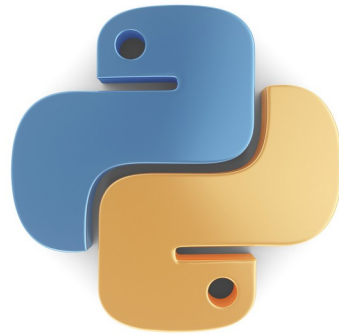


Python & Jupyter



Dott. Alberto Carrera

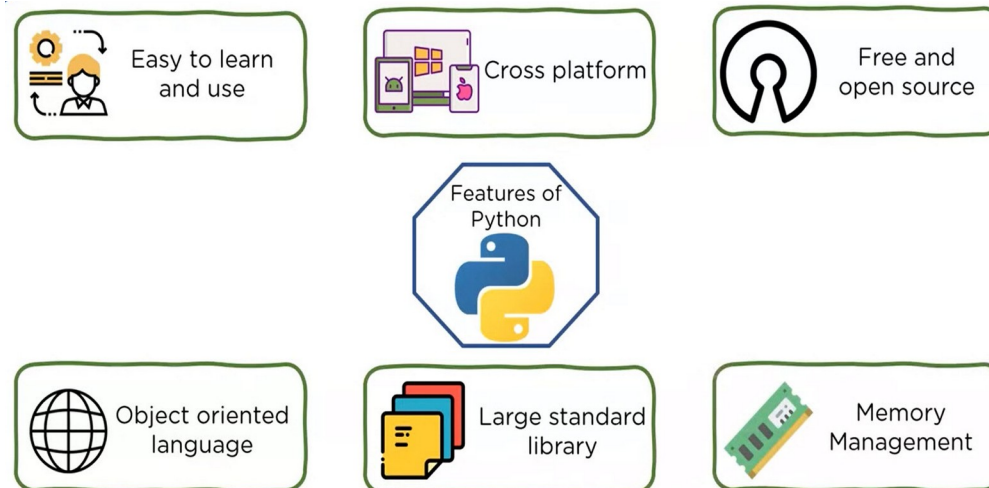


DEPARTMENT
OF GEOSCIENCES
UNIVERSITY OF PADOVA

Python philosophy



- Python is a high-level programming language. It is easy to learn and use, highly readable, and portable across platforms.
- Python is an Object Oriented Programming (OOP) language
- As a scripting language, Python executes instructions directly, yet it is powerful enough to build complex applications — from scientific computing and web development to game creation, machine learning, and data collection.
- Open source and community driven: its large and collaborative community makes it easy to find solutions and resources online.
- “Batteries Included”: a standard distribution (e.g., Anaconda) includes many modules



Modules, packages and libraries



When working with an object-oriented and modular language like Python, it's important to understand the concepts of modules, packages, and libraries.

- A **module** is a single file (.py) containing code (functions, classes, variables) that can be imported into other scripts and reused in any program.

Ex: the math module with the built-in print function to display math.pi (π)

```
[1]: import math
      math.pi

[1]: 3.141592653589793
```

- A **package** is a collection of modules organized in folders

Ex: numpy is a package containing many modules for numerical computing

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

[2]: array([1, 2, 3])
```

- A **library** is a bigger collection of packages and modules designed for a specific purpose — the terms library and package are often used interchangeably.

```
[3]: import pandas as pd # → data analysis
      import matplotlib.pyplot as plt # → plotting
      import scipy # → scientific computing
```

Python includes standard libraries covering most common file handling, math, data processing, system operations, text parsing, and debugging.

Downloading/Running Python



Here's a list of different ways to get Python:

- **Official Python Installer**

Download from <https://www.python.org/downloads/>. Comes with IDLE and the Python terminal.

- **Anaconda Distribution**

Download from <https://www.anaconda.com/download>. Includes Python, Jupyter Notebook, and package management.

- **Miniconda (lightweight Anaconda)**

Smaller installer, only Python + Conda, you add packages as needed. Good if you want a minimal setup.

- **Python via Package Manager (Linux/macOS)**

Linux: `sudo apt install python3` (Ubuntu/Debian)

macOS: `brew install python` (with Homebrew)

- **Online Python (no installation needed)**

Google Colab: <https://colab.research.google.com/>



Python IDEs



IDE stands for **Integrated Development Environment**. It's a software application that helps you write, run, and debug code more easily.

An IDE usually includes a **code editor** (with syntax highlighting), a console or **terminal** to run code, **debugging tools** and extra features (project management, version control, plotting, ...)

Here's a list with the most popular Python IDEs:

- **IDLE** – comes with Python, simple and beginner-friendly
- **Spyder** – scientific Python IDE
- **PyCharm** – full-featured, great for big projects (Community version is free)
- **VS Code** – lightweight, highly customizable
- **Jupyter Notebook / JupyterLab** – interactive code notebooks



Jupyter Notebook



Notebooks are an excellent way to document a scientific process because they allow you to combine nicely formatted **text** with **code** and **results**.

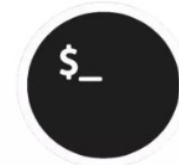
- The Jupyter Notebook is a **web application** that allows you to create and share documents that contain live code, equations, visualizations and explanatory text
- Jupyter Notebooks use a serialized data storage format, JSON, to store the document
- They allow you to integrate many different text processing formats, including HTML, Markdown and LaTeX (a popular word processor in STEM)
- Notebooks are integrated into Anaconda



Code



Rich Text



Terminal-like

Jupyter Notebooks are documents that contain both code and rich text with some of the magic of the terminal

Installing Python through Anaconda



Anaconda is very popular **Python distribution** that includes the Python executor (python.exe), common Python packages, and the Jupyter Notebook IDE (integrated development environment)

- Go to <https://www.anaconda.com/download>
- Select your operating system (Windows, macOS, Linux)
- Download the installer and run it
- Follow the installation prompts (default options are usually fine)



Installing packages




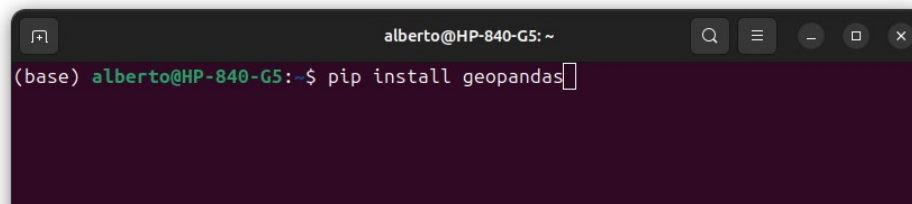
- **pip** is the official **package manager** for Python (stands for “Pip Installs Packages”).
- It allows you to install, upgrade, and manage Python libraries from the Python Package Index (PyPI) or other sources.
- It comes bundled with modern Python versions.
- Works from the terminal or command prompt.
- Can install packages globally or inside virtual environments.

In order to install a new package, open a terminal (macOS/Linux) / Anaconda prompt (Windows) and digit `pip install "packagename"`

```
bash

pip install numpy      # install a package
pip install --upgrade numpy # upgrade a package
pip uninstall numpy    # remove a package
pip list               # list installed packages
```

 Copy code



```
alberto@HP-840-G5: ~
(base) alberto@HP-840-G5:~$ pip install geopandas
```


Creating Virtual Environments



- A **virtual environment** is an isolated Python workspace.
- It lets you install and manage packages separately from your system's global Python.
- Each virtual environment has its own Python interpreter and libraries.
- It avoids conflicts between packages required by different projects.
- Lets you test different Python versions or package versions safely.

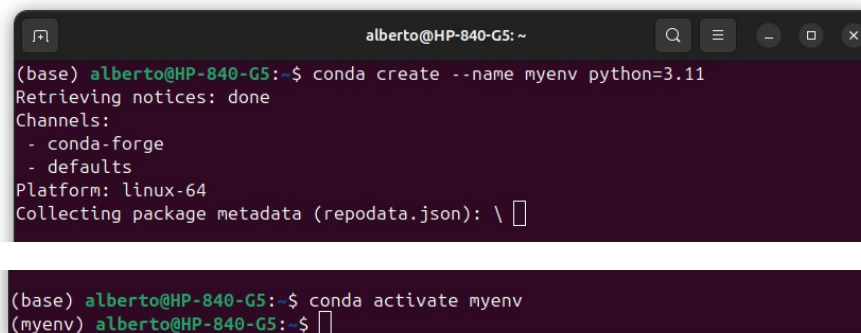
In order to create a new environment, open a terminal (macOS/Linux) / Anaconda prompt (Windows) and digit `conda create --name "myenv"`

(more details here <https://docs.conda.io/projects/conda/en/latest/user-guide/tasks/manage-environments.html>)

bash

 Copy code

```
conda create --name myenv python=3.11 # create environment
conda activate myenv                  # activate
conda deactivate                       # deactivate
```



```
alberto@HP-840-G5: ~
(base) alberto@HP-840-G5:~$ conda create --name myenv python=3.11
Retrieving notices: done
Channels:
- conda-forge
- defaults
Platform: linux-64
Collecting package metadata (repodata.json): \
(base) alberto@HP-840-G5:~$ conda activate myenv
(myenv) alberto@HP-840-G5:~$
```

Launch Jupyter



Jupyter Notebook is the classic, simple interface for running Python code in cells.

A more modern and powerful version is represented by **JupyterLab**. It lets you open multiple notebooks, terminals, text files, and plots side by side, all within one flexible workspace.

To open JupyterLab and start working with your notebook you have to

- Open your terminal
On Windows: open Anaconda Prompt
On macOS/Linux: open a regular terminal
- Activate your conda environment (skip this if you're using the base environment)
`conda activate myenv`
- Launch JupyterLab
`jupyter lab`

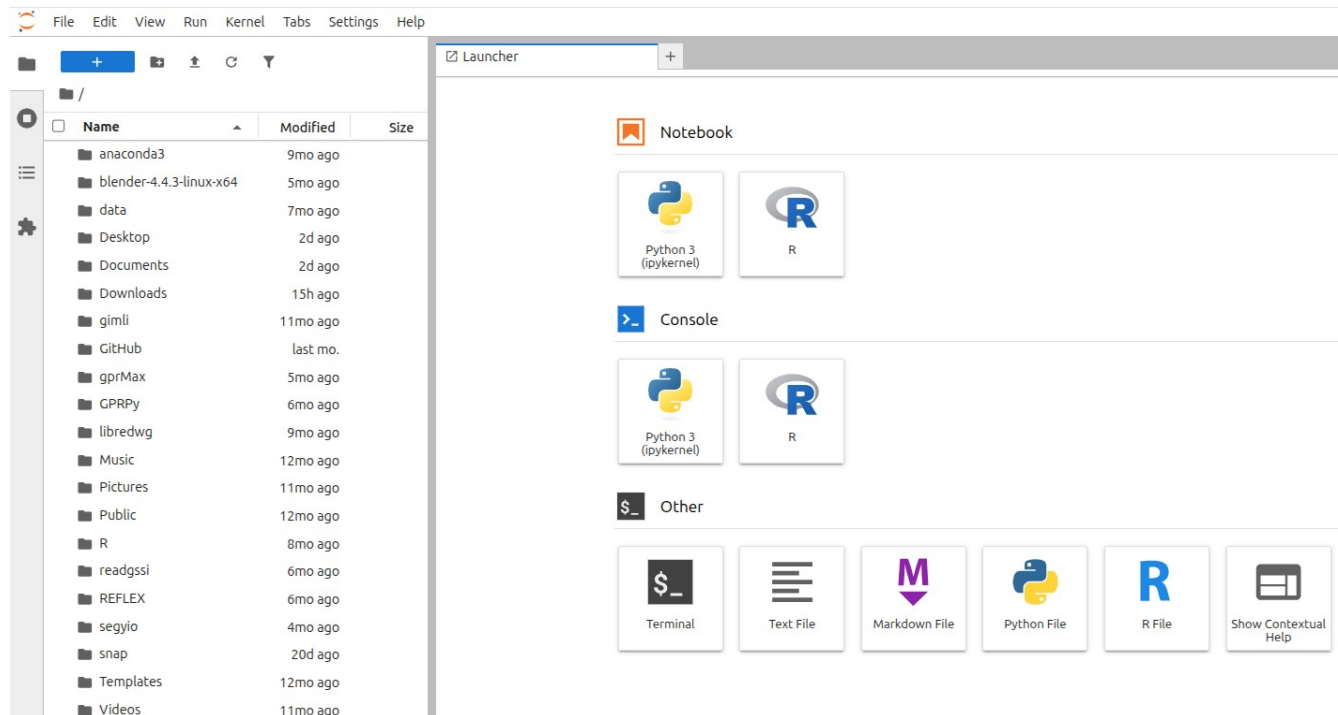


```
alberto@HP-840-G5: ~  
(base) alberto@HP-840-G5:~$ conda activate electro  
(electro) alberto@HP-840-G5:~$ jupyter lab  
[I 2025-10-09 09:29:29.753 ServerApp] jupyter_lsp | extension was successfully l  
inked.  
[I 2025-10-09 09:29:29.754 ServerApp] jupyter_server_proxy | extension was succe  
ssfully linked.  
[I 2025-10-09 09:29:29.757 ServerApp] jupyter_server_terminals | extension was s
```

Launch Jupyter



Browser opens automatically at <http://localhost:8888/lab> (by default).
You'll see your file explorer and can start coding in new notebooks (.ipynb files).

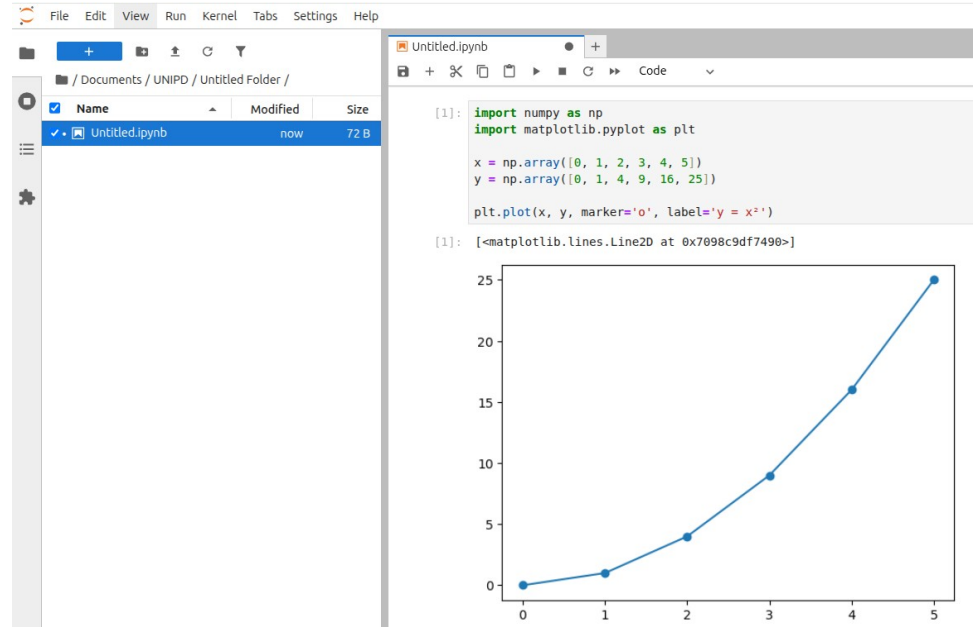
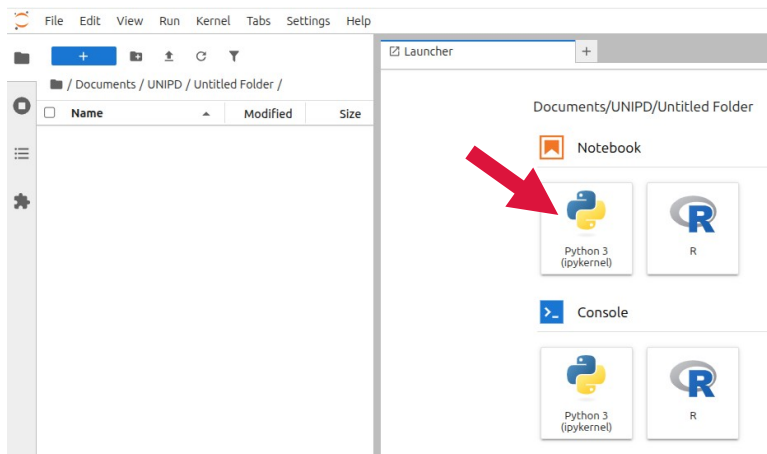


Launch Jupyter



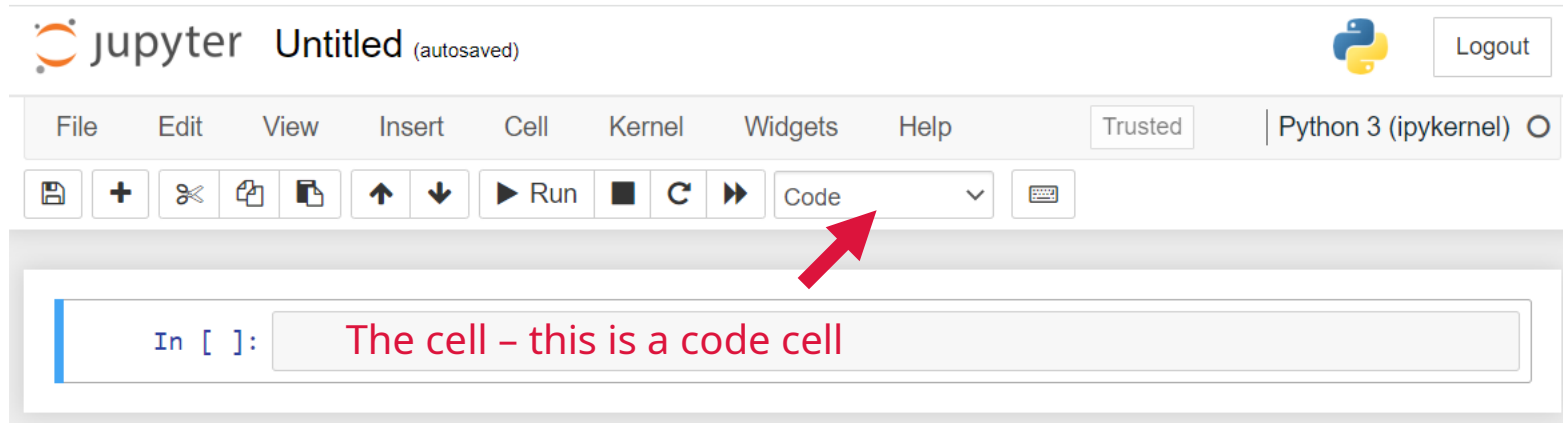
Navigate around in your file system – create new folders or text files, upload other files or use the normal drag and drop of your OS file navigation system.

Create your notebook by clicking the **Python 3 (ipykernel)** button in the Notebook menu. Then, you can start interacting with the environment.



Before proceeding, define the name of your notebook from the menu on the left.

Jupyter Notebook – notebook screen



The cell is the basic unit in the Notebook, which are small editable blocks of content.

- **Code** cells: where you write and run Python code.
- **Markdown** cells: for text, titles, math, and explanations (using Markdown syntax).
- **Raw** cells: plain text that Jupyter does not execute or format — they're left "as is."

To run a cell, click inside the cell and press the ► Run button or Ctrl + Enter.
The output area shows results, plots, or errors from code cells.

Jupyter Notebook – the cell



Cell in command mode (markdown)

Only one cell can be active at a time. It is clear which one is active due to the colored blue line and the box (command mode).

In command mode all actions are done on the cell; move it up or down, copy, paste, cut, delete, insert, run, etc.

When Running a markdown cell, the text is being formatted and the grey box around the cell disappears. The cell keeps this run-look until you click on the text for more editing.

The text in Markdown can be formatted in many different ways, and images can also be input. This is achieved by using HTML, the language for web pages.

The size of the header is determined by the number of #'s.

Writing a headline

Headline when cell has been run

And some additional text in the same cell

Jupyter Notebook – the cell



```
In [ ]: print("Hello World")
```

```
In [ ]: print("Hello World")
```

```
In [1]: print("Hello World")
```

Hello World

When using *code* instead of markdown, notice the syntax highlighting. It helps you distinguish between syntactical elements of the language, here the function in **green** and the string (text) in **red**.

The **In []:** indicates that it is a code cell. The number in the brackets is the order of execution of the cells, here **[1]** means it was the first cell to be executed.

If you see **[*]** instead, it means the cell is currently executing. It can be interrupted using the menu.

You just learned your first python - the **print** function - which outputs data, usually to the screen.

Jupyter Notebook – the cell



```
In [1]: number = 0
```

```
In [6]: number = number + 1
```

```
In [7]: print(number)
```

5

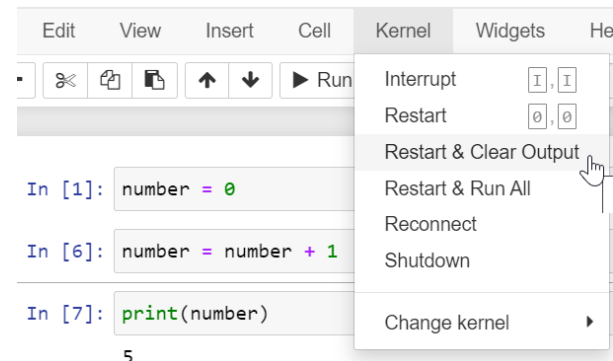
It is important to understand that the notebook remembers the result of every cell that has been executed and that it builds upon previous results.

As can be seen from above example, the top cell was executed first, setting the variable `number` to 0.

Then the middle cell was executed 5 times (hence the number is [6], having gone through 2 to 6), increasing the value of the variable `number` with 1 each time. Finally, the number was `printed` – and it was 5.

You can get a fresh start by selecting the Restart & Clear Output in the menu.

And you learned about variables.....



Jupyter Notebook – the cell



Normally you have many lines of python code in a cell. Every cell has a specific purpose or function and with the markdown cells you can document the code and explain the process.

This is the strength of the Notebook, when using it in a data exploration fashion.

Here getting `input` to your program is introduced.

```
In [*]: name = input("What is your name: ")  
        print("Hello", name)
```

What is your name:

```
In [1]: name = input("What is your name: ")  
        print("Hello", name)
```

What is your name: Peter
Hello Peter

A code cell can also be used as a calculator, as shown. It can even using variables from other cells – if they have been run.

```
In [1]: 4+5*6
```

Out[1]: 34

```
In [2]: number = 42
```

```
In [3]: 4*number
```

Out[3]: 168

Jupyter Notebook – the cell



Three steps to stopping the notebook:

- save your notebook
- go back to the terminal and press Ctrl + C (confirm with y)

```
alberto@HP-840-G5: ~  
bf4b-5cfb573afef8:000f8bd5-2023-44ba-ba79-eec39801652c  
[W 2025-10-09 10:59:07.459 ServerApp] delete /Documents/UNIPD/Untitled Folder  
^C[I 2025-10-09 10:59:21.305 ServerApp] interrupted  
[I 2025-10-09 10:59:21.305 ServerApp] Serving notebooks from local directory: /home/alberto  
4 active kernels  
Jupyter Server 2.15.0 is running at:  
http://localhost:8888/lab?token=de052ffc63c8977647834bafae1a600a221a885d53184f1  
http://127.0.0.1:8888/lab?token=de052ffc63c8977647834bafae1a600a221a885d53184f1  
Shut down this Jupyter server (y/[n])? [I 2025-10-09 10:59:26.310 ServerApp] No answer for 5s:  
[I 2025-10-09 10:59:26.310 ServerApp] resuming operation...  
^C[I 2025-10-09 11:00:11.528 ServerApp] interrupted  
[I 2025-10-09 11:00:11.529 ServerApp] Serving notebooks from local directory: /home/alberto  
4 active kernels  
Jupyter Server 2.15.0 is running at:  
http://localhost:8888/lab?token=de052ffc63c8977647834bafae1a600a221a885d53184f1  
http://127.0.0.1:8888/lab?token=de052ffc63c8977647834bafae1a600a221a885d53184f1  
Shut down this Jupyter server (y/[n])? 
```

It is not correct to just shutdown your browser window. The notebook server will continue to run and so will the notebook/kernel.

Jupyter Notebook – the cell

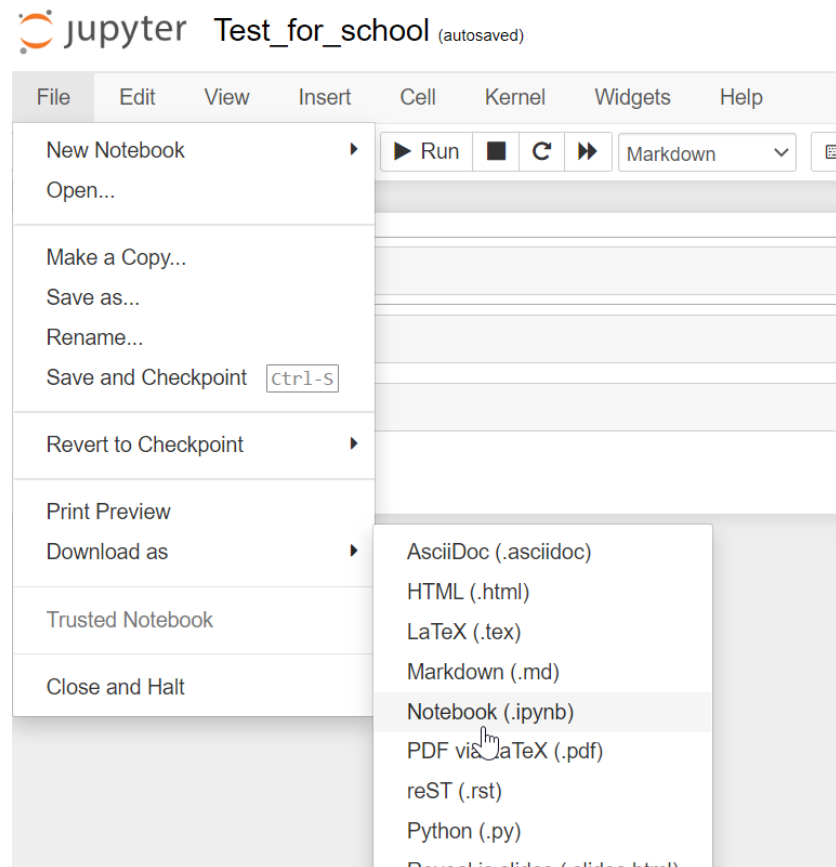


If you want to share your notebook with other people, then it is easy enough to download.

In the menu:

File ->
Download as ->
Notebook

You can choose other formats, as can be seen.



Learning Resources



- **Official Python Tutorial (Python.org)** - a clear, structured introduction straight from the official Python documentation.
<https://docs.python.org/3/tutorial/>
- **W3Schools Python Tutorial** - interactive lessons, ideal for absolute beginners.
<https://www.w3schools.com/python/>
- **Real Python** - high-quality articles, tutorials, and projects for all levels.
<https://realpython.com/>
- **Kaggle Learn: Python** - short, hands-on courses with built-in coding exercises.
<https://www.kaggle.com/learn/python>
- **Google's Python Class** - a free Python course with slides, videos, and practice exercises.
<https://developers.google.com/edu/python/>
- **FreeCodeCamp: Python Full Course (YouTube)** - a full 4-hour video course covering Python basics with live example.
<https://www.youtube.com/watch?v=rfscVS0vtbw>
- **O'Reilly: Learning Python** - one of the most comprehensive Python books — excellent for solid foundations.
<https://www.oreilly.com/library/view/learning-python-5th/9781449355722/>
- **Project Jupyter** - official documentation and setup guide.
<https://docs.jupyter.org/en/latest/>
- **Real Python – Jupyter Introduction** - great tutorial for learning how to use notebooks effectively.
<https://realpython.com/jupyter-notebook-introduction/>
- **DataCamp Jupyter Tutorial** - clear overview with examples and tips.
<https://www.datacamp.com/tutorial/tutorial-jupyter-notebook>