.xlsx*) exists**, and we want to add a second sheet with other data to the same file. To do this, you need to create an **ExcelWriter** object with the name of the file in question, after specifying the sheet in the file to save.

```
1  import os
2  import pandas as pa
3  from pandas.io.excel import ExcelWriter
4
5  if __name__ == "__main__":
6      path_file = os.path.dirname(__file__) + r"\file_excel1.xlsx"
7      data_p2 = pa.DataFrame(
8          [['January', 31],
9          ['February', 28],
10         ['March', 31]],
11        columns=['Month', 'number of days'])
12     with ExcelWriter(path_file,  mode="a", engine="openpyxl") as
13                         exemplaire_write:
14        data_p2.to_excel(exemplaire_write, sheet_name="Name of the
15                         list_2",  index_label="№", startrow=0, startcol=0)
16
```

The result after executing the program codes



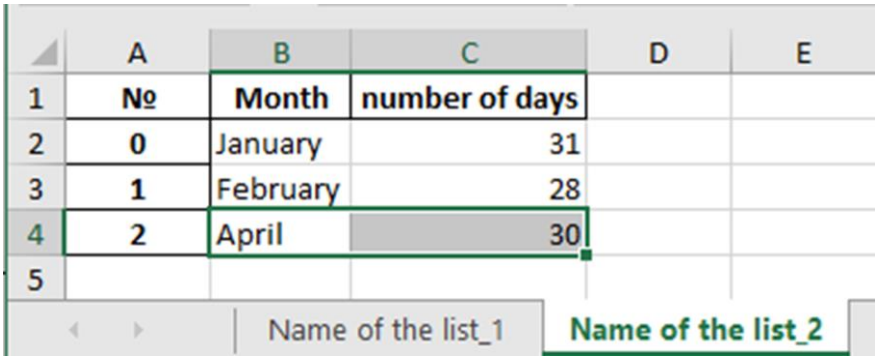
	A	B	C	D	E
1	№	Month	number of days		
2	0	January	31		
3	1	February	28		
4	2	March	31		
5					

Name of the list_1 | Name of the list_2

Example 3 of the code for writing data to Excel using the *to_excel()* function. **How do I update a file** by changing only one information? Let's try to change "March" to "April" with the number of days equal to 30.

```
1  import os
2  import pandas as pa
3  from pandas.io.excel import ExcelWriter
4
5  if __name__ == "__main__":
6      path_file = os.path.dirname(__file__) + r"\file_excel1.xlsx"
7      data_p2 = pa.DataFrame([['April', 30]], columns=None)
8
9      with ExcelWriter(path_file,  mode="a", engine="openpyxl",
       if_sheet_exists="overlay") as exemplaire_write:
10
11         data_p2.to_excel(exemplaire_write, sheet_name="Name of the
12                          list_2", index=False, header=False, startrow=3,
13                          startcol=1)
14
```

The result after executing the program codes



	A	B	C	D	E
1	№	Month	number of days		
2	0	January	31		
3	1	February	28		
4	2	April	30		
5					

Name of the list_1 | Name of the list_2

The main parameter for updating is «***if_sheet_exists***» with the value «***overlay***». With this parameter, you can save old data, erase this old data on one sheet, or automatically create a new sheet to place new data on it.

Archive the file. You can also pack an Excel file into a zip archive. To do this, you need to do the following:

- specify the path where the compressed file will be stored;
- enter data (for example, in a DataFrame) to be stored in an Excel file;
- put the data into an Excel file to be packaged;
- using the « ZipFile » method, you need to compress and package the file.

```
1  import os, zipfile
2  import pandas as pa
3
4  if __name__ == "__main__":
5      path_file1 = os.path.dirname(__file__) + r"\file_zip1.zip"
6      data_p3 = pa.DataFrame(
7          [['January', 31],
8          ['February', 28],
9          ['March', 31]],
10         columns=['Month', 'number of days'])
11
12 with zipfile.ZipFile(path_file1, "w") as zpf:
13     with zpf.open("file_name_excel.xlsx", "w") as data_temp:
14         with pa.ExcelWriter(data_temp) as exemplaire_writer:
15             data_p3.to_excel(exemplaire_writer)
16
```

The result after executing the program codes

There is a file (*file_name_excel.xlsx*) with data compressed in a "*zip*" called: *file_zip1.zip*

Create an Excel document with the following data (see Figure 5.5):

	A	B	C	D
1	№	Name	Surname	Grade
2	1	Petrov	David	3,24
3	2	Ivanov	Ivan	15,02
4	3	Sidorov	Viktor	10,1
5	4	Liskok	Andrei	9,12
6	5	Efimova	Nastia	11,5
7	6	Kuznesov	Kirill	5,76
8	7	Kalinina	Tatiana	18,2
9	8	Zaitseva	Lena	14,11
10	9	Choutov	Maxim	4,97
11	10	Gouceva	Yulia	7,45
12	11	Zemtsova	Alina	8,26
13				

feuille1 feuille2

	A	B	C	D
1	№	Date	Quantity	Religion
2	1	20.02.2014	45	Catholic
3	2	11.08.2003	123	Orthodox
4	3	07.10.2011	23	Protestant
5	4	25.05.2008	56	Muslim
6	5	12.06.2010	201	Catholic
7	6	17.01.2020	67	Orthodox
8	7	02.12.2013	110	Orthodox
9	8	23.03.2015	45	Muslim
10	9	17.07.2003	120	Protestant
11	10	29.10.2018	79	Orthodox
12	11	05.09.2009	88	Pagan
13				

feuille1 feuille2

Figure 5.5. Excel file «*data_excel1.xlsx*» on 2 sheets

The «***read_excel***» method can read local system files with the following extensions: *xls, xlsx, xlsm, xls, odt, ods* and *odt*, or it can still read these files from a URL. The syntax of the ***pandas.read_excel*** function.

*pandas.**read_excel**(io, sheet_name=0, header=0, names=None, index_col=None, usecols=None, squeeze=None, dtype=None, engine=None, converters=None, true_values=None, false_values=None, skiprows=None, nrows=None, na_values=None, keep_default_na=True, na_filter=True, verbose=False, parse_dates=False, date_parser=None, thousands=None, decimal='.', comment=None, skipfooter=0, convert_float=None, mangle_dupe_cols=True, storage_options=None)*

io – the name (path) of the file to read;

sheet_name: it is used to specify the names(s) of the sheets to read. The default value is 0 (first sheet); with a value of 1: second sheet to use; "Sheet1": sheet to use with the name "Sheet1"; if the value is "None", then all electronic sheets will be read.

header – it is used for column labels;

names – it contains a list of column labels used;

index_col – column to use as row labels;

dtype – data type for columns;

usecols – a list of letters or ranges of Excel columns (for example, "A:F" or "A, D, F:J"; other examples: usecols='B:D' or usecols=[0, 2] or usecols=['Nom', 'Grade'];

engine – various manipulation engines: "*xlrd*", "*openpyxl*", "*odf*", "*pyxlsb*";

"xlrd" supports old Excel files with the extension (*.xls*).

"openpyxl" supports new Excel file formats.

"odf" supports OpenDocument file formats (*.odf, .ods, .odt*).

"pyxlsb" supports Excel binaries.

converters –allows you to convert values in specific columns;

skiprows – allows you to exclude one or more lines; for example: skiprows=[1] excludes the second line; skiprows=[2,3] – eliminates the third and fourth lines; skiprows=2 – eliminates the first 2 lines.

How to read data from sheet No. 2 of an Excel file "*data_excel1.xlsx*" in Figure 5.5? Print the first 3 lines on the screen.

```
1  import os
2  import pandas as pa
3
4  path_file = os.path.dirname(__file__) + r"\data_excel1.xlsx"
5  data_p = pa.read_excel(path_file, sheet_name='Sheet2')
6
7  print(data_p.head(3))
8
```

The result after executing the program codes

	№	Date	Quantity	Religion
0	1	2014-02-20	45	Catholic
1	2	2003-08-11	123	Orthodox
2	3	2011-10-07	23	Protestant

In the same way, you can read data from another sheet and perform operations.

We can use a special ***lambda*** function in these operations to facilitate some calculations. For example, if we want to exclude a significant number of lines, it will be less convenient to list all these lines in square brackets than to use the following simple formula: the sixth line of the program will look something like this:

```
6  data_p = pa.read_excel(path_file, sheet_name='Лист1',
7                         skiprows=lambda y: y<3)
```

Instead of the digit 3, there may be another digit of a larger size. This function deletes all the first 3 lines.

For more information about the capabilities of «DataFrame», you can follow the link on the following Pandas website: [*https://pandas.pydata.org/pandas-docs/stable/reference/frame.html*]. We

are talking about: data transformation, statistical calculations, label manipulation, data processing and much more.

EN V.5. Exercises on Pandas

Exercise 1. *Write a code program that allows you to determine the maximum value among the negative values of massive data from Table 5.3 (use the Pandas Series structure).*

Exercise 2. *Suppose there is a basket of 10 different fruits with more or less different prices from each other: "lychee" = 67 rubles, "banana" = 70 rubles, "avocado"=156.5 rubles, "kumquat" =105 rubles, "pineapple"=170 rubles, "orange"=120 rubles, "pomelo" =105 rubles, "lemon" = 144 rubles, "Tangerine" =135.5 rubles, "grapefruit" =99 rubles. Use the "sort_index" function to sort the names of fruits in ascending and descending order. Displaying the result on the screen with different prices. Apply the parameter «inplace = True» to the function «sort_values()», what result or data (about the Series structure in Pandas) will the variables «new_data» and «data_se» have?*

Exercise 3. *Create a new module for sorting algorithms by choice and use it to organize the data in the list (use the Pandas Series structure).*

Exercise 4. *Create a new bubble sorting algorithm module and use it to organize the data in the list (use the Pandas Series structure).*

Exercise 5. *Create a new algorithm sorting module by inserting and use it to organize the data in the list (use the Pandas Series structure).*

Exercise 6. *Create a new module of cocktail algorithms or a bidirectional bubble sorting algorithm and use it to organize the data in the list (use the Pandas Series structure).*

Exercise 7 with the answer. *Create a software program that allows you to sort data in a "Series" type structure in ascending and descending order; save the results in a csv file that can be opened using Excel software (each data group will be placed in a well-defined column).*

Exercise 8 with the answer. *Write program codes that allow you to read and write data to a csv file (use a Pandas "Series" type structure).*

Exercise 9 with the answer. *Create a software program that allows you to add a new column to the URL file specified below. The information must correspond to the following formula (or condition): if the value of the column cell "size" is greater than 2, then the value of the new column cell will have the value "Good", otherwise it will have the value "Bad". Display the result of the last 6 lines on the screen.*

url = “https://raw.githubusercontent.com/pandas-dev/pandas/main/pandas/tests/io/data/csv/tips.csv”

Exercise 10.

1) Create a DataFrame that reflects the following data (see figure 5.6) in two different ways (from the dictionary and from the NumPy vector), and then choose one of the ways to continue solving the problem:

Table 5.6 – Product Information

№	Product Name	Unit of measurement	Purchase price	Sale price	Balance
1	*Rice*	*kg.*	110	125	250
2	*Butter*	*l.*	500	575	120
3	*Sugar*	*kg.*	70	125	300
4	*Soap*	*pcs.*	180	195	120
5	*Water*	*l.*	60	85	100
6	*Orange*	*kg.*	115	150	150
7	*Mango*	*kg.*	165	180	160
8	*Pot*	*pcs.*	3500	3805	200
9	*Glass*	*pcs.*	250	300	350
10	*Lemon*	*kg.*	90	105	50

2) In this previous data table, you need to add two columns named «Purchase Date» and «Discount», respectively, in the following positions: immediately after the second column and immediately before the last column.

3) Fill both columns with more or less real random data.

EN V.6. Answers to exercises on Pandas

Answer to exercise 7

```
1  import os, csv
2  import pandas as pa
3
4  x = [67, 70, 156.5, 105, 170, 120, 105, 144, 135.5, 99]
5  ind = ["Lychee", "Banana", "Avocado", "Kumquat", "Pineapple",
6                  "Orange", "Pomelo", "Lemon", "Tangerine",
7                  "Grapefruit"]
8
9  data_se = pa.Series(data=x, index=ind)
10
11 os.makedirs('FolderSort', exist_ok=True)
12 new_data = data_se.sort_values(ascending=False)
13 new_data.to_csv('FolderSort/out1.csv', sep=';')
14 new_data = data_se.sort_values(ascending=True, inplace=True)
15 data_se.to_csv('FolderSort/out2_1.csv', sep=';')
16
17 with open ("FolderSort/out1.csv", 'r', newline='') as csv1file:
18    with open ("FolderSort/out2_1.csv", 'a+', newline='') as csv2file:
19       read_csv = csv.reader(csv1file, delimiter=',')
20       for row in read_csv:
21          writer = csv.writer(csv2file)
```

Comment

Symbol ';' allows Excel to open each row of data in each column.

« *newline*=" » – allows you to prohibit a space between data lines.

Two files are created: the first (*out1.csv*) has the result of decreasing data sorting, and the second (*out2_1.csv*) first has the result of increasing data sorting, and then the result of decreasing data sorting is added.

Answer to exercise 8

Writing data to a csv file.

```
1  import csv, os
2
3  os.makedirs('FolderData', exist_ok=True)
4
5  with open ("FolderData/texte_exo8.csv", 'w+', newline='',
6                      encoding="utf-8") as csv_file8:
7     write_to_csv = csv.writer(csv_file8, delimiter=";")
8     write_to_csv.writerow(["Number", "Name", "Country Code",
9                      "Phone", "Country"])
10    write_to_csv.writerow(["1", "Ivanov", "7", "9103456744",
11                     "Russia"])
12    write_to_csv.writerow(["2", "Magne", "237", "6551579",
13                     "Cameroon"])
14    write_to_csv.writerow(["3", "Titto", "39", "3354477333", "Italy"])
15    write_to_csv.writerow(["4", "Andreenko", "375", "291113322",
16                     "Bielorussia"])
17    write_to_csv.writerow(["5", "Xaili", "86", "1065440011",
18                     "China"])
```

If we open the created file (*texte_exo8.csv*) with a simple text editor, the data structure will be similar to Figure 5.6, and when opened using Excel or OpenOffice calc software, we will get Figures 5.7 and 5.8, respectively.

```
Number;Name;Country Code;Phone;Country
1;Ivanov;7;9103456744;Russia
2;Magne;237;6551579;Cameroon
3;Titto;39;3354477333;Italy
4;Andreenko;375;291113322;Bielorussia
5;Xaili;86;1065440011;China
```

Fig. 5.6. File structure in csv format

	A	B	C	D	E
1	Number	Name	Country Code	Phone	Country
2	1	Ivanov	7	9103456744	Russia
3	2	Magne	237	6551579	Cameroon
4	3	Titto	39	3354477333	Italy
5	4	Andreenko	375	291113322	Bielorussia
6	5	Pǐaili	86	1065440011	China

Fig. 5.7. File structure in “csv” format from OpenOffice Calc

	A	B	C	D	E
1	Number	Name	Country Code	Phone	Country
2	1	Ivanov	7	9103456744	Russia
3	2	Magne	237	6551579	Cameroon
4	3	Titto	39	3354477333	Italy
5	4	Andreenko	375	291113322	Bielorussia
6	5	Xaili	86	1065440011	China

Fig. 5.7. File structure in "csv" format from Excel

Answer to exercise 9

```
1  import pandas as pa
2  import numpy as nu
3
4  url = "https://raw.githubusercontent.com/pandas-
5                        dev/pandas/main/pandas/tests/io/data/csv/tips.csv"
6  data = pa.read_csv(url)
7  data["new_col"] = nu.where(data["size"] > 2, "Good", "Bad")
   infos = data.tail(6)

   print(infos)
```

The result after executing the program codes

	total_bill	tip	sex	smoker	day	time	size	new_col
238	35.83	4.67	Female	No	Sat	Dinner	3	Good
239	29.03	5.92	Male	No	Sat	Dinner	3	Good
240	27.18	2.00	Female	Yes	Sat	Dinner	2	Bad
241	22.67	2.00	Male	Yes	Sat	Dinner	2	Bad
242	17.82	1.75	Male	No	Sat	Dinner	2	Bad
243	18.78	3.00	Female	No	Thur	Dinner	2	Bad

EN V.7. Questions

1) How to get information about the data and why use Pandas Python?

2) What is the use of Pandas when we have a NumPy library?

3) What are the main Pandas data structures?

4) What is the difference between Series and DataFrame?

5) What are the different possibilities of creating a Series structure in Pandas?

6) What are the different possibilities of creating a DataFrame in Pandas?

7) Which features are unique to the Series?

8) What functions are unique to DataFrame?

9) Which functions simultaneously belong to Series and DataFrame?

10) What is the difference between the following two methods: ***.loc()*** and ***.iloc()***

11) What are the different data sorting methods?

12) What algorithms do Pandas use for sorting?

13) How to read files using Pandas?

14) How can we work effectively with big data?

15) How to enter data into an Excel file

16) Is it possible to open a csv file from a simple text editor without risking spoiling its structure by saving it again?

17) Is it possible to open a file in ***xlsx (xls)*** format from a simple text editor without risking spoiling its structure by saving it again?

Continued on page 466


FR Chapitre V. PANDAS – OUTIL D'ANALYSE DE DONNÉES

Cette bibliothèque permet d'intégrer les fonctionnalités des paquets NumPy et matplotlib, pour donner à l'utilisateur un outil pratique afin de pouvoir analyser et visualiser les différentes données. Les tableaux de données que manipulent Pandas sont appelés DataFrames, en d'autres termes, une table de données dans les pandas est nommé comme un fichier DataFrame (fichier de bloc de données).

Pandas est une bibliothèque de python qui permet jouer les rôles suivants:

- permet de manipuler facilement des données à analyser;
- permet de manipuler des tableaux de données avec des étiquettes sur les colonnes et lignes; manipule facilement les données désordonnées sous une forme ordonnée (Notons que les colonnes peuvent être insérées et supprimées des structures de données pendant le traitement);
- permet de tracer des graphes à partir de la librairie matplotlib ;
- permet de manipuler aussi les massives à dimension 1, 2 ou n (plusieurs) avec indexation intégrée;
- fournit un large choix de sources de données: fichiers texte, fichiers de type Excel, fichiers CSV(Comma-Separated Values), des fichiers au format HDF5 (Hierarchical Data Forma), bases de données SQL, et d'autres.

La bibliothèque nécessite une installation dans votre environnement de Python à travers la commande: « *pip install pandas* ». Si vous rencontrez des difficultés lors de l'installation, il est recommandé de consulter la page web suivante pour plus d'informations:

URL : https://pandas.pydata.org/docs/getting_started/install.html

La structure des données dans Pandas est constamment représentée sous 2 formes différentes:

- par série c'est-à-dire en ligne (tableau à une dimension (ligne ou colonne));

- par massive bidimensionnel c'est-à-dire sous forme de tableau à 2 dimensions (lignes et colonnes);

FR 5.1. Pandas en série (Series)

Pour la forme «Series», la syntaxe est la suivante:

*Pandas.Series(**data**=None, **index**=None, **dtype**: Dtype | None=None, **name**=None, **copy**: bool=False, **fastpath**: bool=False)*

où ***data*** – sont des données en séries sous forme de liste;

index – est le label utilisé pour numéroter chaque élément ou valeur dans la massive. Le nombre d'index est égale au nombre d'éléments rangés dans la massive donc ils doivent avoir la même longueur de part et d'autres. Si les valeurs d'index ne sont pas indiquées, alors elles prendrons par défaut respectivement les valeurs de 0 à n. (n-1 est le nombre total de valeurs saisies);

dtype - type de données voulu: int, int8, int64, float, float64, object (ou str), bool, complex etc.

name – Nom de type chaîne(str); ce nom à donner à la série est facultatif.

copy (de type booléen, par défaut aucun) – permet de copiez les données des entrées. Pour les données de type «dictionnaire», la valeur par défaut «None» se comporte comme *copy=True*.

Exemple 1:

Considérons que nous avions un feuille de papier où nous avons énuméré pendant une semaine les différentes températures en degré Celsius (C) dans la ville de Vladimir à une heure bien déterminée. Désignons les chiffres 0–6 comme étant les différents jours de la semaine, en commençant par dimanche. Les températures obtenues sont différentes et se trouvent dans le tableau 5.1 suivant:

Tableau 5.1 – Données de températures en degré Celsius

0	1	2	3	4	5	6
10C	5C	8C	14C	2C	9C	7C

Ces données peuvent être introduites dans l'ordinateur en utilisant les fonctions de Pandas via «Series».

Les chiffres de 0 à 6 seront considérés comme des index, et les températures dans une massive à une dimension (Series). Ainsi, voila comment le code se présentera à partir d'une ***liste***:

```
1  import pandas as pa
2
3  d_s = ["10C", "5C", "8C", "14C", "2C", "9C", "7C"]
4  data_s = pa.Series(data=d_s)
5  print(data_s)
```

Résultat après exécution du code

```
0    10C
1     5C
2     8C
3    14C
4     2C
5     9C
6     7C
dtype: object
```

Nous observons l'apparition automatique des numéros de chaque ligne de 0 à 6, qui représentent les numéros des lignes et en même tant les différentes journées de la semaine. À la place de ces numéros, nous pouvons les remplacer par les 2 premières lettres des jours de la semaine ; c'est-à-dire que di →0 ; lu →1 ; ma→2 ; me→3 ; je→4 ; ve→5 et sa→6. Voyons comment modifier ces numéros par ces préfixes des jours de la semaine avec le paramètre «index». Pour cela, il faudra apporter les modifications suivantes dans le code :

```
1  import pandas as pa
2
3  d_s = ["10C", "5C", "8C", "14C", "2C", "9C", "7C"]
4  ind = ["di", "lu", "ma", "me", "je", "va", "sa"]
5  data_s = pa.Series(data=d_s, index=ind)
   print(data_s)
```

Résultat après exécution du code

```
di    10C
lu     5C
ma     8C
me    14C
je     2C
va     9C
sa     7C
dtype: object
```

Essayons de trouver la valeur moyenne de température dans la semaine avec ces données. Pour cela, il faudra utiliser la fonction «mean».

Le problème qui se pose est que les valeurs ne doivent pas être de type «*object*», mais de type «*int*» par exemple. Avant la conversion des valeurs par la fonction «*astype*», il faudra d'abord retirer le symbole «C» aux données grâce à fonction «*replace*». Voila comment le code sera représenté:

```
1  import pandas as pa
2
3  d_s = ["10C", "5C", "8C", "14C", "2C", "9C", "7C"]
4  ind = ["di", "lu", "ma", "me", "je", "va", "sa"]
5  data_s = pa.Series(data=d_s, index=ind)
6  trans = data_s.str.replace("C","").astype(int).mean()
7  print("les données:\n %s"%data_s)
8  print("La température moyenne est = %.3f C"%trans)
```

Résultat après exécution du code

```
les données:
di      10C
lu       5C
ma       8C
me      14C
je       2C
va       9C
sa       7C
dtype: object
La température moyenne est = 7.857 C
```

Exemple 2 :

Essayons de modifier les codes avec les mêmes données séquentielles (en série) à partir d'un «***dictionnaire***» avec un index spécifié (les jours de la semaine) que nous avons annoncé à l'exemple précédent.

```
1  import pandas as pa
2
3  d_s2 = {"di": "10C", "lu":"5C", "ma":"8C", "me":"14C",
4                       "je":"2C", "va":"9C", "sa":"7C"}
5  data_s2 = pa.Series(data=d_s2)
6  print("les données:\n %s"%data_s2)
```

Résultat après exécution du code

```
les données:
di      10C
lu       5C
ma      8C
me     14C
je       2C
va      9C
sa       7C
dtype: object
```

Nous constatons que les clés du dictionnaire correspondent aux valeurs d'index des données, par conséquent si nous ajoutons de nouvelles valeurs d'index en utilisant le paramètre «index» , il n' y aura aucun effet.

Exemple 3 :

Modifions de nouveau les codes avec les mêmes données séquentielles (en série) du premier exemple à partir d'un massive à une dimension du type «***ndarray***» avec un index bien spécifié (les jours de la semaine). Nous devons faire appel à la bibliothèque «***NumPy***» c'est-à-dire l'importer .

```
1  import pandas as pa
2  import numpy as np
3
4  d_s3 = np.array(["10C", "5C", "8C", "14C", "2C", "9C", "7C"])
5  ind = ["di", "lu", "ma", "me", "je", "va1", "sa"]
6  data_s3 = pa.Series(data=d_s3, index=ind)
7  print("les données:\n %s"%data_s3)
```

Résultat après exécution du code

```
les données:
di      10C
lu       5C
ma       8C
me      14C
je       2C
va       9C
sa       7C
dtype: object
```

Avec la «Series», nous pouvons faire beaucoup de manipulations sur les données. Réalisons quelques exemples avec certains attributs:

Avec l'attribut «axes» nous pouvons s'adresser à l'étiquette des index et faire d'eux ce que nous voulons, par exemple, changer de type par «astype», en mettant «float» par exemple s'il convient.

```
1  ex1 = data_s3.axes[0].astype("float")
2  # ou
3  ex1 = data_s3.axes.astype("float")
```

Avec l'attribut «size» nous pouvons déterminer le nombre d'élément dans la massive et par l'attribut «ndim» le type de massive à travers sa dimension.

```
1  ex1 = data_s3.shape
2  # et
3  ex1 = data_s3.ndim
```

Avec l'attribut «values» nous pouvons faire ressortir les données sans la moindre information supplémentaire.

```
1  ex1 = data_s3.values
```

Les autres attributs (voir le tableau 5.2)

Tableau 5.2 – Quelques attributs utiles du type «series» dans Pandas

T	Permet de renvoyer la transposition de la massive, qui par définition ici est égale à elle même car nous avons affaire à une massive d'une dimension.
array	Pour l'extension des massives sur les données (par exemple avec NumPy)
at	Utiliser une valeur unique de la massive en se servant de l'étiquette
dtype	Renvoie le type utilisé de données dans la massive: int, int8, int32, int64, float, float64, object (ou str), bool, complex ou autres
dtypes	Renvoie le type utilisé de données dans la massive: int, int8, int32, int64, float, float64, object (ou str), bool, complex ou autres
hasnans	Renvoie la valeur True s'il y a au moins une cellule avec la valeur «NaN»
iat	Obtenir ou fixer la valeur d'une cellule en indiquant juste le numéro de la ligne ou colonne
iloc	Indexation purement basée sur l'emplacement des valeurs (nombres entiers) pour la sélection par position. Il permet de sélectionner une valeur qui appartient à une ligne ou colonne particulière dans un ensemble de données
is_monotonic	Renvoie la valeur True si les données de la massive sont croissantes
is_monotonic_decreasing	Renvoie la valeur True si les données de la massive sont décroissantes

Fin du tableau 5.2

is_monotonic_increasing	Même fonction que « *is_monotonic* »
is_unique	Renvoie la valeur True s'il n'y a qu'une seule valeurs dans la massive
loc	Permet d'accéder à un groupe de lignes et de colonnes par étiquette(s) ou un booléen tableau
name	Renvoie le nom de la série
nbytes	Renvoie le nombre d'octets des données utilisées dans la massive

Alors, quelles sont les possibilités que nous pouvons utiliser avec le type «Series»? Pandas nous donne plusieurs méthodes ou fonctions pour manipuler les données dans la massive à une dimension («Series»). Vous pouvez avoir la liste complète de ces fonctions sur le lien URL suivant: [https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.Series.html], parmi ces fonctions, nous allons en choisir certaines pour démontrer (à l'aide de codes de programme) comment les utiliser. Mais avant cela, définissons les nouvelles données dans le tableau № 5.3 suivant pour manipulation, où la première ligne est l'index.

Tableau 5.3 – Données pour manipulation

0	1	2	3	4	5	6	7	8	9
110	57.4	88.1	-46	213	-75.6	204	-86	103	49.3

Comment obtenir la valeur minimale des données parmi les valeurs positives? Pour résoudre cet exercice, nous allons utiliser deux fonctions:

*«**mask()**» – pour choisir les données à partir d'une condition et*

*«**min()**» – pour déterminer cette valeur minimale.*

Ainsi, nous aurons le code suivant:

```
1  import pandas as pa
2
3  d_se = [110, 57.4, 88.1, -46, 213, -75.6, 204, -86, 103, 49.3]
4  data_se = pa.Series(data=d_se)
5  new_data = data_se.mask(data_se<0).min()
6  print("Valeur minimale positive: %.2f"%new_data)
```

Résultat après exécution du code

```
Valeur minimale positive: 49.30
```

Une autre fonction supplémentaire ou une autre possibilité d'appliquer la méthode mask()?

Si nous changeons le code de la 5-ième ligne par ceci:

```
5  new_data = data_se.mask(data_se<0, 50)
6  print("Les données: \n%s"% new_data)
```

Nous obtiendrons comme résultat le remplacement de toutes les valeurs négatives par le chiffre 50.

Résultat après exécution du code

```
Les données:
0    110.0
1     57.4
2     88.1
3     50.0
4    213.0
5     50.0
6    204.0
7     50.0
8    103.0
9     49.3
dtype: float64
```

*Comment renvoyer un certain nombre **n** d'éléments par ordre décroissant ou par ordre croissant. Par ordre décroissant, nous utiliserons la fonction nlargest() et par ordre croissant, nous utiliserons la fonction nsmallest().*

La syntaxe de la fonction *nlargest()* :

Séries.*nlargest* (*n* = 5 , *keep* = 'first')

Description des paramètres

n entier, par défaut est égale à 5 – elle représente le nombre de valeurs triées par ordre décroissant.

Le parametre «***keep***» – peut prendre les valeurs suivantes : {'*first*', '*last*', '*all*'}, par défaut nous avons la valeur '*first*'.

Lorsque dans une liste, certaines valeurs se répètent, il est possible d'orienter le choix à partir des valeurs suivantes:

«***first***»: renvoie les ***n*** premières éléments selon la condition par ordre d'apparition.

«***last***»: renvoie les ***n*** dernières éléments selon la condition dans l'ordre inverse d'apparition.

«***all***»: conserve toutes les éléments qui peuvent satisfaire la condition, ce qui peut entraîner à un résultat de taille supérieure à ***n***.

Exemple 4:

Changeons les valeurs de la massive ***d_se[i]*** (où ***i*** – index de la massive) dans le but de vous faire comprendre visuellement la différence entre les paramètres de la fonction nlargest() à partir du résultat; prenons n=4 et varions les différentes valeurs du paramètres keep. Nous obtenons le code suivant:

```
1  import pandas as pa
2
3  d_se = [110, 57.4, 88.1, 110, 213, 55, 215, 104, -86, 201]
4  data_se = pa.Series(data=d_se)
5  new_data = data_se.nlargest(n=4, keep="first")
6  print("les données: \n%s"%new_data)
```

Résultat après exécution du code

```
les données:
6    215.0
4    213.0
9    201.0
0    110.0
dtype: float64
```

Si nous changeons le code de la ligne 5 par ceci :

```
5  new_data = data_se.nlargest(n=4, keep="last")
```

Résultat après exécution du code

```
les données:
6    215.0
4    213.0
9    201.0
3    110.0
dtype: float64
```

Si nous changeons de nouveau le code de la ligne 5 par ceci :

```
5  new_data = data_se.nlargest(n=4, keep="all")
```

Résultat après exécution du code

```
les données:
6    215.0
4    213.0
9    201.0
0    110.0
3    110.0
dtype: float64
```

La syntaxe de la fonction *nsmallest()*:

```
Series.nsmallest(n=5, keep='first')
```

Description des paramètres

Nous avons ici la même description des paramètres que la fonction précédente *nlargest()*.

Exemple 5:

```
1  import pandas as pa
2
3  d_se = [110, 57.4, 88.1, 110, 213, 55, 215, 104, -86, 201]
4  data_se = pa.Series(data=d_se)
5  new_data = data_se.nsmallest(n=4, keep="first")
6  print("les données: \n%s"%new_data)
```

Résultat après exécution du code

```
les données:
8   -86.0
5    55.0
1    57.4
2    88.1
dtype: float64
```

Si nous changeons de nouveau le code de la ligne 5,

```
5  new_data = data_se.nsmallest(keep="all")
```

alors nous obtiendrons 5 lignes correspondant à la condition de la fonction «*nsmallest*», car ***n*** par défaut est égale à 5:

Résultat après exécution du code
les données: 8 -86.0 5 55.0 1 57.4 2 88.1 7 104.0 dtype: float64

*Comment encore faire le tri des données sur les «Series» ? Pour cela, «Pandas» nous donne la possibilité d'utiliser la fonction «**sort_index**» pour trier les index de la massive et «**sort_values**» pour trier les valeurs.*

La syntaxe de la fonction *sort_index()*:

Series.***sort_index***(*axis*=0, *level*=None, *ascending*=True, *inplace*=False, *kind*='quicksort', *na_position*='last', *sort_remaining*=True, ignore_index=False, key=None)

La fonction renvoie une nouvelle massive (à une dimension) triée par son index dans le cas où l'argument «*inplace*» a la valeur «*False*», dans la cas contraire (si «*True*»), elle met à jour la massive d'origine et renvoie la valeur «*None*».

Description des paramètres

axis – prend la valeur 0 car nous avons la Série ;

level – il est facultatif; s'il n'est pas égal à «None», alors elle trie les valeurs dans le(s) niveau(x) d'index spécifié(s) ;

ascending – par défaut, il prend la valeur «True», ainsi le trie sera fait par ordre croissant; si «False», le trie est décroissant ;

inplace – s'il a la valeur «True», alors l'opération se passe sur place ;

kind: {'quicksort', 'mergesort', 'heapsort', 'stable'}, par défaut «quicksort»; à travers ce paramètre, nous pouvons choisir le type d'algorithme de tri. Voir aussi le thème sur le tri pour plus d'informations. Notons que «mergesort» et «stable» sont les seuls algorithmes stables ;

na_position: {'first', 'last'}, par défaut «last»; Si nous avons «first», alors il met «NaN» au début, et si «last», il met «NaN» à la fin ;

sort_remaining: par défaut est «True»; Il fonctionne dans le cas où ce paramètre a la valeur «True» et que les index sont à plusieurs niveaux, ainsi le tri se fera aussi dans d'autres niveaux de l'index ;

ignore_index – par défaut est «False»; Si la valeur est «True», l'axe résultant sera étiqueté de 0 à n – 1 ;

key – si ce n'est pas «None», il est necessaire d'utiliser la fonction *key* aux valeurs d'index avant le tri.

NB. NaN: Not a Number – ce n'est pas un nombre.

Exemple 6:

Considérons que nous avons un panier de 10 différents fruits avec des prix plus ou moins différents l'un de l'autre: "Litchi"=67 roubles, "Banane"=70 roubles, "Avocat"=156.5 roubles, "Kumquat"=105 roubles, "Ananas"=170 roubles, "Orange"=120 roubles, "Pomelo"=105 roubles, "Citron"=144 roubles, "Mandarin"=135.5 roubles, "Pamplemousse"=99 roubles. Utiliser la fonction «sort_index» pour faire le tri des noms des fruits par ordre croissant et décroissant. Afficher le résultat sur l'écran avec les différents prix.

```
1  import pandas as pa
2
3  d_se = [67, 70, 156.5, 105, 170, 120, 105, 144, 135.5, 99]
4  ind = ["Litchi", "Banane", "Avocat", "Kumquat", "Ananas",
5                  "Orange", "Pomelo", "Citron", "Mandarin",
6                  "Pamplemousse"]
7  data_se = pa.Series(data=d_se, index=ind)
8  new_data = data_se.sort_index(axis=0, level=None,
9                  ascending=True, kind='stable')
10 print("Les noms des fruits triés par ordre croissant:
11                 \n%s"%new_data)
```

Résultat après exécution du code

```
Les fruits triés par ordre croissant:
Ananas          170.0
Avocat          156.5
Banane           70.0
Citron          144.0
Kumquat         105.0
Litchi           67.0
Mandarin        135.5
Orange          120.0
Pamplemousse     99.0
Pomelo          105.0
dtype: float64
```

Pour faire le tri par ordre décroissant, il faut modifier la 5-ième et la 7-ième ligne comme suite (ascending=False):

```
5  new_data = data_se.sort_index(axis=0, level=None,
6                  ascending=False, kind='stable')
7  print("Les noms des fruits triés par ordre décroissant:
8  \n%s"%new_data)
```

Résultat après exécution du code

```
Les noms des fruits triés par ordre décroissant:
Pomelo          105.0
Pamplemousse     99.0
Orange          120.0
Mandarin        135.5
Litchi           67.0
Kumquat         105.0
Citron          144.0
Banane           70.0
Avocat          156.5
Ananas          170.0
dtype: float64
```

La syntaxe de la fonction *sort_values()*:

```
Series.sort_values(axis=0, ascending=True, inplace=False,
kind='quicksort', na_position='last', ignore_index=False, key=None)
```

Permet de trier les données (les valeurs) d'une série (liste) dans l'ordre croissant ou décroissant selon les critères dictés par les paramètres suivants: *axis*, *ascending*, *inplace*, *na_position*, *ignore_index*, *key* et surtout du paramètre *kind* qui permet d'orienter l'algorithme ou la méthode pour faire le tri : méthode "*quicksort*" – «tri Rapide», "*mergesort*" – «Tri par fusion» ou "heapsort" – «tri Pyramidal ou par tas». L'algorithme utilisé par défaut est «*quicksort*».

Description des paramètres

Nous avons ici la même description des paramètres que la fonction précédente *sort_index()*.

À partir des données de l'exemple №1 précédent, essayons de différents manières de trier les prix des fruits par ordre croissant et décroissant.

```
1  import pandas as pa
2
3  x = [67, 70, 156.5, 105, 170, 120, 105, 144, 135.5, 99]
4  ind = ["Litchi", "Banane", "Avocat", "Kumquat", "Ananas",
5                    "Orange", "Pomelo", "Citron", "Mandarin",
6                    "Pamplemousse"]
7  data_se = pa.Series(data=x, index=ind)
8  new_data = data_se.sort_values()
9  print("Les prix des fruits triés par ordre croissant:
10                   \n%s"%new_data)
```

Résultat après exécution du code

```
Les fruits triés par ordre croissant:
Litchi          67.0
Banane          70.0
Pamplemousse    99.0
Kumquat        105.0
Pomelo         105.0
Orange         120.0
Mandarin       135.5
Citron         144.0
Avocat         156.5
Ananas         170.0
dtype: float64
```

À la ligne 8 du code, nous n'avons mis aucun paramètre, cela veut dire que la fonction *sort_values()* utilise les valeurs par défaut où *ascending=True* et *kind="quicksort"*. Pour obtenir les valeurs par ordre décroissant, il faudra remplacer la ligne 8, 9 et 10 par ces codes:

```
8  new_data = data_se.sort_values(ascending=False)
9  print("Les prix des fruits triés par ordre décroissant:
10                                   \n%s"%new_data)
```

Résultat après exécution du code
Les prix des fruits triés par ordre décroissant: Ananas 170.0 Avocat 156.5 Citron 144.0 Mandarin 135.5 Orange 120.0 Kumquat 105.0 Pomelo 105.0 Pamplemousse 99.0 Banane 70.0 Litchi 67.0 dtype: float64

FR V.2. Transfert de données d'un tableau unidimensionnel (Series)vers un fichier à différents formats

À travers plusieurs fonctions, nous pouvons convertir les données du tableau unidimensionnel (de structure «Series») en d'autres formats données ou même il est possible de transférer ces données dans des fichiers de type par exemple csv, excel et bien d'autres. Pour en savoir plus sur ces fonctions, referons nous au tableau 5.4.

Tableau 5.4 – Quelques fonctions sur les «Series»

1	to_csv([path_or_buf, sep, na_rep, ...])	Permet d'écrire un objet dans un fichier de type *csv*
	*Series.to_csv(**path_or_buf**=None, **sep**=',', na_rep='', float_format=None, columns=None, header=True, index=True, index_label=None, mode='w', encoding=None, compression='infer', quoting=None, quotechar='"', line_terminator=None, chunksize=None, date_format=None, doublequote=True, escapechar=None, decimal='.', errors='strict', storage_options=None) ...*	

Suite tableau 5.4

1	***path_or_buf*** *– objet de type fichier,par défaut la valeur est None, cette valeur est de type String, (implémentation du système d'exploitation: os.PathLike[str]), ou objet de type fichier implémentant une fonction write().*	
2	to_excel(excel_writer[, sheet_name, na_rep, ...])	Permet d'écrire un objet dans une feuille Excel
	Series.to_excel(excel_writer, sheet_name='Sheet1', na_rep='', float_format=None, columns=None, header=True, index=True, index_label=None, startrow=0, startcol=0, engine=None, merge_cells=True, encoding=None, inf_rep='inf', verbose=True, freeze_panes=None, storage_options=None)	
3	to_clipboard([excel, sep])	Permet de copier l'objet dans un espace de stockage temporaire (appelé le presse-papiers) du système
	*Series.to_clipboard(excel=True, sep=None, **kwargs)* *Par défaut sep='\t' , c'est le délimiteur de champ.* ***kwargs – ces paramètres seront transmis à DataFrame.to_csv.*	
4	to_dict([into])	Permet de convertir une série en {label→ value} objet de type dictionnaire
5	to_list() ou tolist()	Permet de renvoyer une liste des valeurs
6	to_string([buf, na_rep, float_format, ...])	Rend une représentation sous forme de chaîne de la série
7	to_frame([name])	Permet de transformer de Série en DataFrame
8	to_hdf(path_or_buf, key[, mode, complevel, ...])	Permet d'écrire les données contenues dans un fichier de type HDF5(Hierarchical Data Format – taille maximale d'un fichier HDF : 2 Go) à l'aide de HDFStore
	Series.to_hdf(path_or_buf, key, mode='a', complevel=None, complib=None, append=False, format=None, index=True, min_itemsize=None, nan_rep=None, dropna=None, data_columns=None, errors='strict', encoding='UTF-8')	

Fin tableau 5.4

9	to_json([path_or_buf, orient, date_format, ...])	Convertissez l'objet en une chaîne JSON – qui est l'abréviation de «JavaScript Object Notation», un format indépendant de tout langage pour échanger des données sur le Web
	Series.to_json(path_or_buf=None, orient=None, date_format=None, double_precision=10, force_ascii=True, date_unit='ms', default_handler=None, lines=False, compression='infer', index=True, indent=None, storage_options=None)	
10	to_latex([buf, columns, col_space, header, ...])	Permet de rendre l'objet dans un tableau Latex (langage et un système de composition de documents), une table longue ou une table imbriquée
11	to_sql(name, con[, schema, if_exists, ...])	Permet d'écrire des enregistrements stockés dans une base de données SQL
	Series.to_sql(name, con, schema=None, if_exists='fail', index=True, index_label=None, chunksize=None, dtype=None, method=None)	
12	to_markdown([buf, mode, index, storage_options])	Permet d'imprimer des séries au format «Markdown» – (un langage de balisage léger)
13	to_numpy([dtype, copy, na_value])	Permet de transformer en massive de type NumPy
14	to_xarray()	Permet de renvoyer un objet tableau (massive) de l'objet pandas
15	to_period([freq, copy])	Permet de convertir les données de l'objet «Series» en index d'une fréquence spécifique
16	to_pickle(path[, compression, protocol, ...])	Permet une sérialisation d'objets dans un fichier
17	to_timestamp([freq, how, copy])	Permet de convertir un index de période en index de date et d'heure (DatetimeIndex)

FR V.3. Les autres méthodes sur les «Series»

Il existe en principe une multitude de fonctions qui peut être utilisée pour élargir la fonctionnalité du traitement des données sur la «Series». Dans le tableau 5.5, nous avons une centaine de méthodes.

Tableau 5.5 – Les autres fonctions sur les «Series»

1	abs()
2	add(other[, level, fill_value, axis])
3	add_prefix(prefix)
4	add_suffix(suffix)
5	agg([func, axis])
6	aggregate([func, axis])
7	align(other[, join, axis, level, copy, ...])
8	all([axis, bool_only, skipna, level])
9	any([axis, bool_only, skipna, level])
10	concat (objs, axis = 0, join='outer',...)
11	apply(func[, convert_dtype, args])
12	argmax([axis, skipna])
13	argmin([axis, skipna])
14	argsort([axis, kind, order])
15	asfreq(freq[, method, how, normalize, ...])
16	asof(where[, subset])
17	astype(dtype[, copy, errors])
18	at_time(time[, asof, axis])
19	autocorr([lag])
20	backfill([axis, inplace, limit, downcast])
21	between(left, right[, inclusive])
22	between_time(start_time, end_time[, ...])
23	bfill([axis, inplace, limit, downcast])
24	bool()
25	cat
26	clip([lower, upper, axis, inplace])
27	combine(other, func[, fill_value])
28	combine_first(other)
29	compare(other[, align_axis, keep_shape, ...])

Suite 1 du tableau 5.5

30	convert_dtypes([infer_objects, ...])
31	copy([deep])
32	corr(other[, method, min_periods])
33	count([level])
34	cov(other[, min_periods, ddof])
35	cummax([axis, skipna])
36	cummin([axis, skipna])
37	cumprod([axis, skipna])
38	cumsum([axis, skipna])
39	describe([percentiles, include, exclude, ...])
40	diff([periods])
41	div(other[, level, fill_value, axis])
42	divide(other[, level, fill_value, axis])
43	divmod(other[, level, fill_value, axis])
44	dot(other)
45	drop([labels, axis, index, columns, level, ...])
46	drop_duplicates([keep, inplace])
47	droplevel(level[, axis])
48	dropna([axis, inplace, how])
49	dt
50	duplicated([keep])
51	eq(other[, level, fill_value, axis])
52	equals(other)
53	ewm([com, span, halflife, alpha, ...])
54	expanding([min_periods, center, axis, method])
55	explode([ignore_index])
56	factorize([sort, na_sentinel])
57	ffill([axis, inplace, limit, downcast])
58	fillna([value, method, axis, inplace, ...])
59	filter([items, like, regex, axis])
60	first(offset)
61	first_valid_index()
62	floordiv(other[, level, fill_value, axis])
63	ge(other[, level, fill_value, axis])
64	get(key[, default])

Suite 2 du tableau 5.5

65	groupby([by, axis, level, as_index, sort, ...])
66	gt(other[, level, fill_value, axis])
67	head([n])
68	hist([by, ax, grid, xlabelsize, xrot, ...])
69	idxmax([axis, skipna])
70	idxmin([axis, skipna])
71	infer_objects()
72	info([verbose, buf, max_cols, memory_usage, ...])
73	interpolate([method, axis, limit, inplace, ...])
74	isin(values)
75	isna()
76	isnull()
77	item()
78	items()
79	iteritems()
80	keys()
81	kurt([axis, skipna, level, numeric_only])
82	kurtosis([axis, skipna, level, numeric_only])
83	last(offset)
84	last_valid_index()
85	le(other[, level, fill_value, axis])
86	lt(other[, level, fill_value, axis])
87	mad([axis, skipna, level])
88	map(arg[, na_action])
89	mask(cond[, other, inplace, axis, level, ...])
90	max([axis, skipna, level, numeric_only])
91	mean([axis, skipna, level, numeric_only])
92	median([axis, skipna, level, numeric_only])
93	memory_usage([index, deep])
94	min([axis, skipna, level, numeric_only])
95	mod(other[, level, fill_value, axis])
96	mode([dropna])
97	mul(other[, level, fill_value, axis])
98	multiply(other[, level, fill_value, axis])
99	ne(other[, level, fill_value, axis])

Suite 3 du tableau 5.5

100	nlargest([n, keep])
101	notna()
102	notnull()
103	nsmallest([n, keep])
104	nunique([dropna])
105	pad([axis, inplace, limit, downcast])
106	pct_change([periods, fill_method, limit, freq])
107	pipe(func, *args, **kwargs)
108	plot
109	pop(item)
110	pow(other[, level, fill_value, axis])
111	prod([axis, skipna, level, numeric_only, ...])
112	product([axis, skipna, level, numeric_only, ...])
113	quantile([q, interpolation])
114	radd(other[, level, fill_value, axis])
115	rank([axis, method, numeric_only, ...])
116	ravel([order])
117	rdiv(other[, level, fill_value, axis])
118	rdivmod(other[, level, fill_value, axis])
119	reindex(*args, **kwargs)
120	reindex_like(other[, method, copy, limit, ...])
121	rename([index, axis, copy, inplace, level, ...])
122	rename_axis([mapper, index, columns, axis, ...])
123	reorder_levels(order)
124	repeat(repeats[, axis])
125	replace([to_replace, value, inplace, limit, ...])
126	resample(rule[, axis, closed, label, ...])
127	reset_index([level, drop, name, inplace])
128	rfloordiv(other[, level, fill_value, axis])
129	rmod(other[, level, fill_value, axis])
130	rmul(other[, level, fill_value, axis])
131	rolling(window[, min_periods, center, ...])
132	round([decimals])
133	rpow(other[, level, fill_value, axis])
134	rsub(other[, level, fill_value, axis])

Fin du tableau 5.5

135	rtruediv(other[, level, fill_value, axis])		
136	sample([n, frac, replace, weights, ...])		
137	searchsorted(value[, side, sorter])		
138	sem([axis, skipna, level, ddof, numeric_only])		
139	set_axis(labels[, axis, inplace])		
140	set_flags(*[, copy, allows_duplicate_labels])		
141	shift([periods, freq, axis, fill_value])		
142	skew([axis, skipna, level, numeric_only])		
143	shift(...)		
144	sort_index([axis, level, ascending, ...])		
145	sort_values([axis, ascending, inplace, ...])		
146	sparse	147	str
148	squeeze([axis])		
149	std([axis, skipna, level, ddof, numeric_only])		
150	sub(other[, level, fill_value, axis])		
151	subtract(other[, level, fill_value, axis])		
152	sum([axis, skipna, level, numeric_only, ...])		
153	swapaxes(axis1, axis2[, copy])		
154	swaplevel([i, j, copy])	155	tail([n])
156	take(indices[, axis, is_copy])		
157	transform(func[, axis])		
158	transpose(*args, **kwargs)		
159	truediv(other[, level, fill_value, axis])		
160	truncate([before, after, axis, copy])		
161	tz_convert(tz[, axis, level, copy])		
162	tz_localize(tz[, axis, level, copy, ...])		
163	unique()		
164	unstack([level, fill_value])		
165	update(other)		
166	value_counts([normalize, sort, ascending, ...])		
167	var([axis, skipna, level, ddof, numeric_only])		
168	view([dtype])		
169	where(cond[, other, inplace, axis, level, ...])		
170	xs(key[, axis, level, drop_level])		

FR V.4. Pandas en massive bidimensionnel - DataFrame

FR V.4.1. Stockage et insertion des données

Comment stocker les données dans une table en utilisant DataFrame? DataFrame a une structure de données bidimensionnelle (voir figure 5.1) qui peut stocker des données de différents types: caractères, des nombres entiers, des valeurs à virgule flottante, des valeurs textuelles, etc. Retenons également que chaque colonne de la DataFrame est une «Series» (une succession verticale des valeurs (données)).

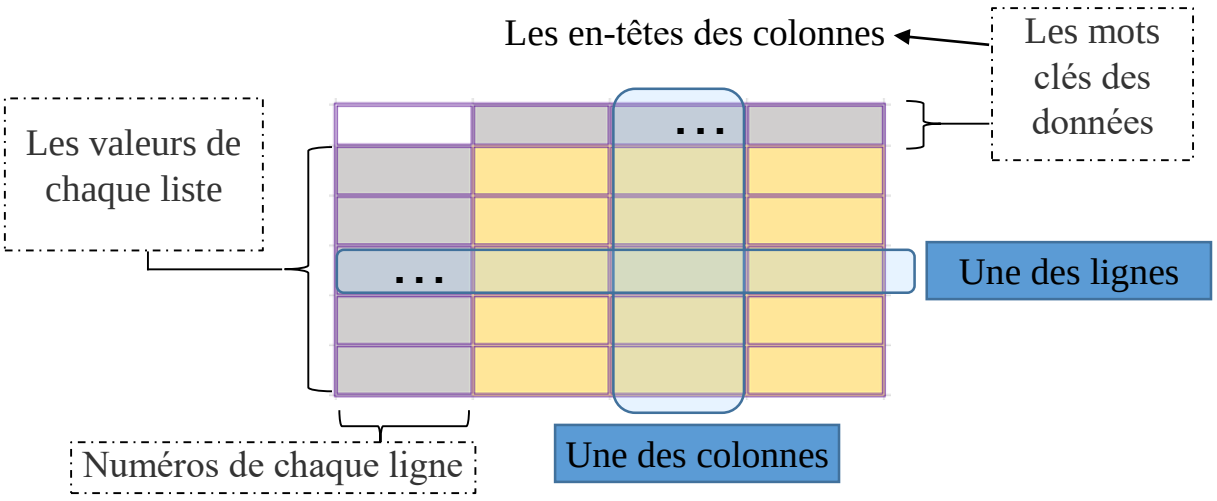


Fig. 5.1. La structure de table «DataFrame»

Exemple 1

Considérons que nous avons dans une salle de classe 4 étudiants absents. Il nous est demandé de représenter les données de ces étudiants dans un «DataFrame» sachant que nous connaissons leurs noms complets (Ivanov Ivan, Petrov Dima, Sidorov Sacha et Panova Irina); leurs dates de naissance (10.02. 2000, 23.11. 2001, 12.05.1999 et 27.04.2000) et leurs numéros de carte d'identité (233, 632, 595 et 810). Comme nous constatons,

ces données ont des types différents (valeurs textuelles – caractères, dates et nombres entiers). Le devoir consiste à stocker ces données des étudiants en utilisant Pandas. Comment le réaliser de manière à obtenir le résultat de la figure 5.2 ?

	Noms et prénoms	Dates de naissance	Numéros de carte d'identité
0	Ivanov Ivan	10.02.2000	233
1	Petrov Dima	23.11.2001	632
2	Sidorov Sacha	12.05.1999	595
3	Panova Irina	27.04.2000	810

Fig. 5.2. DataFrame de l'exemple №1 (résultat)

La syntaxe de «DataFrame» dans Pandas est la suivante:

DataFrame(data=None, index: Axes | None=None, columns: Axes | None=None, dtype: Dtype | None=None, copy: bool | None=None),

où ***data*** – peut contenir: soit des données en forme «dictionnaire», soit des données en séries, soit des données en tableau, soit des données constantes, soit des classes de données ou des objets de type liste;

index – est le label utilisé pour l'image résultante. Par défaut, il va de 0 au nombre de ligne à faire sortir;

columns – les noms des étiquettes de colonne à utiliser pour les différentes données;

dtype - type de données à forcer (par défaut aucun, s'il existe, un seul «dtype» est autorisé) : int, int8, int64, object, etc.

copy (de type booléen, par défaut aucun) – permet de copiez les données des entrées. Pour les données de type «dictionnaire», la valeur par défaut «None» se comporte comme *copy=True*. Pour l'entrée «DataFrame» ou 2d ndarray, la valeur par défaut *None* se comporte comme *copy=False*.

Réponse : Utilisons un dictionnaire pour construire la DataFrame:

```
1  import pandas as pa
2
3  data_p = pa.DataFrame(
4      {
5          "Noms et prénoms": [
6              "Ivanov Ivan",
7              "Petrov Dima",
8              "Sidorov Sacha",
9              "Panova Irina"
10         ],
11         "Dates de naissance": [
12             "10.02.2000",
13             "23.11.2001",
14             "12.05.1999",
15             "27.04.2000"
16             ],
17         "Numéros de carte d'identité": [
18             233,
19             632,
20             595,
21             810
22             ]
23     }
24 )
25
26 print(data_p)
```

Résultat après exécution du code

	Noms et prénoms	Dates de naissance	Numéros de carte d'identité
0	Ivanov Ivan	10.02.2000	233
1	Petrov Dima	23.11.2001	632
2	Sidorov Sacha	12.05.1999	595
3	Panova Irina	27.04.2000	810

Commentaire et introduction de quelques changements

Nous avons observé l'apparition automatique des numéros de chaque ligne de 0 à 3, étant donné que nous avons effectivement 4 lignes de données, après l'exécution de précédant code. Voyons également comment modifier ces numéros en d'autres par exemple en des labels "a", "b", "c" et "d" avec le paramètre «index».

Comment changer ces numéros?

```
1  import pandas as pa
2
3  d = {
4        "Noms et prénoms":     ["Ivanov Ivan", "Petrov Dima",
5                               "Sidorov Sacha", "Panova Irina"],
6        "Dates de naissance":   ["10.02.2000", "23.11.2001",
7                               "12.05.1999", "27.04.2000"],
8     "Numéros de carte d'identité": [233, 632,595, 810]
9      }
10
11 data_p2 = pa.DataFrame(data=d, index= ["a","b","c","d"])
12 print(data_p2)
```

Résultat après exécution du code

	Noms et prénoms	Dates de naissance	Numéros de carte d'identité
a	Ivanov Ivan	10.02.2000	233
b	Petrov Dima	23.11.2001	632
c	Sidorov Sacha	12.05.1999	595
d	Panova Irina	27.04.2000	810

Les remarques que nous pouvons faire:

Le tableau obtenu comporte 4 colonnes, avec une étiquette sur chaque colonne: « Noms et prénoms », « Dates de naissance » et « Numéros de carte d'identité ».

La colonne « Noms et prénoms » a les valeurs de type chaîne; la colonne « Dates de naissance » a les valeurs de type chaîne qui peuvent être transformées en type « Datetime »; la colonne « Numéros de carte d'identité » comporte les valeurs de type « int » – les entiers.

Dans les tableaux de type « DataFrame », une même colonne peut avoir des valeurs à type différents.

Nous constatons également que chaque colonne est une « Series ».

FR V.4.2 Les manœuvres possibles sur une colonne bien déterminée

Pour s'adresser à une colonne donnée, nous pouvons utiliser la commande suivante :

nom du tableau[“*nom de l'étiquette*”]

Changement à faire dans le code précédent:

```
11 data_p2 = pa.DataFrame(data=d)
12 col_1 = data_p2["Numéros de carte d'identité"]
13 print(col_1)
```

Résultat après exécution du code	
0	233
1	632
2	595
3	810

Pour s'adresser à un élément d'une colonne donnée, nous pouvons utiliser la commande suivante :

Exemple:

```
11 data_p2 = pa.DataFrame(data=d)
12 col_1 = data_p2["Numéros de carte d’identité"]
13 elt = col_1[n]
```

col_1 – est la variable qui recevra tous les éléments de la colonne "Numéros de carte d’identité" ;

data_p2 – tableau DataFrame

n – numéro de l'élément dans la Series (la colonne)

Pour imprimer la colonne dans son intégralité

```
12 col_1 = data_p2["Numéros de carte d’identité"]
13 print(col_1[:])
   # ou alors
13 print(col_1)
```

Pour imprimer les deux premiers éléments de la colonne

```
13 col_1 = data_p2["Numéros de carte d’identité"]
14 print(col1[:2])
15
```

Pour imprimer tous les éléments de la colonne qui se tiennent après le premier

```
13 col_1 = data_p2["Numéros de carte d'identité"]
14 print(col_1[1:])
15
```

Pour imprimer tous les éléments de la colonne qui se tiennent après le deuxieme

```
13 col_1 = data_p2["Numéros de carte d'identité"]
14 print(col_1[2:])
15
```

Pour trouver et imprimer la valeur minimale d'une colonne

```
13 vmin = data_p2["Numéros de carte d'identité"].min()
14 print(vmin)
15
```

Pour trouver et imprimer la valeur maximale d'une colonne

```
13 vmax = data_p2["Numéros de carte d'identité"].max()
14 print(vmax)
15
```

Pour trier une colonne donnée par ordre croissant dans le tableau «DataFrame».

Pour trier ce «DataFrame» par une seule colonne, nous devons spécifier le nom de la colonne à l'aide du paramètre «by».

Avec le paramètre «ascending», nous orientons l'ordre de tri de manière croissante(avec une valeur «True») ou décroissante(avec une valeur «False»). Par défaut, la fonction «sort_value» trie par ordre croissant.

```
13 tri1 = data_p2.sort_values(by="Numéros de carte d'identité",
                              ascending=True)
14 print(tri1)
```

Résultat après exécution du code

	Noms et prénoms	Dates de naissance	Numéros de carte d'identité
0	Ivanov Ivan	10.02.2000	233
2	Sidorov Sacha	12.05.1999	595
1	Petrov Dima	23.11.2001	632
3	Panova Irina	27.04.2000	810

Avec le paramètre «inplace» - le tri se fera dans le tableau DataFrame sur place. Une nouvelle variable ne saurait obtenir le résultat de ce tri. Le résultat restera dans l'ancien tableau, pour cela, il faut donner au paramètre «inplace» la valeur «True».

```
13 data_p2.sort_values(by="Noms et prénoms", ascending=True,
                       inplace=True)
14 print(data_p2)
```

Résultat après exécution du code

	Noms et prénoms	Dates de naissance	Numéros de carte d'identité
0	Ivanov Ivan	10.02.2000	233
3	Panova Irina	27.04.2000	810
1	Petrov Dima	23.11.2001	632
2	Sidorov Sacha	12.05.1999	595

Pour trier une colonne donnée par ordre croissant dans le tableau «DataFrame».

Pour faire le tri sur plusieurs colonnes du tableau «DataFrame», en utilisant toujours la méthode «sort_values», nous devons indiquer les noms des colonnes que nous allons utiliser pou trier, séparés par des virgules. Après, avec le paramètre «ascending», il faudra aussi spécifier l'ordre de tri des colonnes séparées par des virgules. Alors, essayonsde faire ce tri sur les colonnes «Noms et prénoms» et «Numéros de carte d'identité».

```
13  tri2 = data_p2.sort_values(["Noms et prénoms","Numéros de carte
                    d'identité"], ascending=[True,True])
14  print(tri2)
```

Résultat après exécution du code

	Noms et prénoms	Dates de naissance	Numéros de carte d'identité
0	Ivanov Ivan	10.02.2000	233
3	Panova Irina	27.04.2000	810
1	Petrov Dima	23.11.2001	632
2	Sidorov Sacha	12.05.1999	595

Ce résultat donne l'impression que la dernière colonne n'a pas été triée. Pour s'assurer que tout fonctionne bien, essayons d'apporter quelques modifications aux données initiales «DataFrame». Changeons «Sidorov Sacha» par un nouveau «Petrov Dima» ayant pour date de naissance 12.05.1999 et pour numéro de carte d'identité égale à 595. Après cette modification, l'exécutions du code précédant donnera le résultat suivant:

Résultat après exécution du code

	Noms et prénoms	Dates de naissance	Numéros de carte d'identité
0	Ivanov Ivan	10.02.2000	233
3	Panova Irina	27.04.2000	810
2	Petrov Dima	12.05.1999	595
1	Petrov Dima	23.11.2001	632

FR V.4.3 Quelques opérations et interactions entre Excel et Pandas

Étant donné que Excel, OpenOffice calc, et bien d'autres feuilles de calcule sont utilisées par bon nombre d'utilisateurs, il est important de savoir comment intégrer ou manipuler les données à partir de Pandas.

Utilisons les données du lien suivant pour apprendre à faire des opérations différentes. Ce fichier contient certaines données (voir fig. 5.3) au format csv.

url = https://raw.githubusercontent.com/pandas-dev/pandas/main/pandas/tests/io/data/csv/tips.csv

```
total_bill,tip,sex,smoker,day,time,size
16.99,1.01,Female,No,Sun,Dinner,2
10.34,1.66,Male,No,Sun,Dinner,3
21.01,3.5,Male,No,Sun,Dinner,3
23.68,3.31,Male,No,Sun,Dinner,2
24.59,3.61,Female,No,Sun,Dinner,4
25.29,4.71,Male,No,Sun,Dinner,4
8.77,2.0,Male,No,Sun,Dinner,2
26.88,3.12,Male,No,Sun,Dinner,4
15.04,1.96,Male,No,Sun,Dinner,2
14.78,3.23,Male,No,Sun,Dinner,2
10.27,1.71,Male,No,Sun,Dinner,2
35.26,5.0,Female,No,Sun,Dinner,4
15.42,1.57,Male,No,Sun,Dinner,2
18.43,3.0,Male,No,Sun,Dinner,4
```

Fig. 5.3. Structure de données dans l'URL du fichier csv

Comment lire les données du fichier qui se trouve dans un url

```
1  import pandas as pa
2
3  url = "https://raw.githubusercontent.com/pandas-
4                  dev/pandas/main/pandas/tests/io/data/csv/tips.csv"
5  data = pa.read_csv(url)
6  infos = data.head(6)
7  print(infos)
```

Résultat après exécution du code

	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
1	10.34	1.66	Male	No	Sun	Dinner	3
2	21.01	3.50	Male	No	Sun	Dinner	3
3	23.68	3.31	Male	No	Sun	Dinner	2
4	24.59	3.61	Female	No	Sun	Dinner	4
5	25.29	4.71	Male	No	Sun	Dinner	4

Si à la place de la ligne 6, nous mettons:

```
6  infos = data.tail(5)
```

Nous aurons les 5 dernières lignes du fichier.

Résultat après exécution du code

	total_bill	tip	sex	smoker	day	time	size
239	29.03	5.92	Male	No	Sat	Dinner	3
240	27.18	2.00	Female	Yes	Sat	Dinner	2
241	22.67	2.00	Male	Yes	Sat	Dinner	2
242	17.82	1.75	Male	No	Sat	Dinner	2
243	18.78	3.00	Female	No	Thur	Dinner	2

La particularité de Pandas est qu'il permet directement d'effectuer des opérations sur des colonnes entières, par rapport aux autres feuilles de calculs. Étudions cet exemple pour comprendre:

Supposons que nous voulons changer les données de la colonne [sex] du fichier « *tips.csv* » par les symboles « F » et « M ». « F » doit remplacer le mot « Female » et « M », le mot « Male ». Dans la mesure où nous n'avons que deux mots, nous pouvons utiliser la méthode « *where* » associée au module NumPy. Alors que dans Excel, nous allons utiliser la condition « si – alors» (voir fig. 5.4).

```
1  import pandas as pa
2  import numpy as nu
3
4  url = "https://raw.githubusercontent.com/pandas-
5                        dev/pandas/main/pandas/tests/io/data/csv/tips.csv"
6  data = pa.read_csv(url)
7  data["sex"] = nu.where(data["sex"] = = "Female", "F", "M")
8  infos = data.head(5)
9  print(infos)
```

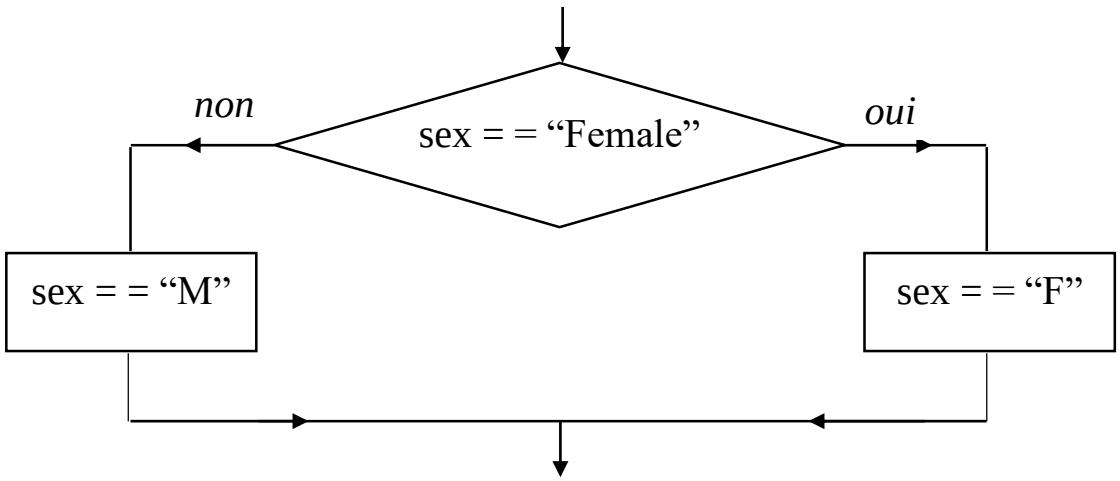


Fig. 5.4. Illustration de la fonction « si – alors »

Résultat après exécution du code

	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	F	No	Sun	Dinner	2
1	10.34	1.66	M	No	Sun	Dinner	3
2	21.01	3.50	M	No	Sun	Dinner	3
3	23.68	3.31	M	No	Sun	Dinner	2
4	24.59	3.61	F	No	Sun	Dinner	4
5	25.29	4.71	M	No	Sun	Dinner	4

Il est possible de supprimer une colonne avec la fonction « drop », de changer le nom de la colonne avec la fonction « rename », de trouver la valeur maximale « max », minimale « min », ... d'une colonne. Ainsi, il existe un grand nombre de fonctions possible d'appliquer sur les colonnes d'un fichier. Avec la fonction « columns », nous pouvons énumérer les noms de différentes colonnes dans «*DataFrame*».

Les codes pourrons ressembler à ces lignes:

```
7  new_value = data["name_colonne"].fonction()
```

```
7  data = data.drop("column_name ", axis=1)
```

```
7  infos = data.rename(columns={"old_name": "new_name"}).tail(6)
```

```
7  infos = data["column_name "].str.find("what_to_find").tail(6)
```

Pour manipuler les chaînes, nous pouvons utiliser les méthodes existantes que nous avons décrit dans le manuel précédent – Tom I ou alors dans d'autres livres.

Est-il possible en Python avec la bibliothèque Pandas d'enregistrer des données dans Excel ? Bien sûr, oui. Pour cela, il suffit juste d'utiliser la fonction ***to_excel()*** avec ses paramètres. La syntaxe de cette fonction est la suivante:

```
DataFrame.to_excel(excel_writer, sheet_name='Sheet1', na_rep='',
float_format=None, columns=None, header=True, index=True,
index_label=None, startrow=0, startcol=0, engine=None,
merge_cells=True, encoding=None, inf_rep='inf', verbose=True,
freeze_panes=None, storage_options=None),
```

où

excel_writer : chemin de fichier ou **ExcelWriter** existant;

sheet_name: nom de la feuille qui contiendra les données (par défaut ‘Sheet1’);
na_rep: représentation des données manquantes;
float_format: format de chaîne pour les nombres à virgule flottante;
columns: colonne(s) qui sera(ont) enregistrée(s) avec les données;
header: indiquer les noms des colonnes. Si une liste de chaînes est donnée, elle est supposée être des alias pour les noms de colonnes.
index: inclure (valeur True) ou exclure (False) les noms (index) des lignes;
index_label: étiquette de colonne pour la rangée des index;
startrow: coordonnée *x* de la première ligne de la cellule d’en haut vers la gauche pour écrire les données;
startcol: coordonnée *y* de la première colonne de la cellule d’en haut vers le bas pour écrire les données;
engine: indiquer le moteur à utiliser : 'openpyxl' ou ’xlsxwriter';
merge_cells : écrit des lignes MultiIndex et hiérarchiques en tant que cellules fusionnées;
encoding: encodage du fichier Excel résultant; {« openpyxl », }
inf_rep: représentation pour l'infini (il n'y a pas de représentation native pour l'infini dans Excel);
verbose: affiche plus d'informations dans les journaux d'erreurs;
freeze_panes: spécifie la ligne la plus en bas et la colonne la plus à droite qui doivent être gelées;
storage_options: options supplémentaires qui ont du sens pour une connexion de stockage particulière, par exemple «host», «port», «username», «password», etc.

☛ Syntaxe de «**ExcelWriter**»

ExcelWriter(*path*: FilePathOrBuffer | ExcelWriter, *engine*=None, *date_format*=None, *datetime_format*=None, *mode*: str="w", *storage_options*: StorageOptions=None, *if_sheet_exists*: str | None=None, *engine_kwargs*: dict | None=None)

où

path: le chemin d'accès au fichier du type *xls* ou *xlsx* ou *ods* ;
engine: moteur à utiliser pour l'écriture ;
date_format: permet de formater la chaîne pour les dates écrites dans des fichiers Excel ;

(Par exemple 'AAAA-MM-JJ') ;

datetime_format: permet de formater la chaîne de format pour les objets datetime écrits dans des fichiers Excel ;

(Par exemple, 'AAAA-MM-JJ HH:MM:SS') ;

mode: mode fichier à utiliser (écrire ou ajouter -{'w', 'a' ou 'r+'}, par défaut 'w') ;
storage_options: options supplémentaires qui ont du sens pour une connexion de stockage particulière, par exemple «host», «port», «username», «password» , etc.
if_sheet_exists: permet d'orienter l'écriture sur une feuille d'excel qui existe déjà (ceci en mode d'ajout uniquement) - {'*error*', '*new*', '*replace*', '*overlay*'}, par défaut, la valeur est 'error' ;

error: déclenchez une erreur ValueError;
new: créer une nouvelle feuille, avec un nom déterminé par le moteur;
replace: supprimez le contenu de la feuille avant d'y écrire un nouveau ;
overlay: écrivez du contenu dans la feuille existante sans supprimer l'ancien contenu

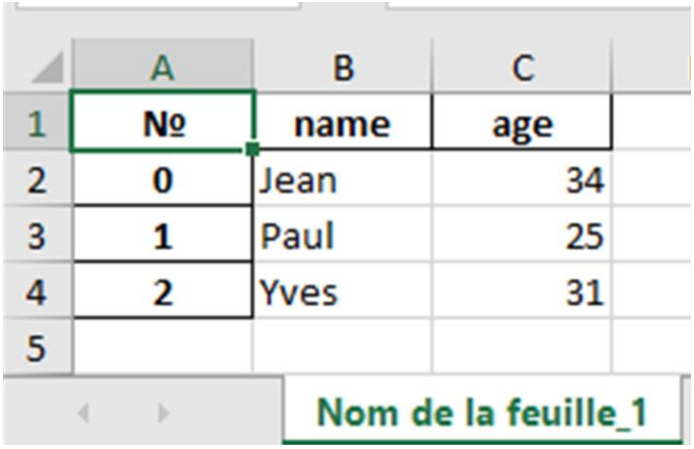
engine_kwargs: arguments de mots clés à transmettre au moteur *engine*.

xlsxwriter (mode écrire)
openpyxl (mode écrire et ajouter)
odswriter

☛ Exemple 1 de code pour écrire des données dans Excel à l'aide de la fonction ***to_excel()***. **Cas où le fichier (*file_excel1.xlsx*) n'existe pas**.

```
1  import os
2  import pandas as pa
3  from pandas.io.excel import ExcelWriter
4
5  if __name__ == "__main__":
6      path_file = os.path.dirname(__file__) +
7                          r"\FolderData\file_excel1.xlsx"
8      data_p1 = pa.DataFrame(
9          [['Jean', 34],
10          ['Paul', 25],
11          ['Yves', 31]],
12         columns=['name', 'age'])
13     with ExcelWriter(path_file,  mode="w") as exemplaire_write:
14         data_p1.to_excel(exemplaire_write, sheet_name="Nom de la
15                          feuille_1",  index_label="№", startrow=0,
16                          startcol=0)
```

Résultat après exécution du code



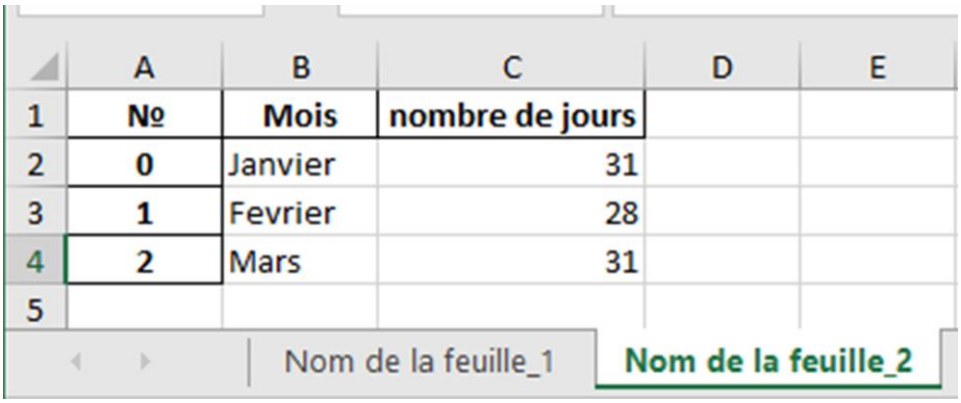
	A	B	C
1	№	name	age
2	0	Jean	34
3	1	Paul	25
4	2	Yves	31
5			

Nom de la feuille_1

Exemple 2 de code pour écrire des données dans Excel à l'aide de la fonction to_excel(). **Cas où le fichier (*file_excel1.xlsx*) existe** et nous voulons ajouter une seconde feuille avec d'autres données dans le même fichier. Pour cela, vous devez créer un objet ExcelWriter avec le nom du fichier en question, après spécifier la feuille dans le fichier à sauvegarder.

```
1  import os
2  import pandas as pa
3  from pandas.io.excel import ExcelWriter
4
5  if __name__ == "__main__":
6      path_file = os.path.dirname(__file__) +
7                      r"\FolderData\file_excel1.xlsx"
8      data_p2 = pa.DataFrame(
9          [['Janvier', 31],
10          ['Fevrier', 28],
11          ['Mars', 31]],
12         columns=['Mois', 'nombre de jours'])
13     with ExcelWriter(path_file,  mode="a", engine="openpyxl") as
14                     exemplaire_write:
15         data_p2.to_excel(exemplaire_write, sheet_name="Nom de la
16                     feuille_2",  index_label="№", startrow=0,
17                     startcol=0)
```

Résultat après exécution du code



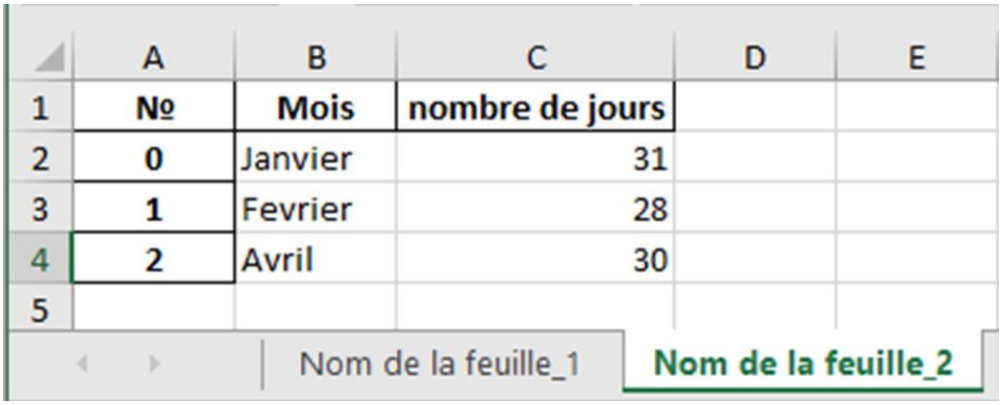
	A	B	C	D	E
1	№	Mois	nombre de jours		
2	0	Janvier	31		
3	1	Fevrier	28		
4	2	Mars	31		
5					

Nom de la feuille_1 | Nom de la feuille_2

de la fonction to_excel(). **Comment mettre à jour le fichier** en changeant juste une information? Essayons de changer «Mars» par «Avril» avec un nombre de jours égale à 30.

```
1  import os
2  import pandas as pa
3  from pandas.io.excel import ExcelWriter
4
5  if __name__ == "__main__":
6      path_file = os.path.dirname(__file__) +
7                  r"\FolderData\file_excel1.xlsx"
8      data_p2 = pa.DataFrame([['Avril', 30]], columns=None)
9
10     with ExcelWriter(path_file,  mode="a", engine="openpyxl",
11     if_sheet_exists="overlay") as exemplaire_write:
12
13         data_p2.to_excel(exemplaire_write, sheet_name="Nom de la
14                 feuille_2", index=False, header=False, startrow=3,
                   startcol=1)
```

Résultat après exécution du code



	A	B	C	D	E
1	№	Mois	nombre de jours		
2	0	Janvier	31		
3	1	Fevrier	28		
4	2	Avril	30		
5					

Nom de la feuille_1 | Nom de la feuille_2

Le principal paramètre pour la mise à jour est «***if_sheet_exists***», avec la valeur «***overlay***», Avec ce paramètre, il est possible de garder les anciennes données, d'effacer ces anciennes données dans la même feuille ou alors de créer automatiquement une nouvelle feuille pour y mettre les nouvelles données.

Emballer un fichier. Il est également possible d'emballer un fichier Excel dans une archive de type *zip*. Pour cela, nous allons procéder de la manière suivante:

- indiquer le chemin où sera stocké le fichier compressé ;
- introduire les données (par exemple dans une DataFrame) qui seront gardées dans le fichier Excel ;
- mettre les données dans un fichier Excel qui sera emballé ;
- à travers la méthode « ZipFile », il faut compresser le fichier.

```
1  import os, zipfile
2  import pandas as pa
3
4  if __name__ == "__main__":
5      path_file1 = os.path.dirname(__file__) +
6                           r"\FolderData\file_zip1.zip"
7      data_p3 = pa.DataFrame(
8          [['Janvier', 31],
9           ['Fevrier', 28],
10          ['Mars', 31]],
11         columns=['Mois', 'nombre de jours'])
12
13 with zipfile.ZipFile(path_file1, "w") as zpf:
14     with zpf.open("file_name_excel.xlsx", "w") as data_temp:
15         with pa.ExcelWriter(data_temp) as exemplaire_writer:
16             data_p3.to_excel(exemplaire_writer)
```

Résultat après exécution du code
Nous avons un fichier (*file_name_excel.xlsx*) avec données compressé en «*zip*» sous le nom: *file_zip1.zip*

Créer un document Excel avec les données suivantes (voir la figure 5.5):

	A	B	C	D
1	№	Nom	Prénom	Note
2	1	Petrov	David	3,24
3	2	Ivanov	Ivan	15,02
4	3	Sidorov	Viktor	10,1
5	4	Liskok	Andrei	9,12
6	5	Efimova	Nastia	11,5
7	6	Kuznesov	Kirill	5,76
8	7	Kalinina	Tatiana	18,2
9	8	Zaitseva	Lena	14,11
10	9	Choutov	Maxim	4,97
11	10	Gouceva	Yulia	7,45
12	11	Zemtsova	Alina	8,26
13				

feuille1 feuille2

	A	B	C	D
1	№	Date	Nombre	Réligion
2	1	20.02.2014	45	Catholique
3	2	11.08.2003	123	Orthodoxe
4	3	07.10.2011	23	Protestant
5	4	25.05.2008	56	Musulman
6	5	12.06.2010	201	Catholique
7	6	17.01.2020	67	Orthodoxe
8	7	02.12.2013	110	Orthodoxe
9	8	23.03.2015	45	Musulman
10	9	17.07.2003	120	Protestant
11	10	29.10.2018	79	Orthodoxe
12	11	05.09.2009	88	Paienne
13				

feuille1 feuille2

Fig. 5.5. Fichier Excel «*data_excel1.xlsx* à *2 feuilles*»

☛ La méthode «***read_excel***» peut lire les fichiers locaux du système qui ont les extensions suivantes : *xls*, *xlsx*, *xlsm*, *xls*, *odt*, *ods* et *odt*

ou peut encore lire ces fichiers de l'URL. La syntaxe de la fonction ***pandas.read_excel***.

*pandas.**read_excel**(io, sheet_name=0, header=0, names=None, index_col=None, usecols=None, squeeze=None, dtype=None, engine=None, converters=None, true_values=None, false_values=None, skiprows=None, nrows=None, na_values=None, keep_default_na=True, na_filter=True, verbose=False, parse_dates=False, date_parser=None, thousands=None, decimal='.', comment=None, skipfooter=0, convert_float=None, mangle_dupe_cols=True, storage_options=None)*

io – nom (chemin) du fichier à lire ;

sheet_name: il est utilisé pour indiquer le ou les noms de feuille à lire. La valeur par défaut est 0 (première feuille); avec la valeur 1: deuxième feuille à utiliser ; "Sheet1": Feuille à utiliser avec le nom "Sheet1"; Si la valeur est «None», alors toutes les feuilles de calcul seront lues.

header –il est utilisé pour les étiquettes de colonnes (indexCe à 0) ;

names – il contient la liste des étiquettes de colonnes à utiliser ;

index_col – colonne à utiliser comme étiquettes de ligne ;

dtype – type de données pour les colonnes ;

usecols – liste des lettres ou rangées de colonne Excel (par exemple "A: F" ou "A, D, F: J" ;autres exemples: usecols='B:D' ou usecols=[0,2] ou usecols=['Nom', 'Note'] ;

engine – les différents moteurs pour la manipulation: “xlrd”, “openpyxl”, “odf”, “pyxlsb” ;

“xlrd” prend en charge les anciens fichiers d’ Excel à extension (.xls).

“openpyxl” prend en charge les nouveaux formats de fichiers d’Excel.

“odf” prend en charge les formats de fichiers OpenDocument (.odf, .ods, .odt).

“pyxlsb” prend en charge les fichiers d’Excel binaires.

converters – permet de convertir des valeurs dans certaines colonnes ;

skiprows – permet d'éliminer un ou des lignes; par exemple: skiprows=[1] élimine la deuxième ligne; skiprows=[2,3] – élimine la troisième et quatrième ligne ; skiprows=2 – élimine les 2 premières lignes.

☛ Comment lire les données de la feuille 2 du fichier d'Excel «*data_excel1.xlsx*» de la figure 5.5. Afficher les 3 premières lignes.

```
1  import os
2  import pandas as pa
3
4  path_file = os.path.dirname(__file__) +
5                                  r"\FolderData\data_excel1.xlsx"
6  data_p = pa.read_excel(path_file, sheet_name='feuille2')
7
8  print(data_p.head(3))
```

Résultat après exécution du code

	№	Date	Nombre	Réligion
0	1	2014-02-20	45	Catholique
1	2	2003-08-11	123	Orthodoxe
2	3	2011-10-07	23	Protestant

De la même manière, il est possible de lire les données d'une autre feuille et faire des opérations.

Nous pouvons utiliser la fonction spéciale lambda dans ces opérations dans le but de faciliter certaines manipulations. Par exemple, nous voulons éliminer un nombre important de ligne, pour cela il sera moins commode d'énumérer toutes ces lignes dans des crochets que d'utiliser la formule simple suivant: la sixième ligne du programme va ressembler à celle ci:

```
6  data_p = pa.read_excel(path_file, sheet_name='feuille1',
7                         skiprows=lambda y: y<3)
```

À la place du chiffre 3, peut avoir un autre chiffre plus grand. Cette fonction efface toutes les 3 premières lignes.

Pour avoir plus d'informations sur les possibilités de «DataFrame», vous pouvez consulter le lien du site Pandas suivant: [*https://pandas.pydata.org/pandas-docs/stable/reference/frame.html*]. Il s'agit: de la conversion des données, des calculs statistiques, manipulation des étiquettes, traitement des données et bien d'autres.

FR V.5. Exercices sur Pandas

☛ ***Exercice 1 :*** *Ecrire un programme codes qui permet d'obtenir la valeur maximale des données parmi les valeurs négatives de la massive donnée du tableau 5.3 (utiliser la structure Series de Pandas).*

☛ ***Exercice 2 :*** *Si nous ajoutons le paramètre «inplace = True» dans la fonction «sort_values()», quel résultat ou données sur l'exemple №6 (sur la structure Series de Pandas) aura t-on dans les variables «new_data» et «data_se» ?*

☛ ***Exercice 3 :*** *Créer un nouveau module de tri – algorithme par sélection et utiliser le pour ordonner les données dans une liste (utiliser la structure Series de Pandas).*

☛ ***Exercice 4 :*** *Créer un nouveau module de tri – algorithme à bulles et utiliser le pour ordonner les données dans une liste (utiliser la structure Series de Pandas).*

☛ ***Exercice 5 :*** *Créer un nouveau module de tri – algorithme par insertion et utiliser le pour ordonner les données dans une liste (utiliser la structure Series de Pandas).*

☛ ***Exercice 6 :*** *Créer un nouveau module de tri – algorithme cocktail ou à bulles bidirectionnel et utiliser le pour ordonner les données dans une liste (utiliser la structure Series de Pandas).*

☛ ***Exercice 7 avec réponse:*** *Créer un programme codes qui permet de faire un tri de données de type «Series» par ordre croissant et décroissant ; enregistrer les résultats dans un fichier à format csv qui sera*

possible d'être ouvert par le logiciel Excel (Chaque groupe de données sera rangée dans une colonne bien déterminée).

Exercice 8 avec réponse: *Ecrire un programme codes qui vous permettent de lire et d'écrire des données dans un fichier de type csv (utiliser la structure Series de Pandas).*

Exercice 9 avec réponse*: Créer un programme codes qui permet d'ajouter une nouvelle colonne dans le fichier URL indiqué ci-dessous. Les informations doivent respecter la condition suivante: si la valeur de la cellule de la colonne « size » est supérieure à 2, alors celle de la nouvelle cellule de la colonne prend la valeur « Bien », dans le cas contraire, elle prend la valeur « Mal ». Afficher le résultat des 6 dernières lignes sur l'écran.*

url = "https://raw.githubusercontent.com/pandas-dev/pandas/main/pandas/tests/io/data/csv/tips.csv"

Exercice 10

1) Créer la DataFrame qui reflète les données suivantes (voir tableau 5.6) de 2 manières différentes (à partir d'un dictionnaire et à partir d'un vecteur NumPy) et choisir une pour la continuité:

Tableau 5.6 – Les informations sur les produits

№	Nom des produits	Unité de mesure	Prix d'achat	Prix de vente	Reste
1	riz	kg	110	125	250
2	huile	litre	500	575	120
3	sucre	kg	70	125	300
4	savon	pièces	180	195	120
5	eau	litre	60	85	100
6	orange	kg	115	150	150
7	mangue	kg	165	180	160
8	marmite	pièces	3500	3805	200
9	verre	pièces	250	300	350
10	citron	kg	90	105	50

2) Dans le tableau précédent de données, il faudra ajouter deux colonnes nommées respectivement «Date d'achat» et «Réduction» à des positions suivantes: juste après la deuxième colonne et juste avant la dernière colonne.

3) Remplir les deux colonnes des données aléatoires plus ou moins réelles.

4) Trouver les valeurs moyennes(à la onzième ligne) des colonnes suivantes : «Prix d'achat», «Prix de vente», «Réduction».

5) Déterminer à partir de Pandas, le bénéfice de l'entreprise après la vente de tous les produits.

6) Faire le tri des données en fonction des unités de mesure et la date d’achat.

7) Stocker tous les résultats dans un nouveau fichier au format Excel.

8) Utilisez les outils Pandas pour emballer l'ancien et le nouveau fichier Excel dans une archive zip.

FR V.6. Réponses aux exercices sur Pandas

Réponse de l'exercice 7

```
1  import os, csv
2  import pandas as pa
3
4  x = [67, 70, 156.5, 105, 170, 120, 105, 144, 135.5, 99]
5  ind = ["Litchi", "Banane", "Avocat", "Kumquat", "Ananas",
6                      "Orange", "Pomelo", "Citron", "Mandarin",
7                      "Pamplemousse"]
8  data_se = pa.Series(data=x, index=ind)
9
10 os.makedirs('FolderSort', exist_ok=True)
11 new_data = data_se.sort_values(ascending=False)
12 new_data.to_csv('FolderSort/out1.csv', sep=';')
13 new_data = data_se.sort_values(ascending=True, inplace=True)
14 data_se.to_csv('FolderSort/out2_1.csv', sep=';')
15
16 with open ("FolderSort/out1.csv", 'r', newline='') as csv1file:
17     with open ("FolderSort/out2_1.csv", 'a+', newline='') as csv2file:
18         read_csv = csv.reader(csv1file, delimiter=',')
19         for row in read_csv:
20             writer = csv.writer(csv2file)
21             writer.writerow(row)
```

Commentaire

Le symbole ';' permet à l'Excel d'ouvrir chaque rangée de données dans chaque colonne.

« *newline*=" » – permet d'interdire le vide entre les lignes des données

Deux fichiers sont crées, le premier (*out1.csv*) a le résultat du tri décroissant des données et le second (*out2_1.csv*) pour un début, possède le résultat du tri croissant des données et après, est ajouté le résultat du tri décroissant des données.

Réponse de l'exercice 8

Écrire des données dans un fichier de type csv

```
1  import csv, os
2
3  os.makedirs('FolderData', exist_ok=True)
4
5  with open ("FolderData/texte_exo8.csv", 'w+', newline='',
6                      encoding="utf-8") as csv_file8:
7     write_to_csv = csv.writer(csv_file8, delimiter=";")
8     write_to_csv.writerow(["Numéro", "Nom", "Code Pays",
9                      "Téléphone", "Pays"])
10    write_to_csv.writerow(["1", "Ivanov", "7", "9103456744",
11                     "Russie"])
12    write_to_csv.writerow(["2", "Magne", "237", "6551579",
13                     "Cameroun"])
14    write_to_csv.writerow(["3", "Titto", "39", "3354477333", "Italie"])
15    write_to_csv.writerow(["4", "Andreenko", "375", "291113322",
16                     "Bielorussie"])
17    write_to_csv.writerow(["5", "Xaili", "86", "1065440011",
18                     "Chine"])
```

Si nous ouvrons le fichier qui a été crée (*texte_exo8.csv*) par un rédacteur de texte simple, la structure des données va ressembler au dessin de la figure 5.3, et en ouvrant par le logiciel Excel ou OpenOffice calc, nous aurons les figures 5.4 et 5.5 respectivement.

```
Numéro;Nom;Code Pays;Téléphone;Pays
1;Ivanov;7;9103456744;Russie
2;Magne;237;6551579;Cameroun
3;Titto;39;3354477333;Italie
4;Andreenko;375;291113322;Bielorussie
5;Xaili;86;1065440011;Chine
```

	A	B	C	D	E	F
1	Numéro	Nom	Code	Pays	Téléphone	Pays
2	1	Ivanov	7	9103456744	Russie	
3	2	Magne	237	6551579	Cameroun	
4	3	Titto	39	3354477333	Italie	
5	4	Andreenko	375	291113322	Bielorussie	
6	5	Xaili	86	1065440011	Chine	

Fig. 5.4. Structure du fichier au format csv dans OpenOffice calc

	A	B	C	D	E	F
1	Numéro	Nom	Code	Pays	Téléphone	Pays
2	1	Ivanov	7	9103456744	Russie	
3	2	Magne	237	6551579	Cameroun	
4	3	Titto	39	3354477333	Italie	
5	4	Andreenko	375	291113322	Bielorussie	
6	5	Xaili	86	1065440011	Chine	

Fig. 5.5. Structure du fichier au format csv dans Excel

Lire et imprimer des données du fichier de type csv

```
1  import csv
2
3  with open ("FolderData/texte_exo8.csv", newline='',
4                                          encoding="utf_8") as csv_file_r:
5      read_csv = csv.DictReader(csv_file_r, delimiter=';')
6      print("Numéro", "Nom", "Code Pays","Téléphone", "Pays")
7      for lign in read_csv:
8          print(lign["Numéro"]," ",lign["Nom"]," ",lign["Code Pays"]," ",
9                      lign["Téléphone"]," ", lign["Pays"])
```

Résultat après exécution du code

```
Numéro  Nom  Code Pays  Téléphone  Pays
1  Ivanov  7  9103456744  Russie
2  Magne  237  6551579  Cameroun
3  Titto  39  3354477333  Italie
4  Andreenko  375  291113322  Bielorussie
5  Xaili  86  1065440011  Chine
```

Réponse de l'exercice 9

```
1  import pandas as pa
2  import numpy as nu
3
4  url = "https://raw.githubusercontent.com/pandas-
5                      dev/pandas/main/pandas/tests/io/data/csv/tips.csv"
6  data = pa.read_csv(url)
7  data["new_col"] = nu.where(data["size"] > 2, "Bien", "Mal")
   infos = data.tail(6)

   print(infos)
```

Résultat après exécution du code

	total_bill	tip	sex	smoker	day	time	size	new_col
238	35.83	4.67	Female	No	Sat	Dinner	3	Bien
239	29.03	5.92	Male	No	Sat	Dinner	3	Bien
240	27.18	2.00	Female	Yes	Sat	Dinner	2	Mal
241	22.67	2.00	Male	Yes	Sat	Dinner	2	Mal
242	17.82	1.75	Male	No	Sat	Dinner	2	Mal
243	18.78	3.00	Female	No	Thur	Dinner	2	Mal

FR V.7. Questions de compréhension

1) Comment obtenir des informations sur les données et pourquoi utiliser Pandas Python ?

2) En quoi l'utilité de Pandas, alors que nous avons la bibliothèque NumPy?

3) Quelles sont les principales structures de données de Pandas?

4) Quelle est la différence entre les Series et les DataFrame?

5) Quelles sont les différentes possibilités de création d'une Series dans Pandas?

6) Quelles sont les différentes possibilités de création d'une DataFrame dans Pandas?

7) Quelles sont les fonctions unique à la Series?

8) Quelles sont les fonctions unique à la DataFrame?

9) Quelles sont les fonctions appartenant en même temps à la Series qu' au DataFrame?

10) Quelle est la différence entre les 2 méthodes suivantes: *.loc()* et *.iloc()*

11) Quelles sont les différentes méthodes pour faire un tri des données?

12) Quelles sont les algorithmes s'utilisent Pandas pour faire un tri?

13) Comment lire des fichiers avec Pandas?

14) Comment pouvons nous travailler efficacement avec des données volumineuses?

15) Comment introduire les données dans un fichier Excel

16) Est-il possible d'ouvrir un fichier au format ***csv*** à partir d'un rédacteur texte simple sans courir le risque de gâcher sa structure en le sauvegardant de nouveau?

17) Est-il possible d'ouvrir un fichier au format ***xlsx (xls)*** à partir d'un rédacteur texte simple sans courir le risque de gâcher sa structure en le sauvegardant de nouveau?

Suite à la page 467

RU ЗАКЛЮЧЕНИЕ

Завершим это пособие надеждой на то, что мы предоставили читателям информацию, которую они ожидали. Мы можем смело утверждать, что полное овладение упражнениями и иллюстрированными примерами из этой книги позволит учащемуся иметь возможность реализовать несколько небольших проектов, связанных с обработкой текстов и манипуляциями с файлами. Также есть возможность проводить крупномасштабный анализ данных (пока без визуального представления) с помощью библиотеки **Pandas**. Это дает возможность вместе с библиотекой **NumPy** предоставлять гибкие, быстрые и выразительные структуры данных, чтобы упростить обработку информации. Благодаря им пользователь может легко интегрироваться с популярными файлами типа CSV, PDF, файлами Excel и многими другими.

Книга позволяет на конкретных примерах дать учащемуся возможность индивидуально понять части программирования без помощи преподавателя. В конце каждой главы предлагаются вопросы для самоконтроля, чтобы дать возможность обучающимся самостоятельно проверить полученные знания. Эти вопросы в любом случае побуждают студента сталкиваться с трудными головоломками или загадками при изучении каждого раздела данного тома.

Продолжение на странице 468

EN CONCLUSION

We will conclude this manual with the hope that we have provided readers with the information they expected. We can safely say that a complete mastery of the exercises and illustrated examples from this book allows the student to have the opportunity to already implement several small projects related to text processing and file manipulation. It is also possible to conduct a large-scale data analysis (so far without visual representation) using the **Pandas** library. This allows, together with the **NumPy** library, to provide flexible, fast and expressive data structures to simplify the processing of this information. Thanks to them, the user can easily integrate with popular files such as CSV, PDF, Excel files and many others.

This book allows using specific examples to give the student an opportunity to individually understand the stated and explained parts of programming without the help of a teacher. At the end of each chapter, comprehension or self-control questions are proposed to allow learners to test their knowledge levels themselves. These questions in any case encourage the student to face difficult puzzles or riddles when studying each section of this volume.

Continued on page 470

FR CONCLUSION

Nous allons conclure ce manuel par l'espérance d'avoir donné aux lecteurs des informations qu'ils attendaient. Nous pouvons affirmer sans crainte en disant que la maîtrise totales des exercices et des exemples illustrés de cet ouvrage permet à l'apprenant d'avoir le potentiel de pouvoir déjà réaliser quelques petits projets liés sur le traitement des textes, la manipulation des fichiers. Il est également possible de pouvoir faire un analyse des données à grande dimension grâce à la bibliothèque Pandas. Elle permet avec celle de NumPy de fournir des structures de données flexibles, rapides et expressives afin de rendre facile la manipulation de ces informations. À travers eux, l'utilisateur peut facilement avoir une intégration avec des fichiers populaires de type CSV, PDF, des fichiers d'Excel et bien d'autres.

Ce livre permet à travers des exemples bien précis, de donner la capacité à l'apprenant de comprendre individuellement les parties exposées et expliquées de la programmation, sans le truchement d'un enseignant. À la fin de chaque chapitre, des questions de contrôle ou de compréhension sont proposées pour permettre aux apprenants de tester eux-mêmes leurs niveaux de connaissances. Ces questions quelque soit les cas, incite l'apprenant à se confronter à des énigmes difficiles dans le domaine où elles ont été posées.

Suite à la page 470

СПИСОК ИСПОЛЬЗОВАННОЙ И РЕКОМЕНДУЕМОЙ ЛИТЕРАТУРЫ
LIST OF LITERATURES USED AND RECOMMENDED
LISTE DES MANUELS (SOURCES) UTILISÉS ET RECOMMANDÉS

1) Таннинг Ж., Ф. Основы программирования в Python: три в одном : учеб. пособие. В 4 т. Т. 1 / Ф. Ж. Таннинг ; Владим. гос. ун-т им. А. Г. и Н. Г. Столетовых. – Владимир : Изд-во ВлГУ, 2021. – 339 с. ISBN 978-5-9984-1228-8 (т. 1). – ISBN 978-5-9984-1227-1.

2) Жуков, Р. А. Язык программирования Python. Практикум : учебное пособие / Р.А. Жуков. — Москва : ИНФРА-М, 2022. — 216 с. + Доп. материалы [Электронный ресурс]. — (Среднее профессиональное образование). - ISBN 978-5-16-015638-5. - Текст : электронный. - URL: https://znanium.com/catalog/product/1856548 (дата обращения: 24.03.2022). – Режим доступа: по подписке.

3) Шелудько, В. М. Основы программирования на языке высокого уровня Python : учебное пособие / В. М. Шелудько ; Южный федеральный университет. - Ростов-на-Дону ; Таганрог : Издательство Южного федерального университета, 2017. - 146 с. - ISBN 978-5-9275-2649-9. - Текст : электронный. - URL: https://znanium.com/catalog/product/1021662 (дата обращения: 24.03.2022). – Режим доступа: по подписке.

4) Маккинни, У. Маккинли, У. Python и анализ данных / Уэс Маккинли ; пер. с англ. А.А. Слинкина. - Москва : ДМК Пресс, 2015. - 482 с. - ISBN 978-5-97060-315-4. - Текст : электронный. - URL: https://znanium.com/catalog/product/1027796 (дата обращения: 21.04.2022). – Режим доступа: по подписке.

5) Зыкова, Г. В. Основы программирования на языке Python : учебно-методическое пособие / Г. В. Зыкова, А. С. Попов, Т. Н. Сапуглецева ; под. ред. Г. В. Зыковой. - 2-е изд., стер. - Москва : ФЛИНТА, 2020. - 135 с. - ISBN 978-5-9765-4430-7. - Текст :

электронный. - URL: https://znanium.com/catalog/product/1860057 (дата обращения: 21.04.2022). – Режим доступа: по подписке.

6) Федоров, Д. Ю. Программирование на языке высокого уровня Python : учебное пособие для прикладного бакалавриата / Д. Ю. Федоров. — 2-е изд., перераб. и доп. — Москва : Издательство Юрайт, 2019. — 161 с. — (Бакалавр. Прикладной курс). — ISBN 978-5-534-10971-9. — Текст : электронный // Образовательная URL: https://urait.ru/viewer/programmirovanie-na-yazyke-vysokogo-urovnya-python-437489 (дата обращения: 21.04.2022).

7) Щерба А.В. Программирование на Python®: первые шаги / Щерба А.В.. — Москва : Лаборатория знаний, 2022. — 251 с. — ISBN 978-5-93208-578-3. — Текст : электронный // IPR SMART : [сайт]. — URL: https://www.iprbookshop.ru/120878.html (дата обращения: 22.04.2022). — Режим доступа: для авторизир. пользователей

8) Дроботун Н.В. Алгоритмизация и программирование. Язык Python : учебное пособие / Дроботун Н.В., Рудков Е.О., Баев Н.А.. — Санкт-Петербург : Санкт-Петербургский государственный университет промышленных технологий и дизайна, 2020. — 119 с. — ISBN 978-5-7937-1829-5. — Текст : электронный // IPR SMART : [сайт]. — URL: https://www.iprbookshop.ru/102400.html (дата обращения: 22.04.2022). — Режим доступа: для авторизир. пользователей. - DOI: https://doi.org/10.23682/102400

9) Шелудько В.М. Основы программирования на языке высокого уровня Python : учебное пособие / Шелудько В.М.. — Ростов-на-Дону, Таганрог : Издательство Южного федерального университета, 2017. — 146 с. — ISBN 978-5-9275-2649-9. — Текст : электронный // IPR SMART : [сайт]. — URL: https://www.iprbookshop.ru/87461.html (дата обращения: 22.04.2022). — Режим доступа: для авторизир. пользователей

RU Интернет-источники EN URL Online FR Les sources

1) La documentation du package PDFminer.six; URL: https://pdfminersix.readthedocs.io/en/latest/ (дата обращения: 21.04.2022).

2) La page d'information de Mike Driscoll - Programmeur en Python/Information sur la bibliotheque PDFMiner URL : https://www.blog.pythonlibrary.org/2018/05/03/exporting-data-from-pdfs-with-python/ (дата обращения: 21.04.2022).

3) binPress - Information sur le développement web, le marketing en ligne d'affaires / La manipulation de fichiers Pdf avec Python/ URL : https://www.binpress.com/manipulate-pdf-python/ (дата обращения: 21.04.2022).

4) Digitology. tech - Материалы по программированию / Создание и изменение PDF файлов в Python URL: https://digitology.tech/posts/sozdanie-i-izmenenie-pdf-fajlov-v-python/ (дата обращения: 21.04.2022).

5) Github - Informations sur la bibliothèque PyDF2/ URL: https://github.com/mstamy2/PyPDF2/commit/1273824c0f8d708f1dbd5872e2216b75e76d46c3 (дата обращения: 21.04.2022).

6) The documentation of the programming language Python.org/ Regular expression operations URL: https://docs.python.org/3.10/library/re.html (дата обращения: 21.04.2022).

7) Информация об алгоритмах: URL:
https://fr.wikipedia.org/wiki/Algorithme_de_tri
https://ru.wikipedia.org/wiki/Алгоритм_сортировки
https://en.wikipedia.org/wiki/Sorting_algorithm
https://www.geeksforgeeks.org/python-programming-language/
https://www.geeksforgeeks.org/quick-sort/
https://www.geeksforgeeks.org/selection-sort/
https://www.geeksforgeeks.org/insertion-sort/
https://www.geeksforgeeks.org/merge-sort/
https://www.geeksforgeeks.org/heap-sort/

https://www.geeksforgeeks.org/bubble-sort/
https://www.geeksforgeeks.org/radix-sort/
https://www.geeksforgeeks.org/counting-sort/
https://www.geeksforgeeks.org/shellsort/
https://www.geeksforgeeks.org/comb-sort/
https://www.geeksforgeeks.org/pigeonhole-sort/
(дата обращения: 21.04.2022).

8) pandas - Pandas Data Analysis Library / URL: https://pandas.pydata.org/ (дата обращения: 21.04.2022).

9) pandas - Getting started: Installation / URL: https://pandas.pydata.org/docs/getting_started/index.html (дата обращения: 21.04.2022).

10) François Chollet, Deep Learning with Python, Second Edition/ October 2021 ISBN 9781617296864 504 pages / URL: https://www.manning.com/books/deep-learning-with-python-second-edition#toc (дата обращения: 21.04.2022).

Учебное электронное издание (Educational electronic publication)

ТАННИНГ Жиогап Фирмэн
TANGNING Jiogap Firmin

ОСНОВЫ ПРОГРАММИРОВАНИЯ В PYTHON: ТРИ В ОДНОМ

THE BASICS OF PYTHON PROGRAMMING: THREE IN ONE

Учебное пособие (The course book)

Том 2 (Volume II)

Издается в авторской редакции (Published in the author's edition)

Системные требования: Intel от 1,3 ГГц; Windows XP/7/8/10/11, Linux, Mac OS; Adobe Reader; дисковод CD-ROM.

Тираж 25 экз.

Владимирский государственный университет
имени Александра Григорьевича и Николая Григорьевича Столетовых
Изд-во ВлГУ
rio.vlgu@yandex.ru

Кафедра информатики и защиты информации
tajifirmin2@yahoo.com