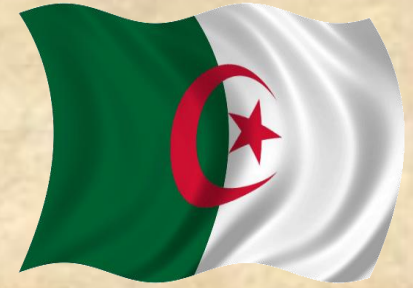




جامعة محمد بوضياف المسيلة
كلية العلوم الاقتصادية
قسم العلوم الاقتصادية



المستوى: أولى جذع مشترك

محاضرات في مقياس:

Programming Fundamentals with Python 01

أساسيات البرمجة على بايثون 01

أ. د : رابح بلعباس
2026-2025

أولاً:
مقدمة في Python
Python Introduction

الاختيار بين البرمجيات

Python software



R software



EvIEWS, SPSS, Stata



لغة البرمجة بايثون (Python—The Programming Language)

تم إنشاء لغة البرمجة بايثون بواسطة **غيدو فان روسوم (Guido Van Rossum)** سنة **1991**،

تتميز لغة بايثون بمجموعة من الصفات التالية:

1. مفسّرة (**Interpreted**)
2. منقولة (**Portable**)
3. كائنية التوجه (**Object-oriented**)
4. تفاعلية (**Interactive**)
5. ذات واجهات متعددة (**Interfaced**)
6. مفتوحة المصدر (**Open source**)
7. سهولة الفهم والاستخدام (**Easy to understand and use**)

What can we do with Python?

What can we do with Python?

Python is used for:

- Web Development: Frameworks like Django, Flask.
- Data Science and Analysis: Libraries like Pandas, NumPy, Matplotlib.
- Machine Learning and AI: TensorFlow, PyTorch, Scikit-learn.
- Automation and Scripting: Automate repetitive tasks.
- Game Development: Libraries like Pygame.
- Web Scraping: Tools like BeautifulSoup, Scrapy.
- Desktop Applications: GUI frameworks like Tkinter, PyQt.
- Scientific Computing: SciPy, SymPy.
- Internet of Things (IoT): MicroPython, Raspberry Pi.
- DevOps and Cloud: Automation scripts and APIs.
- Cybersecurity: Penetration testing and ethical hacking tools.

لماذا Python؟

- **سهولة التعلم:** تشبه اللغة الإنجليزية.
- **متعددة الاستخدامات:** علوم البيانات، الذكاء الاصطناعي، الاقتصاد القياسي.
- **مفتوحة المصدر:** مجانية ومدعومة بمكتبات قوية
(NumPy, Pandas, Matplotlib)
- **شعبية عالمية:** مستخدمة في هارفارد، MIT، Google، NASA

Famous Application Built using Python

تطبيقات شهيرة تم تطويرها باستخدام بايثون

- **يوتيوب (YouTube):** أكبر منصة لمشاركة الفيديوهات في العالم تستخدم بايثون لميزات مثل بث الفيديو والخدمات الخلفية.
- **إنستغرام (Instagram):** يعتمد هذا التطبيق الشهير على بساطة بايثون لتوسيع نطاق الخدمة والتعامل مع ملايين المستخدمين.
- **سبوتيفاي (Spotify):** يُستخدم بايثون في الخدمات الخلفية وتعلم الآلة لتخصيص توصيات الموسيقى.
- **دروب بوكس (Dropbox):** خدمة استضافة الملفات تستخدم بايثون في كل من عميل سطح المكتب والعمليات على الخادم.
- **نتفليكس (Netflix):** بايثون يشغل المكونات الأساسية لمحرك التوصيات وأنظمة توصيل المحتوى (CDN).
- **جوجل (Google):** بايثون هي واحدة من اللغات الأساسية في جوجل لأغراض الزحف على الويب، الاختبار، وتحليل البيانات.
- **أوبر (Uber):** يساعد بايثون أوبر في التعامل مع التسعير الديناميكي وتحسين المسارات باستخدام تعلم الآلة.
- **بنترست (Pinterest):** يُستخدم بايثون لمعالجة وتخزين كميات هائلة من بيانات الصور بكفاءة.

أشهر بيئات العمل في بايثون

أشهر بيئات العمل في بايثون				
الرقم	بيئة العمل	المميزات	العيوب	الاستخدام الأنسب
1	Command Line / Terminal	خفيفة جداً، لا تحتاج تثبيت، سريعة للتنفيذ الفوري	لا تحتوي على واجهة رسومية، صعبة للمبتدئين	تجربة الأوامر السريعة أو البرامج الصغيرة
2	IDLE	تأتي مع بايثون افتراضياً، سهلة وبسيطة، مثالية للتعلم	واجهة محدودة، لا تدعم المشاريع الكبيرة	تعليم المبتدئين والمشاريع الصغيرة
3	Jupyter Notebook	تفاعلية، تسمح بدمج الكود مع النصوص والرسوم، مثالية للتحليل	ليست مناسبة للتطبيقات الكاملة، قد تكون بطيئة أحياناً	تحليل البيانات، التعليم، البحث الأكاديمي
4	Google Colab	مجانية، تعمل على السحابة، تدعم GPU/TPU، لا تحتاج تثبيت	تتطلب اتصال إنترنت، سعة تخزين محدودة	الذكاء الاصطناعي والتعلم العميق، المشاريع البحثية
5	Spyder	واجهة علمية تشبه MATLAB، أدوات تحليل قوية، تكامل مع NumPy و pandas	ليست مناسبة لتطوير الويب، واجهتها تقليدية قليلاً	الباحثين والعاملين في علم البيانات
6	Visual Studio Code (VS Code)	خفيف، متعدد اللغات، يدعم الإضافات، تصحيح الأخطاء ممتاز	يحتاج إعداد أولي، بعض الميزات تتطلب إضافات	المشاريع المتوسطة والكبيرة، الذكاء الاصطناعي، الويب
7	PyCharm	قوي جداً، تكامل ممتاز مع Git وقواعد البيانات، احترافي	ثقيل نوعاً ما، النسخة الكاملة مدفوعة	المطورين المحترفين ومشاريع الشركات
8	Anaconda Distribution	يحتوي على أدوات متعددة (Jupyter, Spyder)، إدارة حزم سهلة، مثالي للعلماء	حجم كبير، يحتاج مساحة تخزين	التحليل الإحصائي، علم البيانات، التعلم الآلي

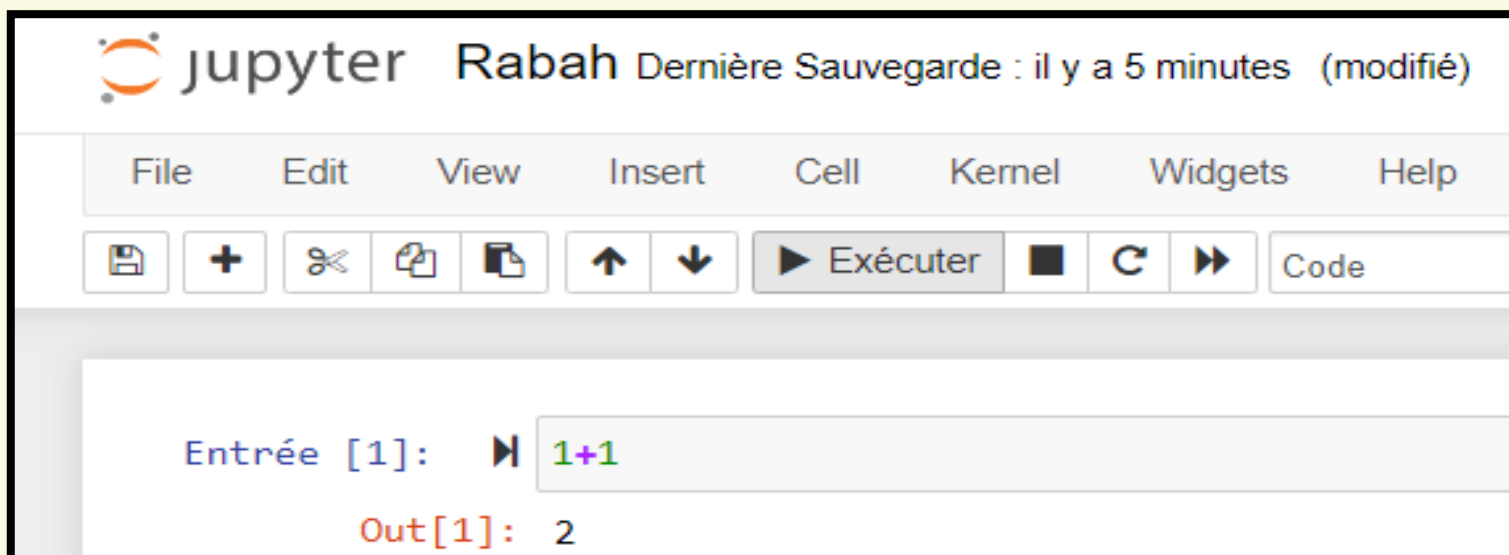
بيئة العمل في بايثون

Jupyter Notebook

- **Jupyter Notebook:** الأكثر استخدامًا في التعليم والبحث،
يسمح بكتابة الكود + التعليقات + الرسوم.
- بيئة احترافية لدعم المشاريع الكبيرة.
- أفضل بيئة للمبتدئين.

بيئة العمل في بايثون

- يُعد **Jupyter Notebook** أحدث تطوّر في بيئة البرمجة التفاعلية. في الواقع، من خلال **Jupyter Notebook** يمكنك دمج الشيفرة القابلة للتنفيذ، والنصوص، والمعادلات، والصور، والرسوم المتحركة في مستند ويب واحد. تُستخدم هذه الأداة في العديد من الأغراض، مثل:
- العروض التقديمية (**Presentations**)
 - الدروس التعليمية (**Tutorials**)
 - اكتشاف الأخطاء وتصحيحها (**Debugging**)
 - وغيرها من التطبيقات التعليمية والبحثية المتنوعة.



ثانياً: كيفية تثبيت
Python باستخدام
anaconda
How to install Python

كيفية تثبيت Python

How to install Python



anaconda download

أكتب **anaconda download**
على صفحة **Google**

All

Videos

Images

Shopping

News

More

Tools

For Windows 10

Individual Edition

For Windows 11

Jupyter Notebook

1997

Wallpaper

About 46,300,000 results (0.26 seconds)



Anaconda

<https://www.anaconda.com> › download

ثم اختر هذا الموقع

Download Anaconda Distribution ✓

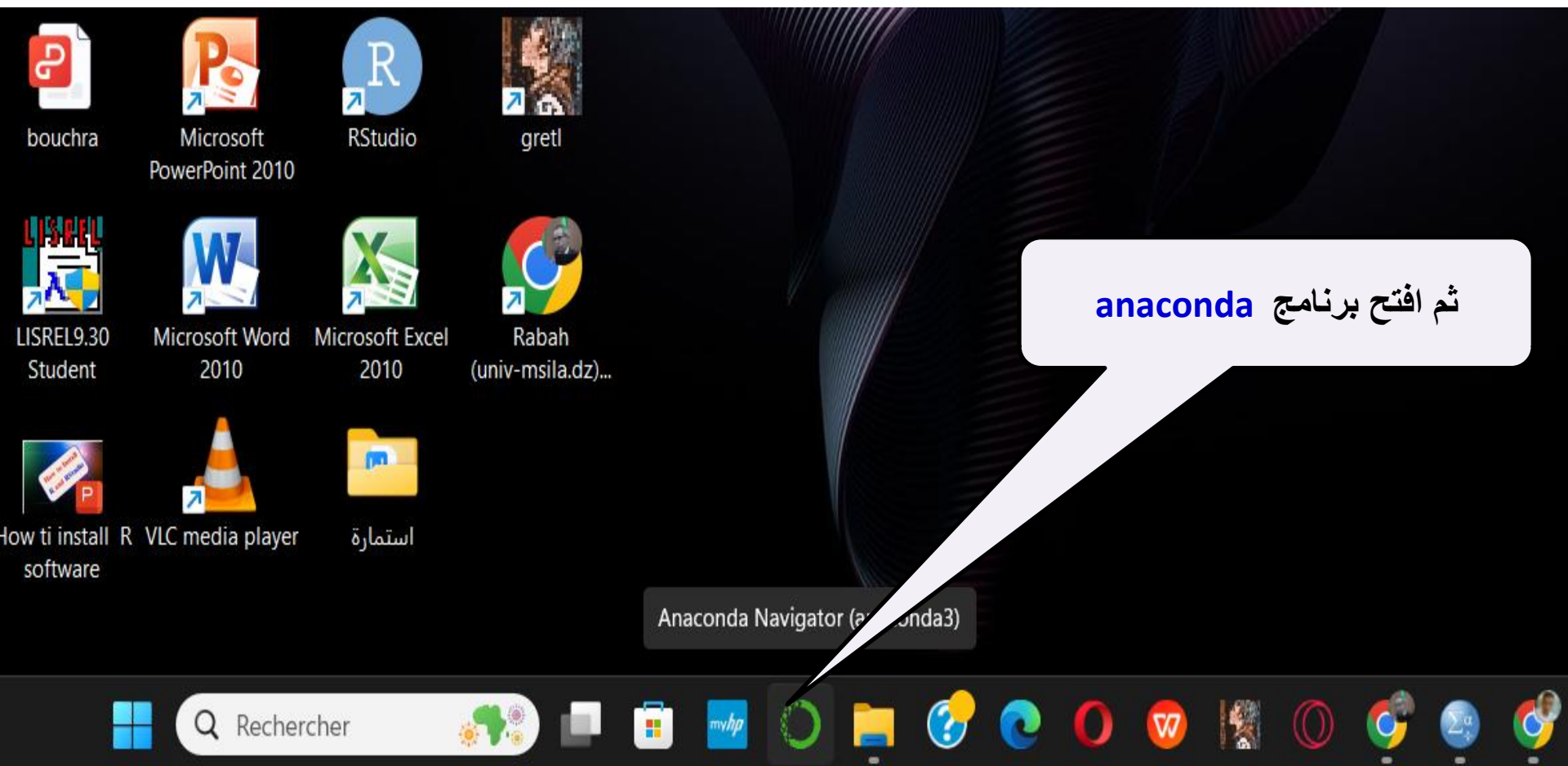
Download Anaconda's open-source Distribution today. Discover the easiest way to perform Python/R data science and machine learning on a single machine.

[Navigator](#) · [About Anaconda](#) · [Pricing](#) · [Partners](#)

ثم حمل anaconda وثبته على حاسوبك

كيفية تثبيت Python

How to install Python



Home

Environments

Learning

Community

All applications

on base (root)

Channels



PyCharm Professional

The Python IDE for data science. It combines the interactivity of Jupyter notebooks with intelligent Python coding assistance, Anaconda support, and scientific libraries.

Install



Anaconda AI Navigator

Access various large language models (LLMs) curated by Anaconda, and start leveraging secure local AI today.

Install



Anaconda Cloud Notebooks

Cloud-hosted notebook service from Anaconda. Launch a preconfigured environment with hundreds of packages and store project files with persistent cloud storage.

Install



Anaconda Cloud Notebooks

Cloud-hosted notebook service from Anaconda. Launch a preconfigured environment with hundreds of packages and store project files with persistent cloud storage.

Launch



anaconda_prompt

1.1.0
Opens a terminal instance with conda activated (requires menuinst 2.1.1 or greater).

Launch



JupyterLab

4.2.5
An extensible environment for interactive and reproducible computing, based on the Jupyter Notebook and Architecture.

Launch



Jupyter Notebook

7.2.2
Web-based, interactive computing notebook environment. Edit and run code, view human-readable docs while describing your data analysis.

Launch



Qt Console

5.5.1
PyQt GUI that supports inline figures, proper multiline editing with syntax highlighting, graphical calltips, and more.

Launch



Spyder

5.5.1
Scientific Python Development Environment. Powerful Python IDE with advanced editing, interactive testing, debugging and introspection features

Launch



EduBlocks

Web-based coding platform from Anaconda designed for students. Learn Python coding through an interactive, block-based visual environment.

Launch



watsonx™

IBM watsonx

IBM watsonx is an enterprise-ready AI platform including a data store, model builder, and AI model management and monitoring.

Launch



ORACLE Cloud Infrastructure

Oracle Data Science Service
OCI Data Science offers a machine learning platform to build, train, manage, and deploy your machine learning models on the cloud with your favorite open-source tools

Launch



PyScript

Code and share Python in the Browser. A vibrant community of makers, builders, and learners.



PythonAnywhere

Host, run, and code Python in the cloud! Get started for free.



CMD.exe Prompt

0.1.1
Run a cmd.exe terminal with your current environment from Navigator activated



console_shortcut_miniconda

0.1.1
Anaconda Powershell Prompt



Glueviz

1.2.4
Multidimensional data visualization across files. Explore relationships within and across datasets.



Orange 3

3.36.2
Component based data mining framework. Data visualization and data analysis for machine learning.

ثم اختر
Jupyter
بالضغط هنا

يمكنك اختيار منصة أخرى

في هذه المحاضرات سنختار منصة Jupyter
لأنها من أكثر المنصات شيوعاً في تفعيل أكواد Python

Files

Running

Clusters

Select items to perform actions on them.

Upload

New ▾



<input type="checkbox"/> 0 ▾	 /	Name ▾	Last Modified	File size
<input type="checkbox"/>	 anaconda3		il y a 3 mois	
<input type="checkbox"/>	 Contacts		il y a un an	
<input type="checkbox"/>	 Documents		il y a 4 mois	
<input type="checkbox"/>	 Downloads		il y a 3 heures	
<input type="checkbox"/>	 Favorites		il y a un an	
<input type="checkbox"/>	 Links		il y a un an	
<input type="checkbox"/>	 Music		il y a un an	
<input type="checkbox"/>	 OneDrive		il y a un an	

كيفية انشاء صفحة محرر بيانات على Python

How to create Python Notebook



Quit

Logout

Files

Running

Clusters

Select items to perform actions on them.

0 /

☐ anaconda3

☐ Contacts

☐ Documents

☐ Downloads

☐ Favorites

☐ Links

Upload

New



Notebook:

Name

Python 3 (ipykernel)

Other:

Text File

Folder

Terminal

Créer un nouveau notebook avec Python 3 (ipykernel)

اضغط على New

ثم اختر Python 3

كيفية تسمية صفحة محرر بيانات على Python

How to renamed Python Notebook

The image shows the Jupyter Notebook interface with the 'Renommer le Notebook' dialog box open. The dialog box contains the following elements:

- Renommer le Notebook** (Title)
- Saisir le nouveau nom du notebook** (Label)
- Rabah** (Text input field)
- Annuler** (Cancel button)
- Renommer** (Rename button)

Callout boxes in Arabic provide instructions:

- اضغط على Renommer** (Click on Renommer)
- انقر هنا مرتين بسرعة لإعادة تسمية الملف** (Click here twice quickly to rename the file)
- سأختار اسمي** (I will choose my name)

معالجة البيانات على Python

Data Processing in Python

The image shows the JupyterLab interface. At the top, the Jupyter logo is followed by the text "Rabah Dernière Sauvegarde : il y a 5 minutes (modifié)". Below this is a menu bar with "File", "Edit", "View", "Insert", "Cell", "Kernel", "Widgets", and "Help". Under the menu bar is a toolbar with icons for saving, adding, deleting, copying, pasting, undo, redo, and a button labeled "Exécuter" (Execute). To the right of the toolbar is a "Code" input field.

In the main workspace, there is a code cell. The prompt "Entrée [1]:" is followed by a right-pointing arrow and the code "1+1". Below the code cell, the output "Out[1]: 2" is displayed.

Three callout boxes provide explanations:

- A box pointing to the "Exécuter" button contains the text: "ثم اضغط على **Executer**" (Then click on **Executer**).
- A box pointing to the code "1+1" contains the text: "أكتب 1+1" (Write 1+1).
- A box pointing to the output "2" contains the text: "ستظهر النتيجة 2" (The result 2 will appear).

رابعًا: أمثلة تطبيقية

المثال 1: أول برنامج

```
print("Hello, World!")
```

النتيجة:

Hello, World!

المثال 2:

```
print("السلام عليكم")
```

النتيجة:

السلام عليكم

رابعًا: أمثلة تطبيقية

jupyter belabbas Python 01 Last Checkpoint: 6 days ago

File Edit View Run Kernel Settings Help

Save + Copy Paste Run Interrupt Restart Code ▾

[1]: `print("Hello, World!")`

Hello, World!

[2]: `print("السلام عليكم")`

السلام عليكم

ثم اضغط على
Executer

اكتب هنا
`print("Hello, World!")`

ستظهر النتيجة

المثال 2: استخدام Python في الاقتصاد

برنامج لحساب متوسط دخل الفرد:

```
gdp = 2000000000 # الناتج المحلي الإجمالي  
population = 100000000 # عدد السكان  
gdp_per_capita = gdp / population  
print("GDP per capita:", gdp_per_capita)
```

النتيجة:

GDP per capita: 20.0

المثال 2: استخدام Python في الاقتصاد

ثم اضغط على
Executer

أكتب كود بايثون

jupyter belabbas Python 01 Last Checkpoint: 6 days ago

File Edit View Run Kernel Settings Help

Save + Copy Paste Run Interrupt Restart Code

```
[1]: gdp = 200000000 # الناتج المحلي الإجمالي  
      population = 10000000 # عدد السكان  
      gdp_per_capita = gdp / population  
      print("GDP per capita:", gdp_per_capita)
```

GDP per capita: 20.0

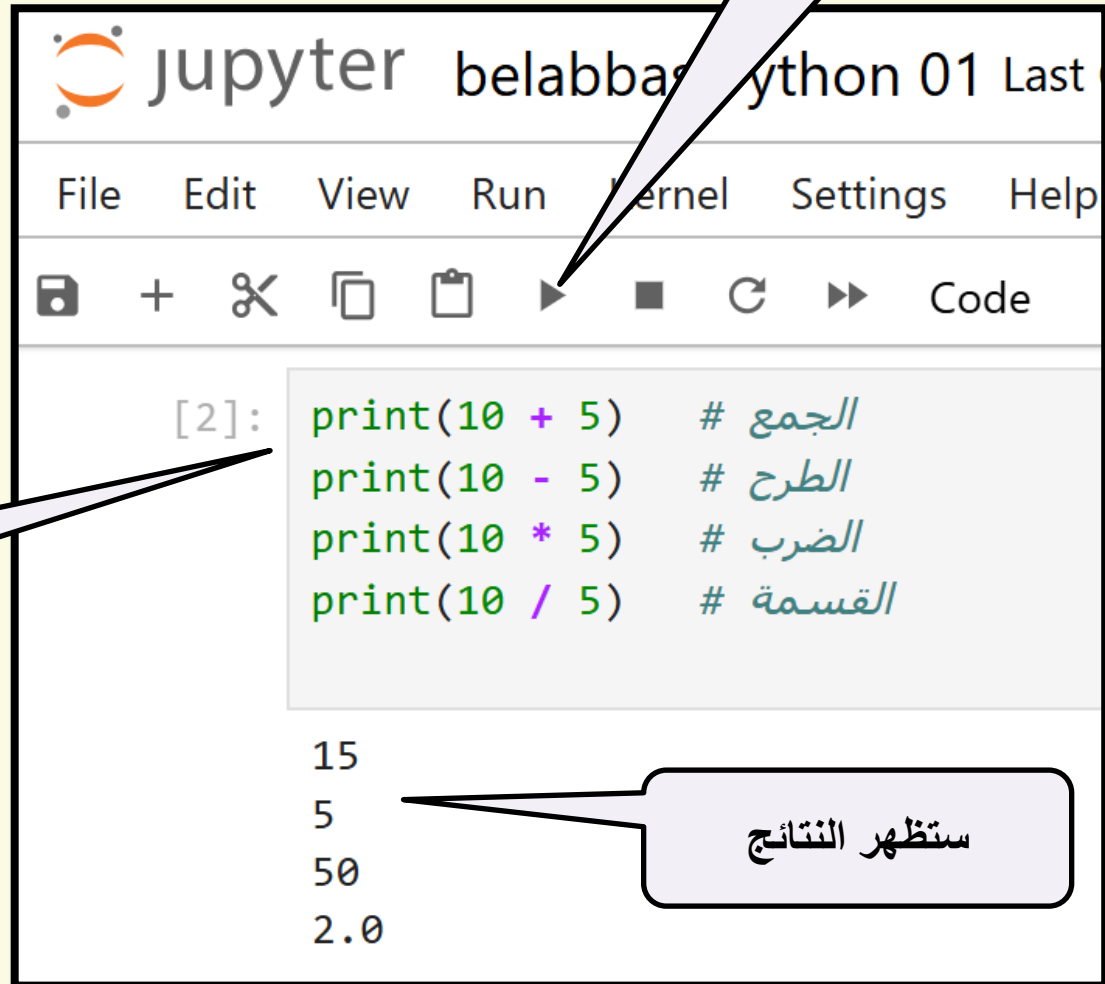
ستظهر النتائج

المثال 3: العمليات البسيطة

`print(10 + 5)` # الجمع
`print(10 - 5)` # الطرح
`print(10 * 5)` # الضرب
`print(10 / 5)` # القسمة

أكتب الأكواد هنا

ثم اضغط على
Executer



The screenshot shows a Jupyter Notebook window with the title 'belabbas python 01 Last'. The menu bar includes 'File', 'Edit', 'View', 'Run', 'Kernel', 'Settings', and 'Help'. The toolbar contains icons for saving, adding, deleting, copying, pasting, running (a play button), and other functions. A callout points to the run button. The code cell, labeled '[2]:', contains the following Python code:

```
print(10 + 5) # الجمع
print(10 - 5) # الطرح
print(10 * 5) # الضرب
print(10 / 5) # القسمة
```

Below the code cell, the output is displayed:

```
15
5
50
2.0
```

A callout points to the output, indicating that the results are displayed.

ستظهر النتائج

خامسًا: نشاط للطلاب

1. افتح Jupyter Notebook.
2. اكتب برنامجًا يطبع:
 - o اسمك.
 - o عمرك.
 - o تخصصك الجامعي.
3. احسب معدل نمو الناتج المحلي إذا كان:
 - o GDP السنة الماضية = 150000
 - o GDP هذه السنة = 180000

إنشاء بيانات اقتصادية تجريبية

إنشاء بيانات سنة - نمو الناتج المحلي

إنشاء بيانات سنة - نمو الناتج المحلي

```
import numpy as np
```

```
import pandas as pd
```

```
years = np.arange(2015, 2025)
```

```
gdp_growth = np.array([2.1, 2.3, 1.9, 0.8, -2.0, 3.1, 4.2, 3.7, 2.9, 3.0])
```

```
data = pd.DataFrame({'Year': years, 'GDP_Growth': gdp_growth})
```

```
data
```

إنشاء بيانات اقتصادية تجريبية

إنشاء بيانات سنة - نمو الناتج المحلي

```
import numpy as np
```

```
import pandas as pd
```

```
years = np.arange(2015, 2025)
```

```
gdp_growth = np.array([2.1, 2.3, 1.9, 0.8, -2.0, 3.1, 4.2, 3.7, 2.9, 3.0])
```

```
data = pd.DataFrame({'Year': years, 'GDP_Growth': gdp_growth})
```

```
data
```


إنشاء بيانات اقتصادية تجريبية

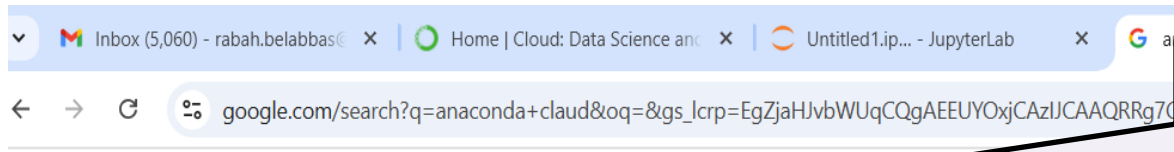
[7]:

	Year	GDP_Growth
0	2015	2.1
1	2016	2.3
2	2017	1.9
3	2018	0.8
4	2019	-2.0
5	2020	3.1
6	2021	4.2
7	2022	3.7
8	2023	2.9
9	2024	3.0

ثالثاً: كيفية العمل على
Anaconda cloud
Anaconda How to work on
cloud


كيفية العمل على **Anaconda cloud**

How to work on **Anaconda cloud**



1. أكتب **anaconda Cloud** على صفحة **Google**

These are results for **anaconda cloud**
Search instead for **anaconda cloud**

 Anaconda Cloud
<https://anaconda.cloud>

Anaconda Cloud ✓

Anaconda, your home for data science and Python.

[Repo.anaconda.com Scheduled...](#) ✓

Welcome to Anaconda Cloud. Your home for data science ...

[We use cookies](#) ✓

Welcome to Anaconda Cloud. Your home for data science ...

[More results from anaconda.cloud »](#)

2. ثم اختر هذا الموقع

Get AI answer

Automatically pauses after inactivity ⓘ

كيفية العمل على Anaconda cloud

Anaconda cloud How to work on



anaconda.cloud will be merging into anaconda.com soon to give you a more streamlined experience. Learn more in our [FAQ](#) X

Welcome Back, rabah!



Launch a Notebook

Get started with a Jupyter Notebook



Take a Course

Build data science and AI skills



Join the Community

Engage with others in our community

2. ثم اختر Jupyter

Explore Anaconda



Notebooks



Learning



Community



Docs



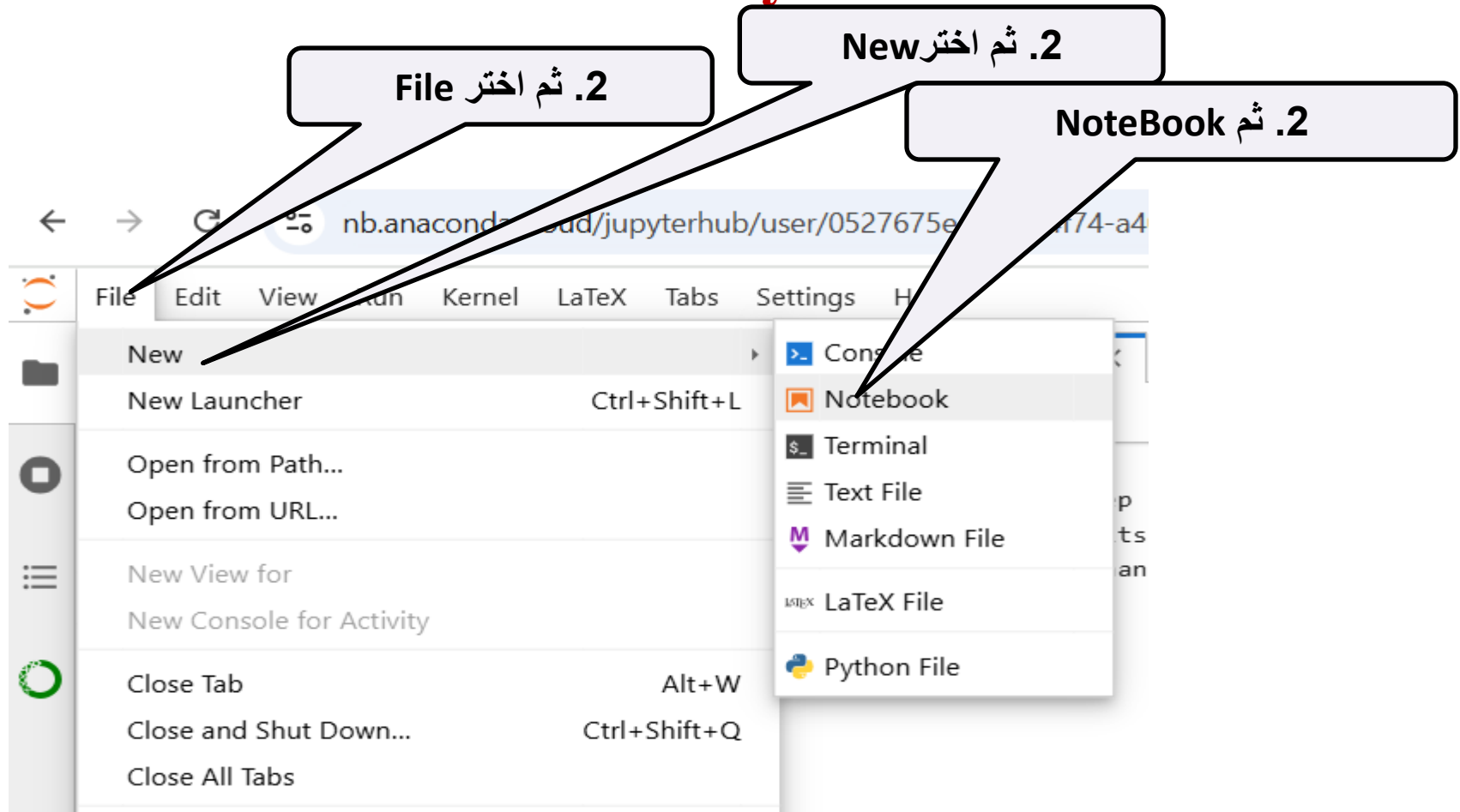
AI Navigator

Anaconda cloud كيفية العمل على

How to work on Anaconda cloud

كيفية انشاء صفحة محرر بيانات على Python

How to create Python Notebook



كيفية العمل على **Anaconda cloud**

How to work on **Anaconda cloud**

كيفية انشاء صفحة محرر بيانات على **Python**

How to create **Python Notebook**

الآن بنفس الطريقة سيتم تفعيل أكواد البايثون

سترى أن النتيجة هي 2

ثم اضغط على run

The screenshot displays the Anaconda Cloud web interface. On the left, a sidebar shows a file explorer with a search bar and a list of files including '1st day Code Ex...', 'datadz_updated...', and 'datadz.xlsx'. The main area shows a notebook titled 'Untitled3.ipynb' with a code cell containing the expression `1+1`. Below the code cell, the output is displayed as `[1]: 2`. A 'Run' button (a green play icon) is visible to the right of the code cell. A 'Share' button is also present in the top right corner of the notebook area. The interface includes a menu bar with options like 'File', 'Edit', 'View', 'Run', 'Kernel', 'LaTeX', 'Settings', and 'Help'.

أكتب 1+1 هنا

Anaconda cloud كيفية العمل على

How to work on Anaconda cloud

كيفية استيراد ملف بيانات من Excel

How to import data from Excel file

ثم اضغط هنا

nb.anaconda.cloud/jupyterhub/user/0527...-434e-4f74-a408-f221fea0e027/lab/tree/Untitled3.ipynb

File Edit View Run Kernel LaTeX Labs Settings Help CPU L

Filter files by name Upload Files

Name	Last Modified
1st day Code Ex...	1 hour ago
datadz_updated...	1 hour ago
datadz.xlsx	27 days ago

Console 1 Untitled3.ipynb

Code

Share

```
[1]: 1+1
```

```
[1]: 2
```

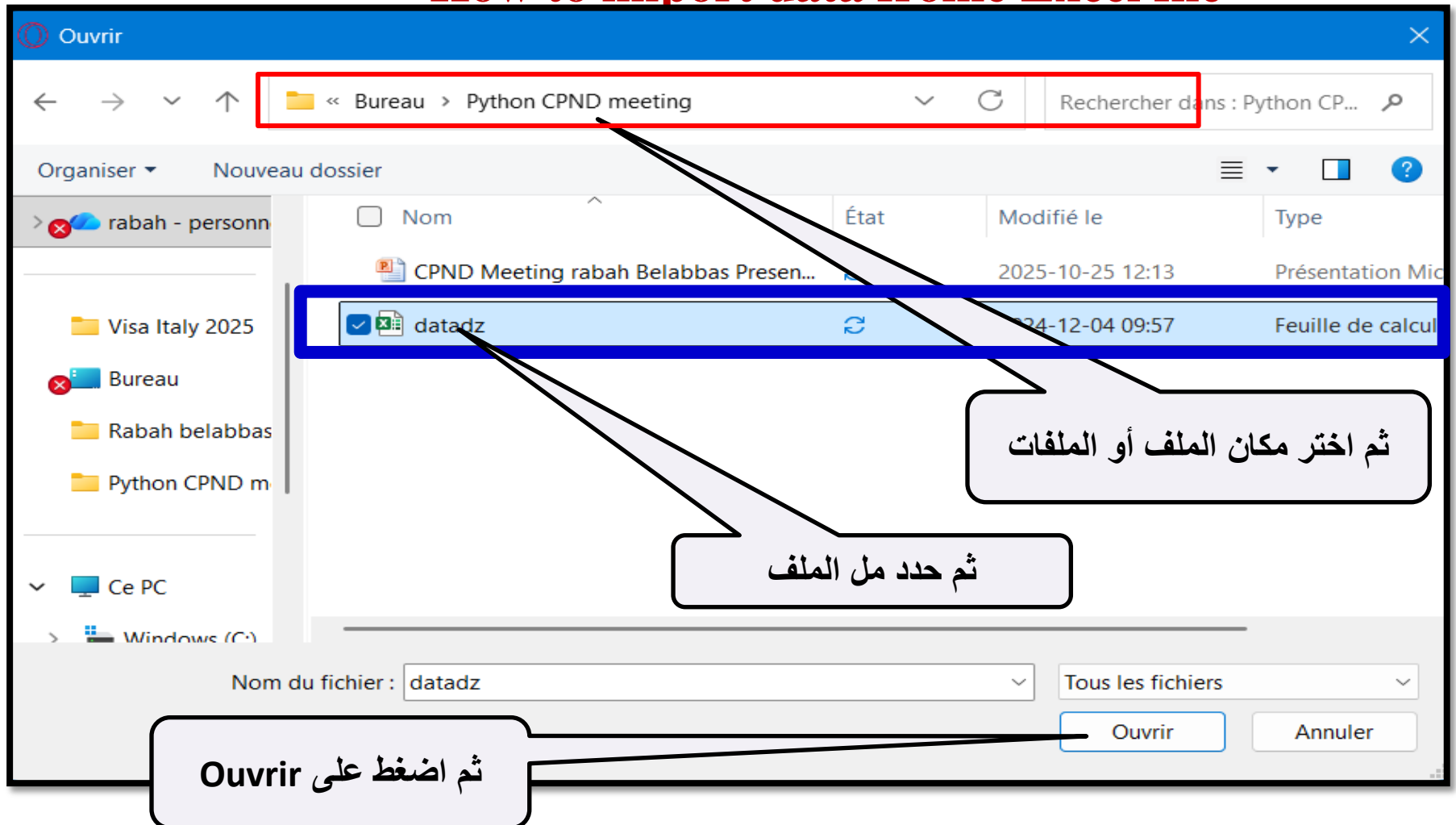
```
[ ]:
```

كيفية العمل على Anaconda cloud

How to work on Anaconda cloud

كيفية استيراد ملف بيانات من Excel

How to import data from Excel file

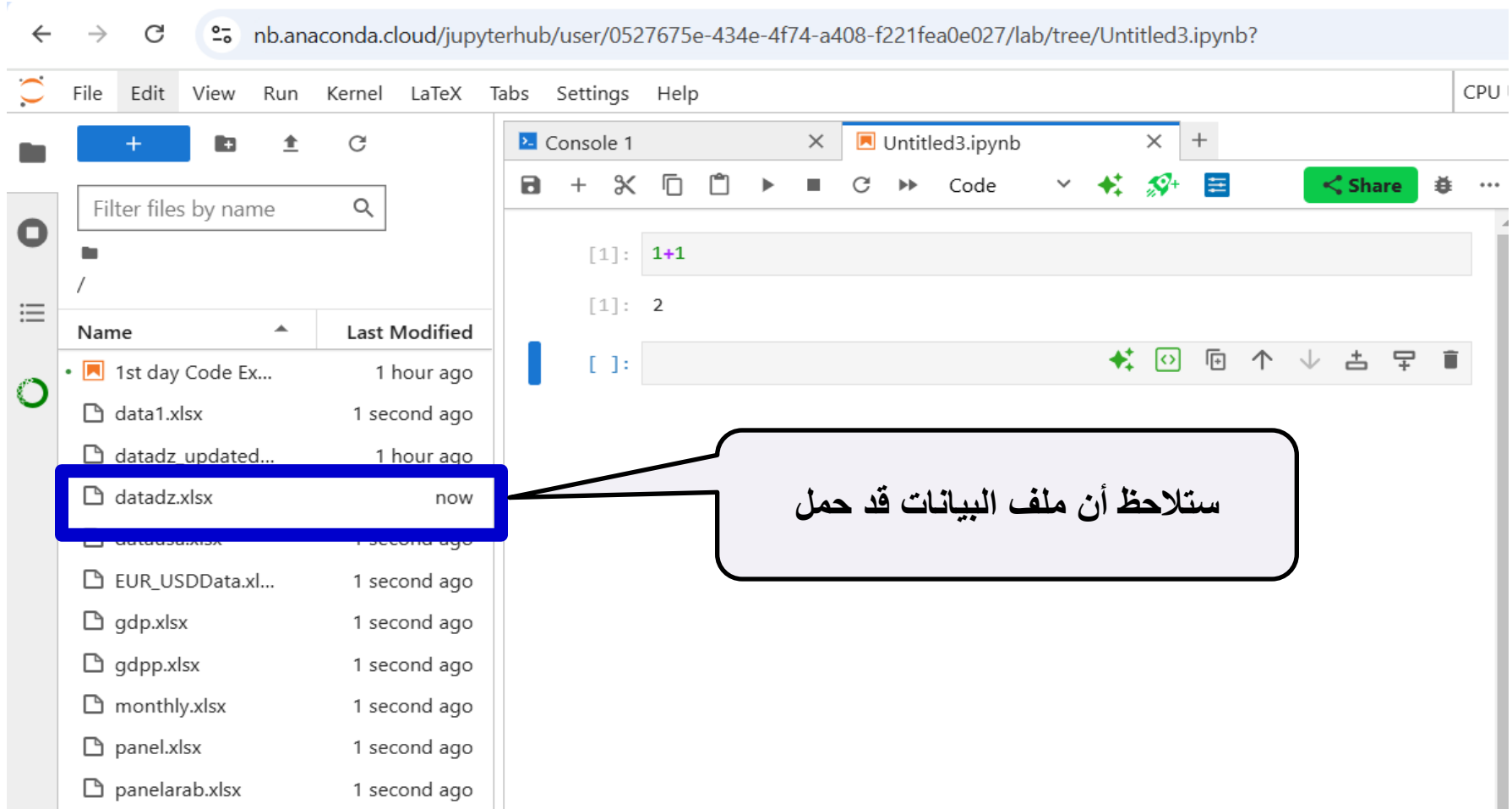


كيفية العمل على Anaconda cloud

How to work on Anaconda cloud

كيفية استيراد ملف بيانات من Excel

How to import data from Excel file



The screenshot shows the Anaconda Cloud Jupyter interface. On the left, a file browser displays a list of files. The file 'datadz.xlsx' is highlighted with a blue box. A callout box points to it with the text 'ستلاحظ أن ملف البيانات قد حمل' (You will notice that the data file has been uploaded).

Name	Last Modified
1st day Code Ex...	1 hour ago
data1.xlsx	1 second ago
datadz updated...	1 hour ago
datadz.xlsx	now
dataus.xlsx	1 second ago
EUR_USDData.xl...	1 second ago
gdp.xlsx	1 second ago
gdpp.xlsx	1 second ago
monthly.xlsx	1 second ago
panel.xlsx	1 second ago
panelarab.xlsx	1 second ago

The right side of the interface shows a code editor with a console output. The console output shows the result of a calculation: `[1]: 1+1` and `[1]: 2`. The code editor has a toolbar with various icons for file operations and a 'Share' button.

Anaconda cloud كيفية العمل على

How to work on Anaconda cloud

كيفية استيراد ملف الأكواد

How to import script files

بنفس الطريقة السابقة اضغط هنا

The screenshot displays the Anaconda Cloud Jupyter interface. On the left, a file browser sidebar shows a list of files and folders. The 'Upload' button (represented by an upward arrow icon) is highlighted with a callout box containing the text 'بنفس الطريقة السابقة اضغط هنا' (Use the same method as before, click here). The main area shows a Jupyter notebook with two code cells. The first cell contains the code `1+1` and the second cell contains the code `2`. The interface includes a top menu bar with options like File, Edit, View, Run, Kernel, LaTeX, and Help. The bottom status bar shows the CPU usage.

Name	Last Modified
1st day Code Ex...	2 hours ago
data1.xlsx	38 minutes ago
datadz_updated...	2 hours ago
datadz.xlsx	38 minutes ago
datausa.xlsx	38 minutes ago
EUR_USDDData.xl...	38 minutes ago
gdp.xlsx	38 minutes ago
gdpp.xlsx	38 minutes ago
monthly.xlsx	38 minutes ago

كيفية العمل على Anaconda cloud

How to work on Anaconda cloud

كيفية استيراد ملف الأكواد

How to import script files

ثم اختر مكان الملف أو الملفات

Organiser Nouveau dossier

Accueil

Galerie

OneDrive - Perso

Bureau

Téléchargem

Documents

Images

Musique

Vidéos

profetionnal acc

data

Rabah Belabbas

data

<input checked="" type="checkbox"/>	1st Day Data Processing.ipynb	2025-03-03 18:20	Fichier IPYNB	34 Ko
<input checked="" type="checkbox"/>	1st day data visualisation.ipynb	2025-03-03 18:20	Fichier IPYNB	8 902 Ko
<input checked="" type="checkbox"/>	1st Day Statistical Analysis.ipynb	2025-03-03 18:20	Fichier IPYNB	17 Ko

Nom du fichier : "1st Day Statistical Analysis.ipynb" "1st Day Data Processing.ipynb" "1st day data visualisation.ipynb"

All Files

Ouvrir Annuler

ثم اضغط على Ouvrir

ثم حدد مل الملفاة

كيفية العمل على Anaconda cloud

How to work on Anaconda cloud

كيفية استيراد ملف الأكواد

How to import script files

The screenshot shows the Anaconda Cloud Jupyter interface. The left sidebar displays a file browser with a search bar and a list of files. The right pane shows a Jupyter notebook with a console and code editor.

File List:

Name	Last Modified
1st Day Data Pr...	1 minute ago
1st day data vis...	21 seconds ago
1st Day Statistic...	1 minute ago
data.xlsx	18 minutes ago
datadz_updated...	2 hours ago
datadz.xlsx	46 minutes ago
datausa.xlsx	46 minutes ago
EUR_USDData.xl...	46 minutes ago
gdp.xlsx	46 minutes ago
gdpp.xlsx	46 minutes ago
monthly.xlsx	46 minutes ago
panel.xlsx	46 minutes ago
panelarab.xlsx	46 minutes ago

Code Editor:

```
[1]: 1+1
```

```
[1]: 2
```

```
[ ]:
```

Callout Box:

ستلاحظ أن الملفاة قد حملت وبالتالي يمكنك فتح أي ملف ترغب في أن تشتغل عليه

كيفية العمل على Anaconda cloud

How to work on Anaconda cloud

كيفية تحميل ملف الأكواد

How to download script files

The screenshot shows the JupyterLab interface with a file browser on the left and a context menu open over a file named 'data visialusation.ipynb'. The context menu includes options like 'Open', 'Rename', 'Delete', 'Copy', 'Paste', 'Duplicate', 'Download', 'Copy Download Link', 'Copy Path', 'Copy Shareable Link', 'New File', 'New Notebook', and 'New Folder'. A callout box points to the 'Download' option in the menu, and another callout box points to the selected file in the browser.

تظهر لك هذه القائمة اختر
Download

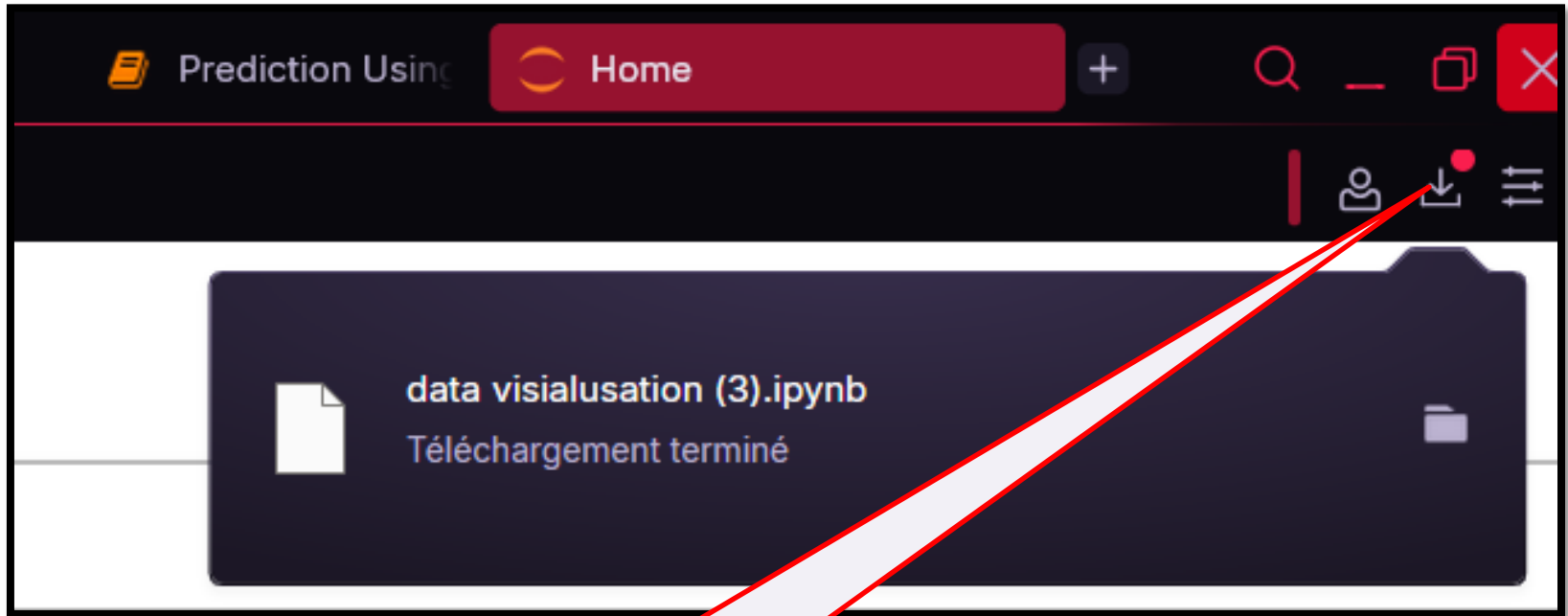
اختر ملف من الملفات ثم اضغط
على يمين الفارة

Anaconda cloud كيفية العمل على

How to work on Anaconda cloud

كيفية تحميل ملف الأكواد

How to download script files



وسترى أن الملف قد حمل على
حاسوبك

Variables and Data Types

المتغيرات وأنواع البيانات

مفهوم المتغير (Variable)

التعريف: المتغير هو اسم يُستخدم لتخزين قيمة في الذاكرة يمكن تغييرها أثناء تنفيذ البرنامج.

مثال:

```
x = 10
y = 5
result = x + y
print(result)
```

15

```
x = 10
y = 5
result = x + y
print(result)
```

تفسير:

المتغير x يحمل القيمة 10، والمتغير y يحمل القيمة 5، ونتيجة جمعها تُخزن في result.

أنواع البيانات الأساسية (Data Types)

النوع	المعنى	مثال	ملاحظة
int	عدد صحيح	<code>x = 12</code>	لا يحتوي على فاصلة عشرية
float	عدد عشري	<code>y = 12.5</code>	يحتوي على فاصلة
str	سلسلة نصية	<code>name = "Rabah"</code>	تُكتب بين علامات اقتباس
bool	قيمة منطقية	<code>is_student = True</code>	True أو False: لها قيمتان فقط

هيكل البيانات (Data Structure)

وتشمل هياكل البيانات في بايثون ما يلي:

- List (القوائم)
- Set (المجموعات)
- Strings (السلاسل النصية)
- Tuples (الصفوف أو المجموعات المرتبة)
- Dictionary (القواميس)
- Deque (الصفوف المزدوجة)
- Heap (الهياكل الهرمية)

أنواع البيانات الأساسية (Data Types)

مثال تطبيقي

```
a = 15 # int
b = 3.14 # float
c = "Python" # str
d = True # bool
print(type(a))
print(type(b))
print(type(c))
print(type(d))
```

```
a = 15          # int
b = 3.14        # float
c = "Python"    # str
d = True        # bool
```

```
print(type(a))
print(type(b))
print(type(c))
print(type(d))
```

```
<class 'int'>
<class 'float'>
<class 'str'>
<class 'bool'>
```

قواعد تسمية المتغيرات (Naming Rules)

- يجب أن يبدأ اسم المتغير بحرف أو شرطة سفلية `_`.
- لا يمكن أن يبدأ برقم.
- لا يجوز احتواء الاسم على فراغات أو رموز خاصة.
- **Python** حساس لحالة الأحرف: **(Case Sensitive)**

Name ≠ name

مثال صحيح:

`age = 20`

`_age = 25`

`userName = "Ali"`

```
age = 20
```

```
_age = 25
```

```
userName = "Ali"
```

قواعد تسمية المتغيرات (Naming Rules)

مثال خاطئ:

2age = 20 خطأ

user name = "Ali" خطأ

```
2age = 20                    # خطأ
user name = "Ali"           # خطأ
```

```
Cell In[6], line 1
```

```
2age = 20                    # خطأ
```

```
^
```

```
SyntaxError: invalid decimal literal
```

أمثلة تطبيقية

برنامج لحساب مجموع رقمين:

a = 5

b = 8

sum = a + b

print(" المجموع هو: ", sum)

```
a = 5
```

```
b = 8
```

```
sum = a + b
```

```
print("المجموع هو:", sum)
```

المجموع هو: 13

إطار البيانات (DataFrame)

إطار البيانات هو بنية بيانات جدوليّة تشبه إلى حدٍ كبير جداول البيانات مثل (Excel).

تم تصميم هذه البنية لتوسيع السلاسل (Series) إلى أبعاد متعددة. في الواقع، يتكوّن إطار البيانات من مجموعة مرتبة من الأعمدة.

DataFrame			
index	columns		
	color	object	price
0	blue	ball	1.2
1	green	pen	1.0
2	yellow	pencil	0.6
3	red	paper	0.9
4	white	mug	1.7

إطار البيانات (DataFrame)

مثال:

```
import pandas as pd
data = {'color' : ['blue','green','yellow','red','white'],
        'object' : ['ball','pen','pencil','paper','mug'],
        'price' : [1.2,1.0,0.6,0.9,1.7]}
frame = pd.DataFrame(data)
frame
```

	color	object	price
0	blue	ball	1.2
1	green	pen	1.0
2	yellow	pencil	0.6
3	red	paper	0.9
4	white	mug	1.7

```
import pandas as pd
data = {'color' : ['blue','green','yellow','red','white'],
        'object' : ['ball','pen','pencil','paper','mug'],
        'price' : [1.2,1.0,0.6,0.9,1.7]}
frame = pd.DataFrame(data)
frame
```

خيار dtype

إن الدالة **array()** لا تقتصر على وسيط (**argument**) واحد فقط. لقد رأينا أن كل كائن من نوع **ndarray** يرتبط بكائن آخر من نوع **dtype**، وهو الذي يحدّد **نوع البيانات** التي سيشغلها كل عنصر داخل المصفوفة.

بشكل افتراضي، تقوم الدالة **array()** باختيار **أنسب نوع بيانات** تلقائياً وفقاً للقيم الموجودة داخل تسلسل القوائم أو الصفوف (**lists** أو **tuples**) ومع ذلك، يمكنك **تحديد نوع البيانات بشكل صريح** من خلال استخدام الخيار **dtype** كوسيط ضمن الدالة.

dtype خيار

مثال توضيحي:

```
import numpy as np  
a = np.array([1, 2, 3],  
dtype=float)  
print(a)
```

```
: import numpy as np  
a = np.array([1, 2, 3],  
dtype=float)  
print(a)
```

```
[1.  2.  3.]
```

الإنشاء الذاتي للمصفوفة (Intrinsic Creation of an Array)

توفر مكتبة **NumPy** مجموعة من **الدوال الجاهزة** التي تُنشئ كائنات من نوع **ndarray** بمحتوى ابتدائي، حيث تختلف القيم الأولية باختلاف الدالة المستخدمة.

خلال هذه المحاضرات، ستكتشف أن هذه الدوال **مفيدة جداً** لأنها تُمكنك من إنشاء كميات كبيرة من البيانات باستخدام سطر واحد فقط من الكود.

مثال:

فعلى سبيل المثال، تقوم الدالة **zeros()** بإنشاء **مصفوفة مملوءة بالأصفار**، وتُحدد أبعادها من خلال المعامل **shape** الذي يُمرّر إليها كوسيط.

```
import numpy as np  
a = np.zeros((3, 3))  
print(a)
```

الإشـاءـة الـذاتـيـة لـلمـصفـوفـة (Intrinsic Creation of an Array)

```
import numpy as np
a = np.zeros((3, 3))
print(a)
```

```
import numpy as np

b= np.ones((3, 3))
print(b)
```

```
[[1. 1. 1.]
 [1. 1. 1.]
 [1. 1. 1.]]
```

```
import numpy as np
```

```
a = np.zeros((3, 3))
print(a)
```

```
[[0. 0. 0.]
 [0. 0. 0.]
 [0. 0. 0.]]
```

Arithmetic Operations & Math

العمليات الرياضية والدوال الرياضية

مكتبة NumPy

الموقع

<https://numpy.org/>

numpy.org



[Install](#) [Documentation](#) [Learn](#) [Community](#) [About Us](#) [News](#) [Contribute](#) [English](#) ▾



The fundamental package for scientific computing with Python

LATEST RELEASE: NUMPY 2.3. VIEW ALL RELEASES

NumPy 2.3.0 released! [2025-06-07](#)

Powerful N-dimensional arrays

Fast and versatile, the NumPy vectorization, indexing, and broadcasting concepts are the de-facto standards of array computing today.

Numerical computing tools

NumPy offers comprehensive mathematical functions, random number generators, linear algebra routines, Fourier transforms, and more.

Open source

Distributed under a liberal [BSD license](#), NumPy is developed and maintained [publicly on GitHub](#) by a vibrant, responsive, and diverse [community](#).

Interoperable

NumPy supports a wide range of hardware and computing platforms, and plays well with distributed, GPU, and sparse array libraries.

Performant

The core of NumPy is well-optimized C code. Enjoy the flexibility of Python with the speed of compiled code.

Easy to use

NumPy's high level syntax makes it accessible and productive for programmers from any background or experience level.

مكتبة NumPy

1. NumPy (Numerical Python)


الوظيفة: إجراء العمليات الرياضية والمعالجة العددية عالية الكفاءة.

المميزات:

- التعامل مع المصفوفات (**Arrays**) والأشعة (**Vectors**).
- تنفيذ العمليات بسرعة أكبر من الحلقات التقليدية.

أوامر أساسية:

```
import numpy as np
a = np.array([1, 2, 3])
print(a * 2)
# الناتج: [6 4 2]
```

Copy code 

```
import numpy as np
a = np.array([1, 2, 3])
print(a * 2) # الناتج: [6 4 2]
```

الاستخدامات: الاقتصاد القياسي، النمذجة الرياضية، المعادلات التفاضلية، المحاكاة.

العمليات الحسابية الأساسية في Python

المحور 01:

المحور 1: العمليات الحسابية الأساسية في Python

العملية	الرمز	المثال	النتيجة	التفسير
الجمع	+	5 + 3	8	جمع رقمين
الطرح	-	45934	6	طرح رقمين
الضرب	*	6 * 3	18	ضرب رقمين
القسمة	/	15 / 4	3.75	قسمة عادية تعطي عددًا عشريًا
باقي القسمة	%	15 % 4	3	باقي القسمة (modulus)
الأس	**	2 ** 3	8	رفع العدد إلى قوة معينة
القسمة الصحيحة	//	15 // 4	3	قسمة بدون كسور عشرية

أمثلة عملية:

a = 10

b = 3

print(a + b)

print(a - b)

print(a * b)

print(a / b)

print(a % b)

print(a ** b)

print(a // b)

```
a = 10
```

```
b = 3
```

```
print(a + b)
```

```
print(a - b)
```

```
print(a * b)
```

```
print(a / b)
```

```
print(a % b)
```

```
print(a ** b)
```

```
print(a // b)
```

```
13
```

```
7
```

```
30
```

```
3.3333333333333335
```

```
1
```

```
1000
```

```
3
```

مثال تطبيقي للطلبة

أحسب نتائج العمليات التالية

$a = 8$

$b = 4$

$c = 2$

`print(a + b + c)`

`print(a - b - c)`

`print((a * b)/c)`

`print(a / (b+c))`

`print(a % b)`

`print(a ** b)`

`print(a // b)`

العمليات الحسابية (Arithmetic Operators)

أول العمليات التي يمكن تنفيذها على المصفوفات هي العمليات الحسابية الأساسية. وأبسط هذه العمليات هي جمع المصفوفة أو ضربها بعدد ثابت (scalar).

مثال:

```
import numpy as np
```

```
a = np.array([1, 2, 3])
```

```
b = a * 2
```

```
c = a + 5
```

```
print("b ضرب المصفوفة في عدد ثابت:", b)
```

```
print("c جمع عدد ثابت إلى المصفوفة:", c)
```

```
import numpy as np
```

```
a = np.array([1, 2, 3])
```

```
b = a * 2
```

```
c = a + 5
```

```
print("b ضرب المصفوفة في عدد ثابت:", b)
```

```
print("c جمع عدد ثابت إلى المصفوفة:", c)
```

```
[6 4 2] ضرب المصفوفة في عدد ثابت:
```

```
[8 7 6] جمع عدد ثابت إلى المصفوفة:
```

العمليات الحسابية على المصفوفات

يمكن أيضاً استخدام هذه **العمليات (operators)** بين **مصفوفتين**.
في مكتبة **NumPy**، تُنفَّذ هذه العمليات بطريقة تُعرف باسم "**العنصر مقابل
العنصر (element-wise)**".

أي أن العملية تُطبَّق فقط بين **العناصر المتقابلة** في كل مصفوفة.
بمعنى آخر، يجب أن تكون المصفوفتان لهما **نفس الأبعاد (shape)**،
ويتم إجراء العملية الحسابية على كل عنصرين يشغلان **نفس الموقع** داخل
المصفوفتين.

ليكون الناتج **مصفوفة جديدة** تحتوي على النتائج في نفس المواقع.

العمليات الحسابية على المصفوفات

مثال توضيحي:

```
import numpy as np
```

```
a = np.array([1, 2, 3])
```

```
b = np.array([4, 5, 6])
```

```
c = a + b
```

```
d = a * b
```

```
print("c مجموع المصفوفتين:", c)
```

```
print("d ناتج ضرب المصفوفتين:", d)
```

```
import numpy as np
```

```
a = np.array([0, 1, 2, 3])
```

```
b = np.array([4, 5, 6, 7])
```

```
c = a + b
```

```
d = a * b
```

```
print("مجموع المصفوفتين:", c)
```

```
print("ناتج ضرب المصفوفتين:", d)
```

```
[10  8  6  4 ] : مجموع المصفوفتين
```

```
[21 12  5  0 ] : ناتج ضرب المصفوفتين
```


العمليات الحسابية على المصفوفات

مثال توضيحي:

```
import numpy as np
```

```
a = np.array([0, 1, 2, 3])
```

```
b = np.array([4, 5, 6, 7])
```

```
c = a + b
```

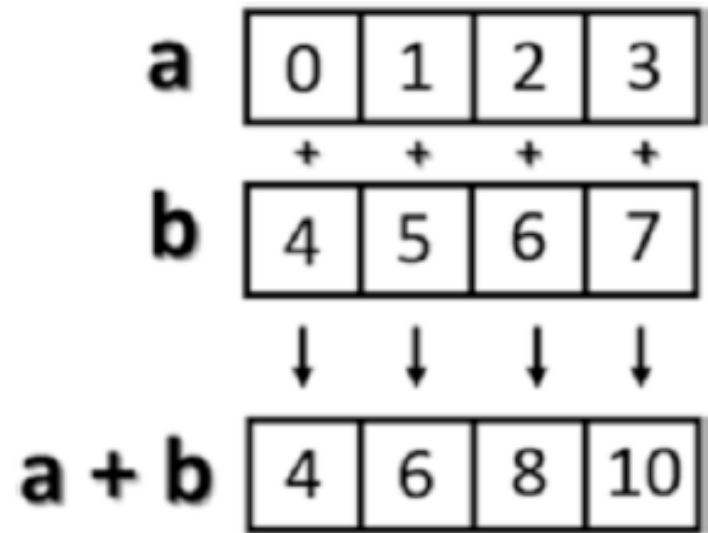
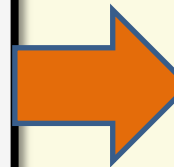
```
d = a * b
```

```
print("مجموع المصفوفتين:", c)
```

```
print("ناتج ضرب المصفوفتين:", d)
```

مجموع المصفوفتين: [4 6 8 10]

ناتج ضرب المصفوفتين: [0 5 12 21]



Element-wise addition

العمليات الحسابية متعددة الأبعاد

بالانتقال إلى الحالة متعددة الأبعاد (**multidimensional case**) تستمر العمليات الحسابية (**arithmetic operators**) في العمل بنفس الطريقة، أي أنها تُطبَّق عنصرًا مقابل عنصر (**element-wise**) حتى في المصفوفات ذات الأبعاد المتعددة.

بمعنى أن العمليات مثل الجمع أو الطرح أو الضرب تُنفَّذ بين العناصر المتقابلة في نفس الموضع داخل كل مصفوفة.

مثال توضيحي:

```
import numpy as np
```

```
A = np.arange(0, 9).reshape(3, 3)
```

```
B = np.ones((3, 3))
```

```
C = A + B
```

```
D = A * B
```

```
print("A:\n", A)
```

```
print("B:\n", B)
```

```
print("A + B =\n", C)
```

```
print("A * B =\n", D)
```

العمليات الحسابية متعددة الأبعاد

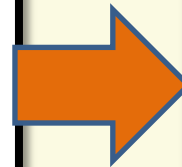
مثال توضيحي:

```
import numpy as np

A = np.arange(0, 9).reshape(3, 3)
B = np.ones((3, 3))

C = A + B
D = A * B

print("المصفوفة A:\n", A)
print("المصفوفة B:\n", B)
print("A + B =\n", C)
print("A * B =\n", D)
```



المصفوفة A:

```
[[0 1 2]
 [3 4 5]
 [6 7 8]]
```

المصفوفة B:

```
[[1. 1. 1.]
 [1. 1. 1.]
 [1. 1. 1.]]
```

A + B =

```
[[1. 2. 3.]
 [4. 5. 6.]
 [7. 8. 9.]]
```

A * B =

```
[[0. 1. 2.]
 [3. 4. 5.]
 [6. 7. 8.]]
```

الضرب المصفوفي (The Matrix Product)

إن تنفيذ العمليات عنصرًا مقابل عنصر (element-wise) هو سمة مميزة لمكتبة **NumPy**.

ففي العديد من أدوات تحليل البيانات الأخرى، يُستخدم العامل * ليعني **الضرب المصفوفي (Matrix Multiplication)** عندما يُطبَّق على مصفوفتين.

لكن في **NumPy**، لا يُمثِّل العامل * الضرب المصفوفي، بل يُستخدم فقط للعمليات العنصرية (element-wise).

أما إذا أردت تنفيذ **الضرب المصفوفي الحقيقي**، فيجب استخدام الدالة **dot()** أو العامل **@**.

الضرب المصفوفي (The Matrix Product)

مثال توضيحي:

```
import numpy as np
```

```
A = np.arange(0, 9).reshape(3, 3)
```

```
B = np.ones((3, 3))
```

```
# ضرب عنصري (element-wise)
```

```
C = A * B
```

```
# ضرب مصفوفي (matrix product)
```

```
D = np.dot(A, B)
```

```
# أو باستخدام العامل @
```

```
E = A @ B
```

```
print("\nالضرب العنصري:", C)
```

```
print("\ndot():\n", D)
```

```
print("\n@:\n", E)
```

الضرب المصفوفي (The Matrix Product)

مثال توضيحي:

```
import numpy as np
```

```
A = np.arange(0, 9).reshape(3, 3)
```

```
B = np.ones((3, 3))
```

```
# ضرب عنصري (element-wise)
```

```
C = A * B
```

```
# ضرب مصفوفي (matrix product)
```

```
D = np.dot(A, B)
```

```
# @ أو باستخدام العامل
```

```
E = A @ B
```

```
print("الضرب العنصري:\n", C)
```

```
print("الضرب المصفوفي باستخدام dot():\n", D)
```

```
print("الضرب المصفوفي باستخدام @:\n", E)
```

الضرب العنصري:

```
[[0. 1. 2.]
```

```
[3. 4. 5.]
```

```
[6. 7. 8.]]
```

dot(): الضرب المصفوفي باستخدام

```
[[ 3.  3.  3.]
```

```
[12. 12. 12.]
```

```
[21. 21. 21.]]
```

@: الضرب المصفوفي باستخدام

```
[[ 3.  3.  3.]
```

```
[12. 12. 12.]
```

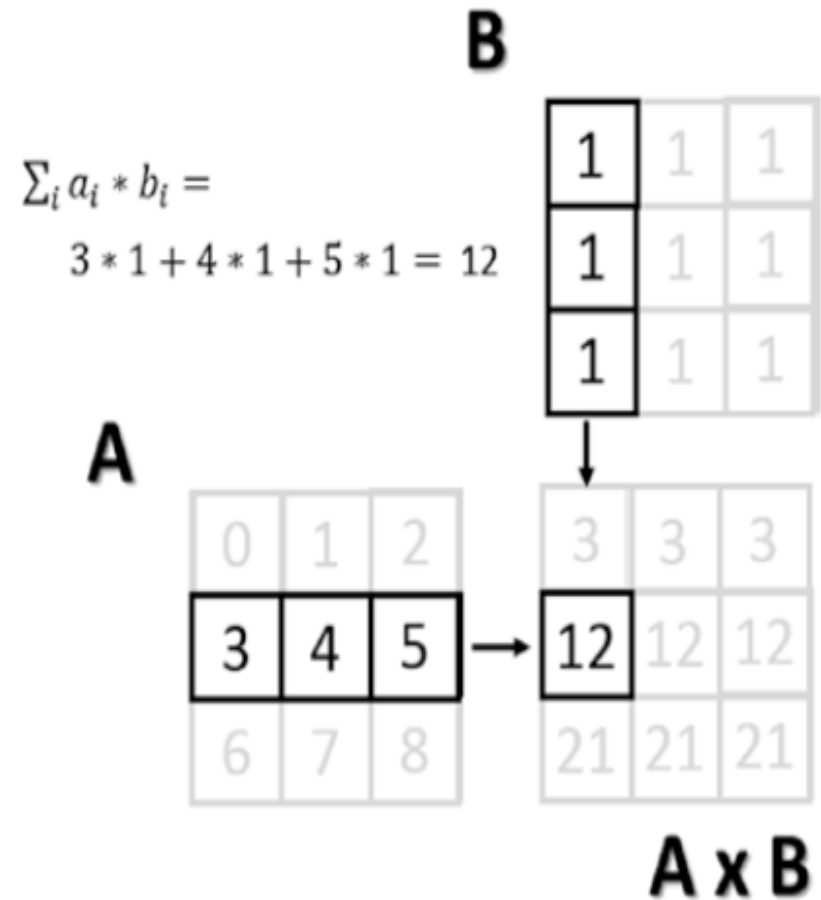
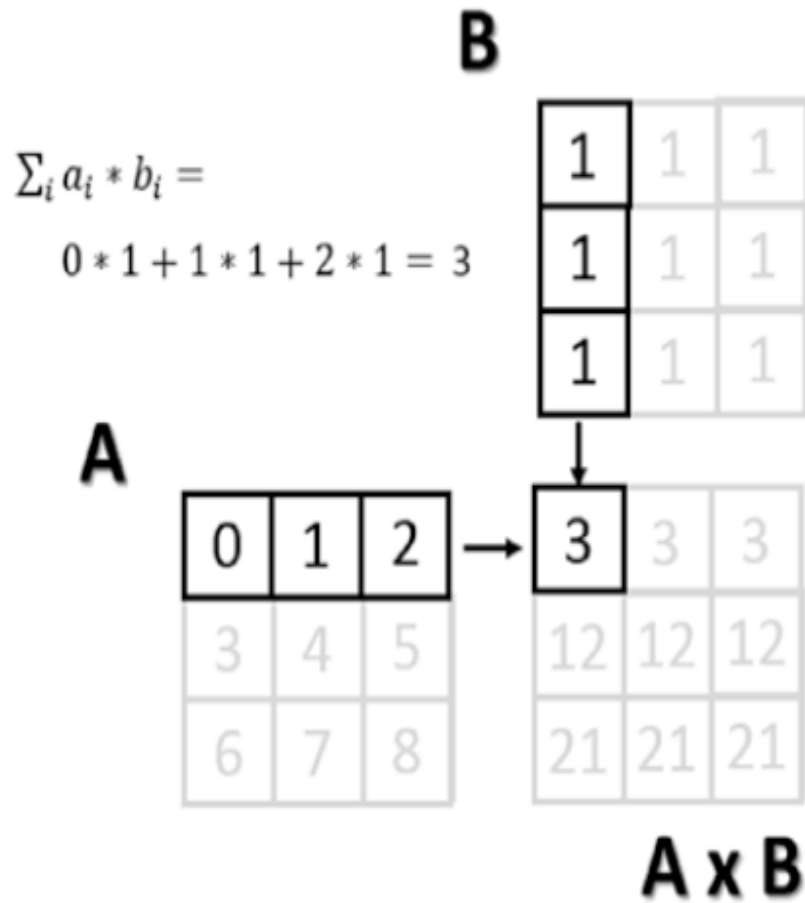
```
[21. 21. 21.]]
```

الضرب المصفوفي (The Matrix Product)

النتيجة في كل موقع من المصفوفة الناتجة هي مجموع حاصل ضرب العناصر المتقابلة بين **صفٍّ من المصفوفة الأولى** و**عمودٍ من المصفوفة الثانية**.

بمعنى آخر، عند حساب العنصر الموجود في الصف i والعمود j من المصفوفة الناتجة: نأخذ الصف رقم i من المصفوفة الأولى، والعمود رقم j من المصفوفة الثانية، ثم نضرب كل عنصر من الصف في العنصر المقابل له من العمود، ونجمع نواتج هذه الضربات معًا للحصول على النتيجة النهائية.

الضرب المصفوفي (The Matrix Product)



Calculating matrix elements as a result of a matrix product

الضرب المصفوفي (The Matrix Product)

طريقة بديلة لكتابة **الضرب المصفوفي (Matrix Product)** هي استخدام الدالة **dot()** كدالة تابعة لأحد الكائنين (المصفوفتين).

أي بدلاً من كتابة:
np.dot(A, B)
يمكنك ببساطة كتابة:
A.dot(B)

```
import numpy as np

A = np.array([[1, 2],
               [3, 4]])

B = np.array([[5, 6],
               [7, 8]])

C = A.dot(B)
print(C)
```

```
[[19 22]
 [43 50]]
```

الضرب المصفوفي (The Matrix Product)

الضرب المصفوفي (Matrix Product) ليس عملية تبادلية

لاحظ أن **الضرب المصفوفي (Matrix Product)** ليس عملية
تبادلية (Non-Commutative Operation)،
أي أن ترتيب العوامل (المصفوفتين) في العملية مهم جداً.
بمعنى آخر:

$$A \cdot B \neq B \cdot A$$

فنتيجة ضرب المصفوفة **A** في **B** ليست بالضرورة مساوية لنتيجة
ضرب **B** في **A**،
بل وقد يكون من غير الممكن أصلاً تنفيذ العملية إذا لم تتوافق الأبعاد
بين المصفوفتين.

الضرب المصفوفي (The Matrix Product)

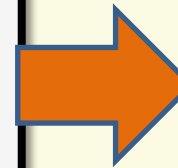
مثال توضيحي:

```
import numpy as np

A = np.array([[1, 2],
              [3, 4]])

B = np.array([[5, 6],
              [7, 8]])

print("A.dot(B):\n", A.dot(B))
print("B.dot(A):\n", B.dot(A))
```



```
A.dot(B):
[[19 22]
 [43 50]]
B.dot(A):
[[23 34]
 [31 46]]
```

عوامل الزيادة والنقصان

(Increment and Decrement Operators)

في الواقع، لا توجد عوامل ++ أو -- في لغة بايثون (Python) كما هو الحال في لغات أخرى مثل C أو Java. لزيادة أو إنقاص القيم، يجب استخدام العوامل المركبة

مثل:

$+=$ لزيادة القيمة،

$-=$ لإنقاص القيمة.

هذه العوامل لا تختلف كثيرًا عن العوامل الحسابية العادية، إلا أنها لا تُنشئ مصفوفة جديدة، بل تُعيد تعيين النتيجة إلى نفس المصفوفة الأصلية (أي يتم التعديل في المكان نفسه *in-place*).

عوامل الزيادة والنقصان (Increment and Decrement Operators)

مثال توضيحي:

```
import numpy as np
```

```
a = np.array([1, 2, 3])
```

زيادة جميع القيم بمقدار 1

```
a += 1
```

```
print("a بعد الزيادة:", a)
```

إنقاص جميع القيم بمقدار 2

```
a -= 2
```

```
print("a بعد النقصان:", a)
```

```
import numpy as np
```

```
a = np.array([1, 2, 3])
```

زيادة جميع القيم بمقدار 1

```
a += 1
```

```
print("a بعد الزيادة:", a)
```

إنقاص جميع القيم بمقدار 2

```
a -= 2
```

```
print("a بعد النقصان:", a)
```

بعد الزيادة: [4 3 2]

بعد النقصان: [2 1 0]

العوامل المركبة (Complex Operators)

لذلك، فإن استخدام هذه **العمليات المركبة** أوسع بكثير من مجرد الزيادة أو النقصان بوحدة واحدة كما في العوامل التقليدية (`++` أو `--`).

فهي تُستخدم في **العديد من الحالات** التي ترغب فيها بتعديل قيم المصفوفة **دون إنشاء مصفوفة جديدة**.

بمعنى آخر، هذه العوامل تسمح لك **بتحديث القيم داخل المصفوفة الأصلية مباشرة** (*in-place modification*) وهو ما يُعدّ **أكثر كفاءة في الذاكرة** وأسرع في التنفيذ.

العوامل المركبة (Complex Operators)

مثال توضيحي:

```
import numpy as np
a = np.array([10, 20, 30])
# مضاعفة القيم دون إنشاء مصفوفة جديدة
a *= 2
Print( "بعد الضرب:", a)
```

```
import numpy as np

a = np.array([10, 20, 30])

# مضاعفة القيم دون إنشاء مصفوفة جديدة
a *= 2
print("بعد الضرب:", a)
```

بعد الضرب: [60 40 20]

المحور 02:
مكتبة الدوال الرياضية
(math library)

المحور 2: مكتبة الدوال الرياضية (math library)

مكتبة **math** تحتوي على دوال وأدوات لإجراء عمليات رياضية متقدمة.

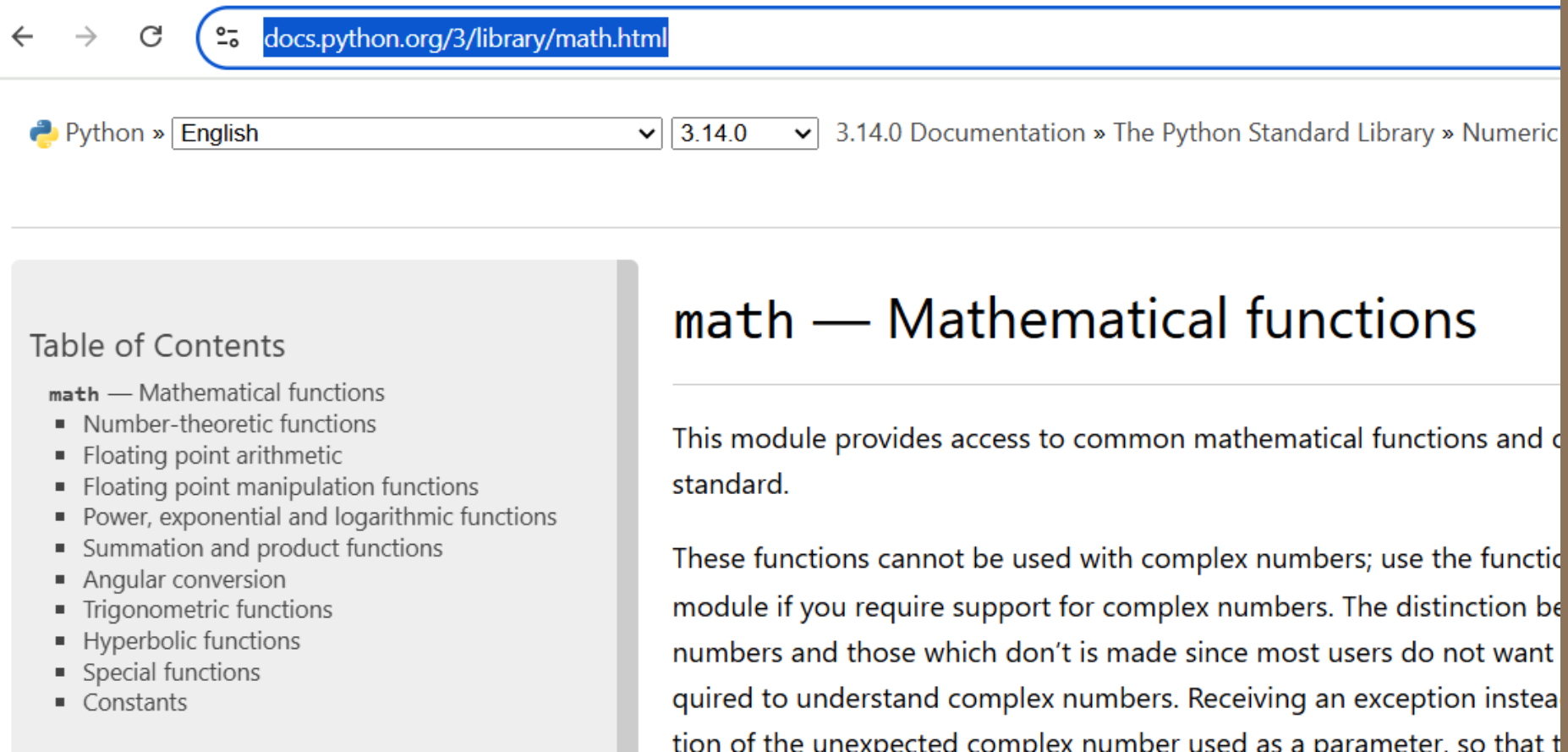
أمر استيراد مكتبة **math**:

```
import math
```

المحور 2: مكتبة الدوال الرياضية (math library)

موقع مكتبة **math** على الرابط التالي:

<https://docs.python.org/3/library/math.html>



The screenshot shows a web browser displaying the Python documentation for the `math` module. The browser's address bar shows the URL `docs.python.org/3/library/math.html`. The page header includes navigation links for Python, English, 3.14.0, and 3.14.0 Documentation. The main content area is titled "math — Mathematical functions" and includes a "Table of Contents" sidebar. The sidebar lists various mathematical functions available in the `math` module, such as "Number-theoretic functions", "Floating point arithmetic", and "Trigonometric functions". The main text area provides an overview of the module, stating that it provides access to common mathematical functions and constants, and that these functions cannot be used with complex numbers.

Python » English » 3.14.0 » 3.14.0 Documentation » The Python Standard Library » Numeric

Table of Contents

- math** — Mathematical functions
 - Number-theoretic functions
 - Floating point arithmetic
 - Floating point manipulation functions
 - Power, exponential and logarithmic functions
 - Summation and product functions
 - Angular conversion
 - Trigonometric functions
 - Hyperbolic functions
 - Special functions
 - Constants

math — Mathematical functions

This module provides access to common mathematical functions and constants from the C standard.

These functions cannot be used with complex numbers; use the `cmath` module if you require support for complex numbers. The distinction between functions that accept complex numbers and those which don't is made since most users do not want to be required to understand complex numbers. Receiving an exception instead of the unexpected complex number used as a parameter, so that t

المحور 2: مكتبة الدوال الرياضية (math library) أهم الدوال الرياضية:

النتيجة	المثال	الاستخدام	الدالة
4.0	<code>math.sqrt(16)</code>	الجذر التربيعي	<code>math.sqrt(x)</code>
8.0	<code>math.pow(2, 3)</code>	القوة (الأس)	<code>math.pow(x, y)</code>
4	<code>math.ceil(3.2)</code>	التقريب للأعلى	<code>math.ceil(x)</code>
3	<code>math.floor(3.8)</code>	التقريب للأسفل	<code>math.floor(x)</code>
3.14159	<code>math.pi</code>	قيمة π	<code>math.pi</code>
2.302	<code>math.log(10)</code>	اللوغاريتم الطبيعي	<code>math.log(x)</code>
1.0	<code>math.sin(math.pi/2)</code>	الجيب	<code>math.sin(x)</code>

المحور 2: مكتبة الدوال الرياضية (math library)

مثال تطبيقي:

```
import math
```

```
x = 16
```

```
print(" الجذر التربيعي ", math.sqrt(x))
```

```
print(" القيمة المقربة للأعلى: ", math.ceil(4.2))
```

```
print(" القيمة المقربة للأسفل: ", math.floor(4.9))
```

```
print(" قيمة  $\pi$  ", math.pi)
```

المحور 2: مكتبة الدوال الرياضية (math library)

مثال تطبيقي:

```
import math

x = 16
print("الجذر التربيعي:", math.sqrt(x))
print("القيمة المقربة للأعلى:", math.ceil(4.2))
print("القيمة المقربة للأسفل:", math.floor(4.9))
print("قيمة  $\pi$ :", math.pi)
```

الجذر التربيعي: 4.0

القيمة المقربة للأعلى: 5

القيمة المقربة للأسفل: 4

قيمة π : 3.141592653589793

المحور 3: تطبيقات اقتصادية بسيطة

حساب معدل النمو الاقتصادي

معدل النمو الاقتصادي هو نسبة زيادة الناتج الداخلي الخام gdp بين سنتين متتاليتين

$$\text{growth} = ((\text{gdp}_t - \text{gdp}_{t-1}) / \text{gdp}_{t-1}) * 100$$

كود بايثون: Python Code

```
gdp_2024 = 1800
```

```
gdp_2023 = 1650
```

```
growth = ((gdp_2024 - gdp_2023) / gdp_2023) * 100
```

```
print( " معدل النمو الاقتصادي هو: ", round(growth, 2), "%")
```

المحور 3: تطبيقات اقتصادية بسيطة

حساب معدل النمو الاقتصادي

معدل النمو الاقتصادي هو نسبة زيادة الناتج الداخلي الخام gdp بين سنتين متتاليتين

$$\text{growth} = ((\text{gdp}_t - \text{gdp}_{t-1}) / \text{gdp}_{t-1}) * 100$$

```
gdp_2024 = 1800
```

```
gdp_2023 = 1650
```

```
growth = ((gdp_2024 - gdp_2023) / gdp_2023) * 100
```

```
print("معدل النمو الاقتصادي هو:", round(growth, 2), "%")
```

% معدل النمو الاقتصادي هو: 9.09

المحور 3: تطبيقات اقتصادية بسيطة

مسألة: المطلوب كتابة كود بايثون

لحساب الفائدة المركبة باستخدام Python

المعادلة الرياضية:

$$n \left(\frac{r}{100} + 1 \right) \times P = A$$

حيث:

A = المبلغ النهائي بعد الفائدة

P = المبلغ الأصلي (رأس المال)

r = نسبة الفائدة السنوية (%)

n = عدد السنوات

المحور 3: تطبيقات اقتصادية بسيطة

حل مسألة:

كود بايثون لحساب الفائدة المركبة باستخدام Python

P = 10000

r = 5

n = 3

A = P * (1 + r/100) ** n

interest = A - P

print("دينار", round(A, 2), "سنوات:", "n, "المبلغ النهائي بعد", " ,"

print("دينار", round(interest, 2), "الفائدة المكتسبة:", " ,"

حساب المتوسط الحسابي للأسعار

price1 = 100

price2 = 120

price3 = 80

avg_price = (price1 + price2 + price3) / 3

print(" المتوسط الحسابي للأسعار ", avg_price)

الدوال العالمية (Universal Functions - ufunc)

الدالة العالمية، والتي تُعرف اختصارًا باسم **ufunc**،
هي دالة تُطبَّق على عناصر المصفوفة عنصرًا تلو الآخر
(**element-by-element**)

بمعنى آخر، تعمل **ufunc** على كل عنصر من عناصر المصفوفة بشكلٍ مستقل،

وتُنتج **مصفوفة جديدة** تحتوي على النتائج المقابلة لكل عنصر في نفس الموضع.
وفي النهاية، تكون **المصفوفة الناتجة لها نفس حجم** (**shape**) المصفوفة الأصلية.

الدوال العالمية (Universal Functions - ufunc)

هناك العديد من العمليات الرياضية والدوال المثلثية التي تندرج ضمن تعريف الدوال العالمية (ufuncs) في مكتبة NumPy. فهي جميعها تُطبَّق عنصرًا تلو الآخر (element-wise) على المصفوفات.

على سبيل المثال:

لكل عنصر. (square root) لحساب الجذر التربيعي → `np.sqrt()`

لكل عنصر. (atural logarithm) لحساب اللوغاريتم الطبيعي → `np.log()`

لكل عنصر (بالراديان). (sine) لحساب جيب الزاوية → `np.sin()`

الدوال العالمية (Universal Functions - ufunc)

مثال توضيحي:

```
import numpy as np
a = np.array([1, 4, 9, 16])

sqrt_result = np.sqrt(a)
log_result = np.log(a)
sin_result = np.sin(a)

print("sqrt_result الجذر التربيعي:", " ")
print("log_result اللوغاريتم الطبيعي:", " ")
print("sin_result جيب الزاوية:", " ")
```

الدوال العالمية (Universal Functions - ufunc)

مثال توضيحي:

```
import numpy as np

a = np.array([1, 4, 9, 16])

sqrt_result = np.sqrt(a)
log_result = np.log(a)
sin_result = np.sin(a)

print("الجذر التربيعي:", sqrt_result)
print("اللوغاريتم الطبيعي:", log_result)
print("جيب الزاوية:", sin_result)
```

الجذر التربيعي: [0.4 0.3 0.2 0.1]

اللوغاريتم الطبيعي: [2.77258872 2.19722458 1.38629436 0.0]

جيب الزاوية: [0.28790332- 0.41211849 0.7568025- 0.84147098]

الدوال التجميعية (Aggregate Functions)

الدوال التجميعية هي دوال تُنفَّذ عملية حسابية على مجموعة من القيم مثل المصفوفة ثم تُنتج نتيجة واحدة فقط تمثل تجميع تلك القيم.

فعلى سبيل المثال، مجموع جميع عناصر المصفوفة يُعدّ دالة تجميعية. وتوجد العديد من الدوال من هذا النوع مضمّنة داخل الفئة **ndarray** في مكتبة **NumPy**،

مما يعني أنه يمكن استدعاؤها مباشرة من الكائن (المصفوفة) الذي نريد إجراء العملية عليه.

أمثلة على الدوال التجميعية:

sum() → حساب مجموع العناصر.

mean() → حساب المتوسط الحسابي.

max() → إيجاد أكبر قيمة.

min() → إيجاد أصغر قيمة.

prod() → حساب ناتج ضرب جميع العناصر.

الدوال التجميعية (Aggregate Functions)

مثال توضيحي:

```
import numpy as np
a = np.array([1, 2, 3, 4, 5])

print("a.sum()", "مجموع العناصر:")
print("a.mean()", "المتوسط الحسابي:")
print("a.max()", "أكبر قيمة:")
print("a.min()", "أصغر قيمة:")
print("a.prod()", "ناتج الضرب:")
```

```
import numpy as np

a = np.array([1, 2, 3, 4, 5])

print("a.sum()", "مجموع العناصر:")
print("a.mean()", "المتوسط الحسابي:")
print("a.max()", "أكبر قيمة:")
print("a.min()", "أصغر قيمة:")
print("a.prod()", "ناتج الضرب:")
```

مجموع العناصر: 15
المتوسط الحسابي: 3.0
أكبر قيمة: 5
أصغر قيمة: 1
ناتج الضرب: 120

الفهرسة، والتقسيم، والتكرار (Indexing, Slicing, and Iterating)

في الأقسام السابقة، رأينا كيف يمكن إنشاء المصفوفات (**arrays**) وتنفيذ عمليات حسابية عليها.

أما في هذا القسم، فستتعلم كيفية التعامل مع هذه المصفوفات مباشرة عن طريق تحديد عناصرها والوصول إليها أو تعديلها.

أولاً: الفهرسة: (**Indexing**)

تُستخدم **الفهرسة** في المصفوفات للوصول إلى العناصر الفردية داخلها. يتم ذلك دائماً باستخدام الأقواس المربعة [] لتحديد موقع العنصر داخل المصفوفة. تُتيح لك الفهرسة:

1. استخراج قيمة معينة من المصفوفة،
2. اختيار عناصر محددة،
3. أو حتى تغيير قيمة عنصر موجود.

الفهرسة، والتقسيم، والتكرار (Indexing, Slicing, and Iterating)

مثال توضيحي:

```
import numpy as np
a = np.array([10, 20, 30, 40, 50])
```

استخراج عناصر فردية #

```
print(a[0]) # 10 → العنصر الأول
print(a[3]) # 40 → العنصر الرابع
```

تعديل قيمة عنصر #

```
a[2] = 99
print("a بعد التعديل:", a)
```

```
import numpy as np

a = np.array([10, 20, 30, 40, 50])
```

استخراج عناصر فردية #

```
print(a[0]) # 10 → العنصر الأول
print(a[3]) # 40 → العنصر الرابع
```

تعديل قيمة عنصر #

```
a[2] = 99
print("a بعد التعديل:", a)
```

10

40

بعد التعديل: [50 40 99 20 10]

الفهرسة، والتقسيم، والتكرار (Indexing, Slicing, and Iterating)

ثانياً: التقسيم (Slicing)

يُستخدم التقسيم (**Slicing**) لاستخراج أجزاء محددة من المصفوفة وإنشاء مصفوفات جديدة منها.

في القوائم العادية في بايثون، يؤدي التقسيم إلى إنشاء نسخة مستقلة من البيانات،

أما في **NumPy**، فإن نتيجة التقسيم هي عرض (**view**) على نفس البيانات الأصلية —

أي أن أي تعديل على الجزء المُستخرج سينعكس مباشرة على المصفوفة الأصلية.

يتم تنفيذ التقسيم باستخدام صيغة الشرائح (**slice syntax**)،

أي باستخدام نقطتين رأسيين (:) بين قيم البداية والنهاية داخل الأقواس المربعة [].

الفهرسة، والتقسيم، والتكرار (Indexing, Slicing, and Iterating)

ثانياً: التقسيم (Slicing)

مثال توضيحي:

```
import numpy as np
```

```
a = np.array([10, 20, 30, 40, 50, 60, 70])
```

استخراج العناصر من الثاني إلى السادس

```
b = a[1:6]
```

```
print("a: المصفوفة الأصلية:", a)
```

```
print("b: الجزء المستخرج:", b)
```

النتيجة:

المصفوفة الأصلية [10 20 30 40 50 60 70]:

الجزء المستخرج [20 30 40 50 60]:

```
import numpy as np
```

```
a = np.array([10, 20, 30, 40, 50, 60, 70])
```

استخراج العناصر من الثاني إلى السادس

```
b = a[1:6]
```

```
print("a: المصفوفة الأصلية:", a)
```

```
print("b: الجزء المستخرج:", b)
```

المصفوفة الأصلية: [70 60 50 40 30 20 10]

الجزء المستخرج: [60 50 40 30 20]

الفهرسة، والتقسيم، والتكرار (Indexing, Slicing, and Iterating)

ثالثًا: تكرار عناصر المصفوفة (Iterating an Array)

في لغة بايثون (Python)، يُعدّ التكرار على عناصر المصفوفة أمرًا بسيطًا جدًا

فكل ما تحتاج إليه هو استخدام **حلقة for** للمرور على العناصر واحدًا تلو الآخر.

مثال توضيحي:

```
import numpy as np
a = np.array([10, 20, 30, 40, 50])
for element in a:
    print(element)
```

```
import numpy as np

a = np.array([10, 20, 30, 40, 50])

for element in a:
    print(element)
```

```
10
20
30
40
50
```

الشروط والمصفوفات المنطقية (Conditions and Boolean Arrays)

حتى الآن، استخدمنا **الفهرسة (indexing)** و**التقسيم (slicing)** لاختيار جزء معين من المصفوفة باستخدام **فهرسة رقمية**. لكن هناك طريقة أخرى أكثر مرونة وهي استخدام **الشروط (conditions)** و**العوامل المنطقية (Boolean operators)** لاستخراج عناصر محددة بناءً على قيمها.

الفكرة الأساسية:

بدلاً من تحديد المواقع بالأرقام، يمكنك تحديد **شرط منطقي** مثل:

(**<**, **>**, **==**, **!=**)

وسيعيد **NumPy** مصفوفة منطقية تحتوي على قيم (**True** أو **False**) توضح أي العناصر تحقق الشرط.

ثم يمكنك استخدام هذه المصفوفة المنطقية لاستخراج القيم المطلوبة من المصفوفة الأصلية.

الشروط والمصفوفات المنطقية

(Conditions and Boolean Arrays)

مثال توضيحي:

```
import numpy as np
# إنشاء مصفوفة 4x4 تحتوي على أعداد عشوائية بين 0 و 1
a = np.random.rand(4, 4)
print("\nmصفوفة الأصلية:", a)

# إنشاء شرط منطقي: القيم الأقل من 0.5
mask = a < 0.5
print("\nmصفوفة المنطقية:", mask)

# استخراج القيم التي تحقق الشرط
selected = a[a < 0.5]
print("\nmالقيم الأقل من 0.5:", selected)
```

الشروط والمصفوفات المنطقية

(Conditions and Boolean Arrays)

مثال توضيحي:

```
import numpy as np

# إنشاء مصفوفة 4x4 تحتوي على أعداد عشوائية بين 0 و 1
a = np.random.rand(4, 4)

print("\nالمصفوفة الأصلية:\n", a)
|

# إنشاء شرط منطقي: القيم الأقل من 0.5
mask = a < 0.5

print("\nالمصفوفة المنطقية:\n", mask)

# استخراج القيم التي تحقق الشرط
selected = a[a < 0.5]

print("\nالقيم الأقل من 0.5:\n", selected)
```


الشروط والمصفوفات المنطقية

(Conditions and Boolean Arrays)

مثال توضيحي:

النتائج

المصفوفة الأصلية:

```
[[0.95632821 0.70461212 0.50478665 0.86606083]
 [0.02274177 0.86761746 0.64132829 0.14369029]
 [0.19384728 0.88090747 0.84995347 0.71815552]
 [0.37010807 0.36550859 0.39814462 0.09448449]]
```

المصفوفة المنطقية:

```
[[False False False False]
 [ True False False  True]
 [ True False False False]
 [ True  True  True  True]]
```

القيم الأقل من 0.5:

```
[0.02274177 0.14369029 0.19384728 0.37010807 0.36550859 0.39814462
 0.09448449]
```

تغيير شكل المصفوفة (Shape Manipulation)

كما رأينا سابقًا عند إنشاء مصفوفة ثنائية الأبعاد،

يمكنك تحويل مصفوفة أحادية البعد (**1D array**) إلى مصفوفة ثنائية أو متعددة الأبعاد (**matrix**)

بسهولة باستخدام الدالة **reshape()** من مكتبة **NumPy**.
هذه العملية لا تُغيّر محتوى البيانات، بل فقط طريقة تنظيمها (شكلها) داخل المصفوفة.

تغيير شكل المصفوفة (Shape Manipulation)

مثال توضيحي:

```
import numpy as np
```

```
# إنشاء مصفوفة أحادية البعد من 0 إلى 8
```

```
a = np.arange(9)
```

```
print("a المصفوفة الأصلية:", "
```

```
# تحويلها إلى مصفوفة ثنائية الأبعاد 3×3
```

```
b = a.reshape(3, 3)
```

```
print("\nb بعد إعادة التشكيل (3×3):\n", b)
```

تغيير شكل المصفوفة (Shape Manipulation)

مثال توضيحي:

```
import numpy as np
```

```
# إنشاء مصفوفة أحادية البعد من 0 إلى 8  
a = np.arange(9)  
print("المصفوفة الأصلية:", a)
```

```
# تحويلها إلى مصفوفة ثنائية الأبعاد 3x3  
b = a.reshape(3, 3)  
print("\n(3x3) بعد إعادة التشكيل\n", b)
```

المصفوفة الأصلية: [8 7 6 5 4 3 2 1 0]

بعد إعادة التشكيل (3x3):

```
[[0 1 2]  
 [3 4 5]  
 [6 7 8]]
```

التعامل مع المصفوفات (Array Manipulation)

في كثير من الأحيان، قد تحتاج إلى إنشاء مصفوفة جديدة بالاعتماد على مصفوفات موجودة مسبقًا.

في هذا القسم، سنتعلم كيفية دمج (joining) المصفوفات معًا أو تقسيم (splitting) مصفوفة إلى عدة أجزاء فرعية.

أولاً: دمج المصفوفات (Joining Arrays)

يمكنك دمج عدة مصفوفات لتكوين مصفوفة واحدة باستخدام دوال مثل:

`np.concatenate()`

للمرج أفقيًا (على المحور الأفقي). `np.hstack()`

للمرج عموديًا (على المحور العمودي). `np.vstack()`

التعامل مع المصفوفات (Array Manipulation)

أولاً: دمج المصفوفات (Joining Arrays)

مثال تطبيقي:

```
import numpy as np
```

```
a = np.array([1, 2, 3])
```

```
b = np.array([4, 5, 6])
```

```
# concatenate الدمج البسيط )
```

```
c = np.concatenate((a, b))
```

```
print("Concatenate:", c)
```

```
# الدمج الأفقي والعمودي
```

```
A = np.array([[1, 2], [3, 4]])
```

```
B = np.array([[5, 6], [7, 8]])
```

```
print("\n:دمج أفقي", np.hstack((A, B)))
```

```
print("\n:دمج عمودي", np.vstack((A, B)))
```

التعامل مع المصفوفات (Array Manipulation)

أولاً: دمج المصفوفات (Joining Arrays)

مثال تطبيقي:

```
import numpy as np
```

```
a = np.array([1, 2, 3])
```

```
b = np.array([4, 5, 6])
```

```
# الدمج البسيط (concatenate)
```

```
c = np.concatenate((a, b))
```

```
print("Concatenate:", c)
```

```
# الدمج الأفقي والعمودي
```

```
A = np.array([[1, 2],  
              [3, 4]])
```

```
B = np.array([[5, 6],  
              [7, 8]])
```

```
print("\nدمج أفقي:\n", np.hstack((A, B)))
```

```
print("\nدمج عمودي:\n", np.vstack((A, B)))
```

Concatenate: [1 2 3 4 5 6]

دمج أفقي:

[[1 2 5 6]

[3 4 7 8]]

دمج عمودي:

[[1 2]

[3 4]

[5 6]

[7 8]]

التعامل مع المصفوفات (Array Manipulation)

ثانياً: تقسيم المصفوفات (Splitting Arrays)

يمكنك أيضاً تقسيم مصفوفة إلى عدة أجزاء باستخدام:

`np.split()`

`np.hsplit()` (تقسيم أفقي).

`np.vsplit()` (تقسيم عمودي).

مثال:

```
x = np.arange(1, 10)
y = np.split(x, 3)
print("y إلى 3 أجزاء:", "D تقسيم مصفوفة 1")

M = np.arange(1, 10).reshape(3, 3)
print("\n\tقسيم أفقي:", np.hsplit(M, 3))
print("\n\tقسيم عمودي:", np.vsplit(M, 3))
```


المصفوفات المُهيكلَة (Structured Arrays)

حتى الآن، تعاملنا في الأمثلة السابقة مع **مصفوفات أحادية البعد (1D)** وثنائية البعد **(2D)** تحتوي على قيم رقمية بسيطة.

لكن مكتبة **NumPy** تتيح إنشاء مصفوفات أكثر تعقيدًا من حيث البنية وليس فقط من حيث الحجم،

ويُطلق على هذا النوع من المصفوفات اسم **المصفوفات المُهيكلَة (Structured Arrays)**

المصفوفات المُهيكلَة (Structured Arrays)

ما هي المصفوفات المُهيكلَة؟

هي مصفوفات يمكن أن تحتوي على **سجلات** (**records**) أو **هياكل بيانات** (**structs**) بدلاً من عناصر رقمية بسيطة. كل سجل يمكن أن يضم **عدة حقول** (**fields**) ذات أنواع بيانات مختلفة، مثل الأعداد الصحيحة، والعشرية، والسلاسل النصية، ... إلخ.

كيفية إنشائها

يمكنك إنشاء مصفوفة مُهيكلَة باستخدام المعامل **dtype** لتحديد الحقول داخل كل سجل.

يُكتب ذلك على شكل **قائمة من المواصفات** (**specifiers**) كل منها يحدّد اسم الحقل، ونوع البيانات، والترتيب داخل السجل.

المصفوفات المُهيكلَة (Structured Arrays)

مثال توضيحي:

```
import numpy as np
```

```
# تعريف نوع البيانات (هيكل السجل)
```

```
student_dtype = [('name', 'U10'), ('age', 'i4'),  
('grade', 'f4')]
```

```
# إنشاء مصفوفة مهيكلَة تحتوي على سجلات الطلاب
```

```
students = np.array([  
    ('Ali', 20, 15.5),  
    ('Sara', 22, 17.2),  
    ('Omar', 19, 12.8)  
, dtype=student_dtype)
```

```
print(students)
```

المصفوفات المُهيكلَة (Structured Arrays)

مثال توضيحي:

```
import numpy as np

# تعريف نوع البيانات (هيكل السجل)
student_dtype = [('name', 'U10'), ('age', 'i4'), ('grade', 'f4')]

# إنشاء مصفوفة مهيكلة تحتوي على سجلات الطلاب
students = np.array([
    ('Ali', 20, 15.5),
    ('Sara', 22, 17.2),
    ('Omar', 19, 12.8)
], dtype=student_dtype)

print(students)

[('Ali', 20, 15.5) ('Sara', 22, 17.2) ('Omar', 19, 12.8)]
```

قراءة وكتابة بيانات المصفوفات في الملفات (Reading and Writing Array Data on Files)

يُعدّ التعامل مع الملفات من الجوانب الأساسية في مكتبة **NumPy**، وخاصةً عندما تكون لديك كميات كبيرة من البيانات المخزنة على القرص وتريد معالجتها أو تحليلها.

في معظم تطبيقات تحليل البيانات (**Data Analysis**)، تكون أحجام البيانات كبيرة جدًا،

لذلك لا يمكن أو لا يُنصح بإدارتها يدويًا،

بل يجب قراءة البيانات من الملفات مباشرة وتحويلها إلى مصفوفات **NumPy**،

أو حفظ نتائج العمليات الحسابية في ملفات لتُستخدم لاحقًا.

قراءة وكتابة بيانات المصفوفات في الملفات (Reading and Writing Array Data on Files)

أولاً: تحميل وحفظ البيانات في الملفات الثنائية (Loading and Saving Data in Binary Files)

تُوفّر مكتبة **NumPy** زوجًا من الدوال القوية — **save()** و **load()** — تُستخدم لحفظ البيانات واسترجاعها بتنسيق **ثنائي (Binary Format)** بكفاءة عالية.

هذا النوع من الملفات يُخزّن البيانات **بشكل مضغوط وسريع القراءة** مقارنة بالملفات النصية.

ويُستخدم كثيرًا في مشاريع **التحليل الإحصائي والتعلّم الآلي** حيث تكون المصفوفات كبيرة جدًا.

قراءة وكتابة بيانات المصفوفات في الملفات

(Reading and Writing Array Data on Files)

أولاً: تحميل وحفظ البيانات في الملفات الثنائية

(Loading and Saving Data in Binary Files)

1. حفظ البيانات باستخدام `save()`

بمجرد أن تمتلك مصفوفة تريد حفظها — مثل نتائج تحليل بيانات — يمكنك استدعاء الدالة `np.save()` وتحديد اسم الملف والمصفوفة كوسيطين. يُضاف تلقائيًا الامتداد `np` إلى اسم الملف.

قراءة وكتابة بيانات المصفوفات في الملفات

(Reading and Writing Array Data on Files)

أولاً: تحميل وحفظ البيانات في الملفات الثنائية

(Loading and Saving Data in Binary Files)

1. حفظ البيانات باستخدام `save()`

مثال:

```
import numpy as np
```

```
# إنشاء مصفوفة كمثال
```

```
a = np.array([[10, 20, 30],  
              [40, 50, 60]])
```

```
# حفظ المصفوفة في ملف ثنائي
```

```
np.save("results", a) # سيُنشأ ملف باسم:  
results.npy
```


قراءة وكتابة بيانات المصفوفات في الملفات

(Reading and Writing Array Data on Files)

أولاً: تحميل وحفظ البيانات في الملفات الثنائية

(Loading and Saving Data in Binary Files)

2. تحميل البيانات باستخدام `load()`

عند الحاجة إلى استرجاع المصفوفة لاحقاً، يمكنك استخدام الدالة `np.load()` لقراءة الملف وإعادة المصفوفة إلى الذاكرة.

مثال:

```
# تحميل المصفوفة من الملف
```

```
loaded_array = np.load("results.npy")
```

```
print("\nالمصفوفة المحملة:", loaded_array)
```

قراءة وكتابة بيانات المصفوفات في الملفات

(Reading and Writing Array Data on Files)

أولاً: تحميل وحفظ البيانات في الملفات الثنائية

(Loading and Saving Data in Binary Files)

2. تحميل البيانات باستخدام load()

مثال:

```
# تحميل المصفوفة من الملف
loaded_array = np.load("results.npy")

print("المصفوفة المحملة:\n", loaded_array)
```

المصفوفة المحملة:

```
[[10 20 30]
 [40 50 60]]
```

قراءة الملفات ذات البيانات الجدولية (Reading Files with Tabular Data)

في العديد من الحالات، تكون البيانات التي ترغب في قراءتها أو حفظها موجودة في **صيغة نصية** مثل ملفات **TXT** أو **CSV Comma-Separated Values**).

غالبًا ما يُفضّل هذا النوع من الملفات لأنه يمكن **فتحه والتعامل معه خارج بيئة NumPy**، باستخدام تطبيقات أخرى مثل **Excel** أو **Google Sheets** أو حتى **محرر النصوص**، على عكس الملفات الثنائية التي تُستخدم عادة داخل البرامج فقط.

قراءة الملفات ذات البيانات الجدولية

(Reading Files with Tabular Data)

ما هو ملف CSV؟

ملف **CSV** هو ملف نصي بسيط يحتوي على بيانات منظمّة في شكل جدول، حيث تُفصل القيم بالفواصل (,)، بينما تمثّل كل سطر صفّاً من البيانات.

مثلاً:

يمكن أن يحتوي الملف التالي على بيانات درجات طلاب:

name,age,grade

Ali,20,15.5

Sara,22,17.2

Omar,19,12.8

قراءة الملفات ذات البيانات الجدولية

(Reading Files with Tabular Data)

ما هو ملف CSV؟

Fichier

Accueil

Insertion


Mise en page

Formules


Données

Révision


Affichage




Coller



Couper



Copier



Reproduire la mise en forme

Presse-papiers

Calibri

11


A⁺


A⁻


G


I


S





































Police

Ali

A1





Date,Open,High,Low,Close,Adj Close,Volume

A

B

C

D

E

F

1

Date,Open,High,Low,Close,Adj Close,Volume

2

2014-09-17,465.864014,468.174011,452.421997,457.334015,457.334015,21056800

3

2014-09-18,456.859985,456.859985,413.104004,424.440002,424.440002,34483200

4

2014-09-19,424.102997,427.834991,384.532013,394.795990,394.795990,37919700

5

2014-09-20,394.673004,423.295990,389.882996,408.903992,408.903992,36863600

6

2014-09-21,408.084991,412.425995,393.181000,398.821014,398.821014,26580100

7

2014-09-22,399.100006,406.915985,397.130005,402.152008,402.152008,24127600

8

2014-09-23,402.092010,441.557007,396.196991,435.790985,435.790985,45099500

9

2014-09-24,435.751007,436.112000,421.131989,423.204987,423.204987,30627700

قراءة الملفات ذات البيانات الجدولية

(Reading Files with Tabular Data)

ما هو ملف XLSX اكسال؟

EUR_USDDat

Fichier Accueil Insertion Mise en page Formules Données Révision Affichage

J29

$\frac{f}{x}$

	A	B	C	D	E	F	G	H	I
1	Date	Price	Open	High	Low	Vol.	Change %		
2	03/01/201	1,1377	1,1372	1,141	1,1353		0.05%		
3	02/28/201	1,1371	1,1371	1,1421	1,1357		0.02%		
4	02/27/201	1,1369	1,1396	1,1404	1,136		-0.16%		
5	02/26/201	1,1387	1,1366	1,1404	1,1343		0.25%		
6	02/25/201	1,1359	1,1338	1,1369	1,1323		0.14%		
7	02/22/201	1,1343	1,1337	1,136	1,1315		0.06%		
8	02/21/201	1,1336	1,1338	1,1368	1,132		-0.01%		
9	02/20/201	1,1337	1,1342	1,1374	1,1324		-0.04%		
10	02/19/201	1,1341	1,131	1,1358	1,1274		0.27%		
11	02/18/201	1,1311	1,1288	1,1337	1,1284		0.14%		
12	02/15/201	1,1295	1,1295	1,1313	1,1233		-0.05%		
13	02/14/201	1,1301	1,1262	1,1313	1,1247		0.30%		
14	02/13/201	1,1267	1,1326	1,1346	1,126		-0.51%		
15	02/12/201	1,1325	1,128	1,1342	1,1256		0.43%		

واجب منزلي
HomeWork

الواجب المنزلي – Python Libraries for Data Analysis

الموضوع:

تطبيق عملي حول استخدام مكتبات

Matplotlib / Seaborn و NumPy, pandas,

في تحليل بيانات اقتصادية بسيطة.

الواجب المنزلي – Python Libraries for Data Analysis

الأهداف التعليمية

من خلال هذا الواجب، يُتوقع من الطالب أن:

1. يُنشئ مجموعة بيانات اقتصادية بسيطة باستخدام **pandas**.
2. يُطبق عمليات تحليل إحصائي أساسية باستخدام **NumPy**.
3. يُمثّل النتائج في شكل بياني باستخدام **Matplotlib** أو **Seaborn**.
4. يُفسّر الرسوم البيانية بطريقة علمية مختصرة.

الواجب المنزلي – Python Libraries for Data Analysis

المطلوب:

الجزء 1 – إنشاء البيانات

أنشئ DataFrame يحتوي على البيانات التالية حول:

الناتج المحلي الخام (GDP) ومعدل التضخم (Inflation)

استخدم مكتبة **pandas** لإنشاء الجدول وعرضه.

السنة	مليار دولار ((GDP	التضخم (%)
2018	160	2.3
2019	170	2.5
2020	150	4.2
2021	165	3.1
2022	180	2.8
2023	190	2.4

الواجب المنزلي – Python Libraries for Data Analysis

الجزء 2 – التحليل العددي

باستخدام **NumPy** أو **pandas**:

1. احسب متوسط الناتج المحلي الخام.
2. احسب الانحراف المعياري للناتج المحلي.
3. احسب متوسط التضخم خلال الفترة.
4. أضف النتائج أسفل الجدول في الخلية نفسها أو اطبعها منفصلة.

الواجب المنزلي – Python Libraries for Data Analysis

الجزء 3 – التمثيل البياني

باستخدام مكتبة **Seaborn** أو **Matplotlib**:

1. ارسم تطور **GDP** بين 2018–2023 (مخطط خطي).
2. ارسم تطور التضخم في نفس الفترة بلون مختلف.
3. أضف عنوانًا للمخطط ومحوري X و Y بشكل واضح.

الواجب المنزلي – Python Libraries for Data Analysis

الجزء 4 – التحليل والتفسير

1. اكتب فقرة قصيرة (5 إلى 7 أسطر) تفسر فيها النتائج:
2. هل هناك علاقة بين انخفاض الناتج المحلي وارتفاع التضخم؟
3. ما هي السنة التي شهدت أكبر تغير اقتصادي؟
4. كيف يمكن تفسير الاتجاه العام للبيانات؟

الواجب المنزلي – Python Libraries for Data Analysis

إرشادات التقديم

- يرسل الواجب في ملف **Jupyter Notebook** باسم:
- **اسم الطالب(ة) Student-name**
- أو في ملف PDF بعد تصدير النتائج.
- يرسل الواجب على الايميل المهني للأستاذ المحاضر.
- آخر أجل للتسليم: أسبوعان (**01 نوفمبر**).
- تمنح علامة اضافية لكل طالب أرسل الواجب
- الطالب الذي يقدم أداء نوعي ومتميز سيكون له حوافز إضافية في التقييم النهائي للمقياس



**Thank you so much
for your best listening**

حافظوا عليها
Protégez-la

