

Introduction to Python: Why, Install & Jupyter Notebooks

Learn what programming and Python are, how Python compares with other languages, how code runs (interpreters, compilers, bytecode), the properties of Python, and set up your environment (Anaconda, VS Code, Jupyter, Google Colab). Write and run your first cells, with markdown, magic commands and good notebook habits.

● Python class 1

✦ Python 3.13, Jupyter, VS Code, Google Colab (2026)

Beginner

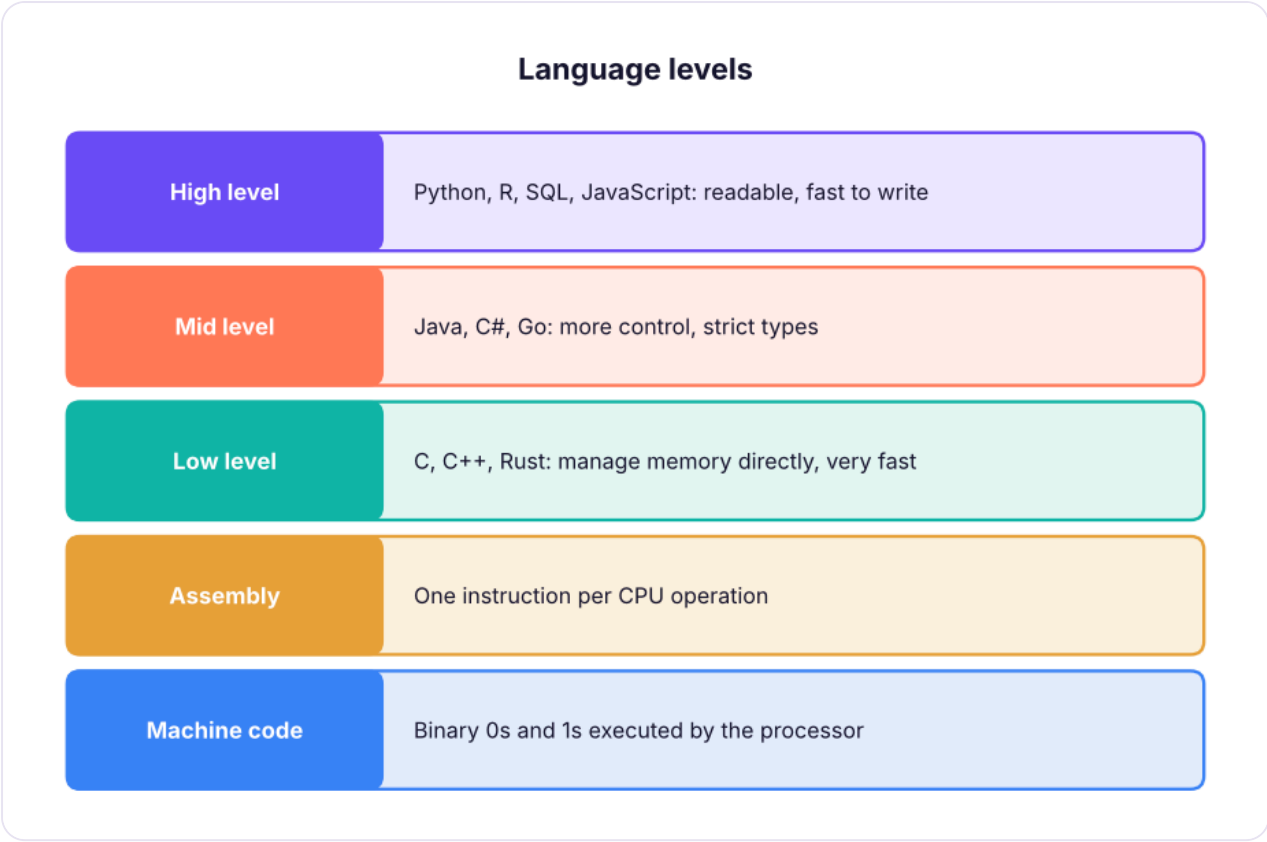
🕒 90 min

By the end of this lesson you can

- Explain what a programming language is and why Python is the default language for data work
- Describe the difference between compiled and interpreted languages and how Python runs your code
- Install Python and Jupyter (or use Google Colab) and run code in cells
- Use Markdown cells, keyboard shortcuts, magic commands and restart-and-run-all
- Follow good habits: naming, comments, reproducibility and virtual environments

What is programming?

A **program** is a set of precise instructions a computer follows. Computers only understand machine code (0s and 1s), so we write instructions in a **programming language** that humans can read, and a translator turns them into something the machine can run.



Languages sit on levels. High-level languages like Python are close to human language; low-level ones are close to the machine.

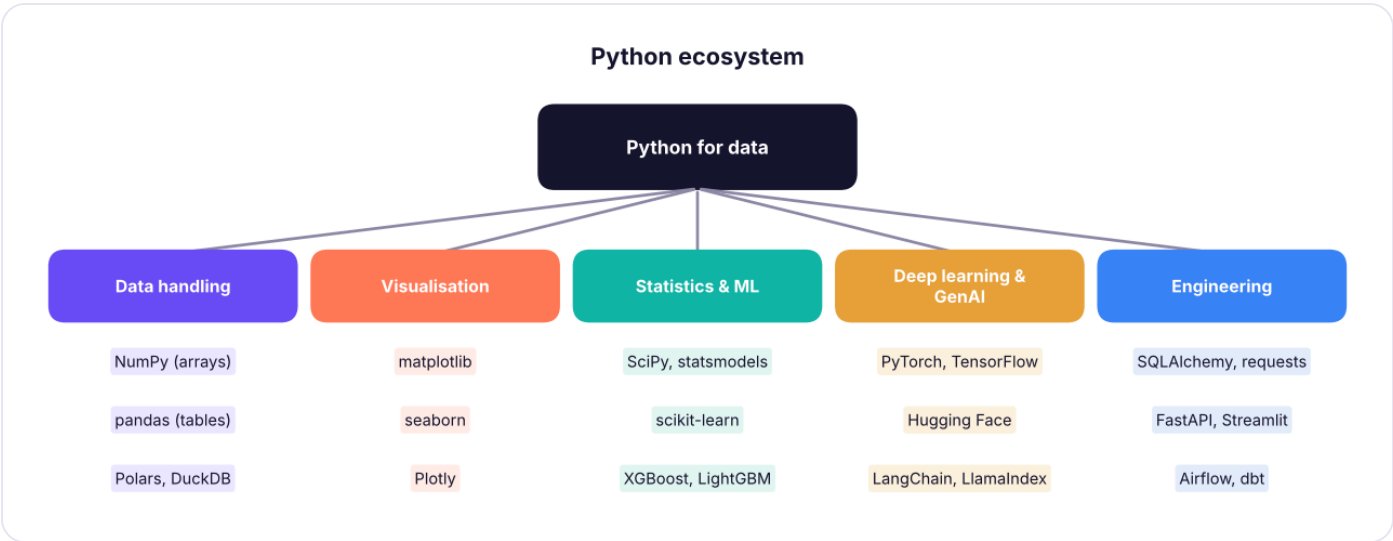
Compiled vs interpreted

	Compiled (C, C++, Rust, Go)	Interpreted (Python, JavaScript, R)
How it runs	Whole program translated to machine code before running	Translated and executed line by line while running
Speed	Usually faster at run time	Slower per operation, much faster to develop
Errors	Many caught before running	Some appear only when that line runs
Output	A separate executable file	Source file runs under an interpreter

Python is usually called **interpreted**. In reality CPython (the standard version) first compiles your code to **bytecode** (`.pyc` files) and then a virtual machine executes it. Python code that needs speed calls libraries written in C (NumPy, pandas), so data work is fast anyway.



Why Python for data?



The Python data ecosystem. You will learn each of these libraries in this track.

Property of Python	Meaning
Readable	Indentation and English -like keywords; fewer lines than most languages
Interpreted, dynamically typed	No compile step; variables take the type of the value they hold
High level	Memory is managed for you (garbage collection)
Multi -paradigm	Procedural, object -oriented and functional styles
Huge ecosystem	500,000+ packages on PyPI; pandas, scikit -learn, PyTorch lead data and AI
Open source and portable	Free; runs on Windows, Mac, Linux, cloud notebooks
Great glue	Calls SQL, APIs, C/C++ code, Excel, Spark

NOTE

Python versions

Use Python **3.11, 3.12 or 3.13** (the current stable series in 2026). Python 2 ended in 2020 and must not be used. Check your version with `python --version` .

Getting set up: three options

Option	Best for	Steps
Google Colab (free, in the browser)	Starting today, no installation, GPU for ML	Go to colab.research.google.com → New notebook → sign in with Google
Anaconda / Miniconda	Full local data - science setup with packages and environments	Install Anaconda Distribution → open Anaconda Navigator or a terminal → <code>jupyter lab</code>
Python + VS Code	Professional workflow, notebooks and scripts in one editor	Install Python from python.org (tick "Add to PATH") → install VS Code → add the Python and Jupyter extensions

STEP BY STEP

Local install with pip (any OS)

1. Install Python 3.12+ from python.org (Windows: tick **Add python.exe to PATH**).

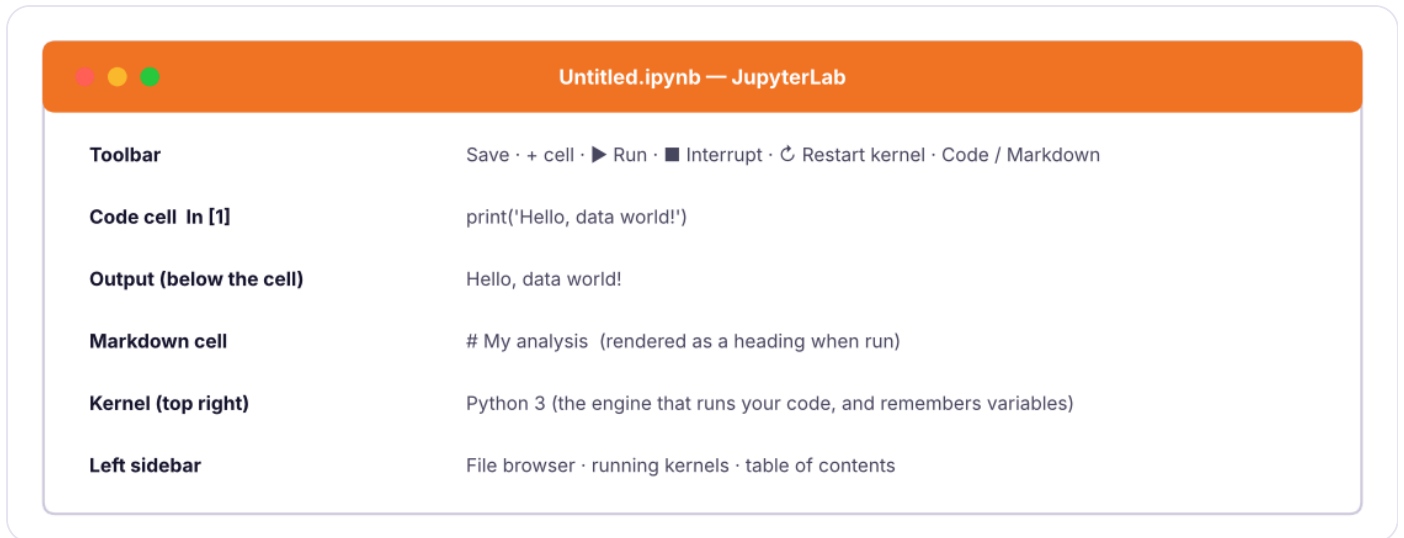
2. Open a terminal and create a **virtual environment** (an isolated set of packages for one project): `python -m venv .venv` then activate it: Windows `.venv\Scripts\activate` ; Mac/Linux `source .venv/bin/activate` .

3. Install the course packages: `pip install jupyterlab numpy pandas matplotlib seaborn scikit-learn scipy statsmodels`

4. Start Jupyter: `jupyter lab` (opens in your browser).

Jupyter notebooks

A **Jupyter notebook** (`.ipynb`) mixes **code cells**, **Markdown cells** (text, headings, images) and **outputs** (tables, charts) in one document. It is the standard tool for exploring data and sharing analysis. The name comes from Julia, Python and R.



Anatomy of a Jupyter notebook.

The kernel

The **kernel** is the Python process behind the notebook. Variables you create stay in its memory until you restart it. Cells can be run in any order, which is flexible but dangerous: a notebook can look right while depending on a cell you deleted. **Before sharing, always Restart Kernel and Run All** to prove the notebook works top to bottom.

Essential shortcuts (Jupyter / Colab)

Action	Shortcut
Run cell, stay	Ctrl + Enter
Run cell, go to next	Shift + Enter
Insert cell above / below (command mode)	A / B
Delete cell (command mode)	D, D
Convert to Markdown / Code	M / Y
Command mode / edit mode	Esc / Enter
Comment / uncomment lines	Ctrl + /
Show function help	Shift + Tab (inside the brackets)
Autocomplete	Tab
Undo cell delete	Z (command mode)

Your first code

Python

```
print("Hello, data world!")
```

OUTPUT

Hello, data world!

Python is also a calculator:

Python

```
2 + 3 * 4
```

OUTPUT

14

Python

```
(2 + 3) * 4, 10 / 4, 10 // 4, 10 % 4, 2 ** 10
```

OUTPUT

(20, 2.5, 2, 2, 1024)

The **last expression** in a notebook cell is displayed automatically; to show several things use `print()` .

Python

```
name = "Varun"
learners = 25
print("Welcome to the course,", name)
print(f"This batch has {learners} learners")
```

OUTPUT

Welcome to the course, Varun
This batch has 25 learners

Comments

Python

```
# This is a comment: Python ignores everything after the hash
price = 120 # you can also put a comment at the end of a line
"""A triple-quoted string on its own line is often used
as a multi-line comment or docstring"""
```

Write comments to explain **why**, not what. `x = x + 1 # add one` is noise; `# compensate for the header row` is useful.

Errors are normal

Every programmer sees errors daily. Learn to read them from the **bottom line up**:

Python

```
print("Total:", total_amount)
```

OUTPUT

```
NameError: name 'total_amount' is not defined
```

`NameError: name 'total_amount' is not defined` says exactly what happened: we used a name we never created. Common beginner errors:

Error	Usual cause
<code>NameError</code>	Typo, or the cell that defines the variable has not been run
<code>SyntaxError</code>	Missing bracket, quote or colon; wrong indentation
<code>TypeError</code>	Using a value of the wrong type (<code>"5" + 3</code>)
<code>ValueError</code>	Right type, wrong value (<code>int("abc")</code>)
<code>IndexError</code> , <code>KeyError</code>	Asking for a position or key that does not exist
<code>FileNotFoundError</code>	Wrong path or the file is not where you think

TIP How to ask for help

Copy the **last line** of the error, search it, or paste the code and the error into an AI assistant (without secrets) and ask: *"What does this error mean and how do I fix it? Explain step by step."* Then fix it **yourself** and re-run: you learn by doing.

Markdown cells

Change a cell to **Markdown** to write notes next to your code:

MARKDOWN

```
# Title
## Section
Some bold and italic text, a [link](https://example.com), and a list:
- item one
- item two

| Column | Value |
|---|---|
| A | 1 |
```

Good notebooks tell a story: a title, the question, the data, the steps with short explanations, and a conclusion.

Magic commands and useful tools

Jupyter adds special commands beginning with `%`:

Command	Purpose
<code>%timeit expr</code>	Time a short expression precisely
<code>%%time</code>	Time a whole cell
<code>%pwd</code> , <code>%ls</code>	Current folder, list files
<code>%matplotlib inline</code>	Show charts in the notebook (default now)
<code>%who</code>	List variables
<code>!pip install package</code>	Run a shell command (install a package) from a cell

Reproducibility habits

1. **Folder per project:** `data/` , `notebooks/` , `src/` , `README.md` .
2. **Virtual environment per project** and a `requirements.txt` (`pip freeze > requirements.txt`).
3. **Use relative paths** (`data/sales.csv`), not `C:\Users\you\Desktop\...` .
4. **Set random seeds** (`np.random.seed(42)`) so results repeat.
5. **Restart and Run All** before sharing; keep notebooks short and focused.
6. **Version control:** commit to Git; clear outputs of big notebooks.
7. **Do not commit** data with personal information or secrets (API keys); use environment variables.

DO IT WITH AI Do it with AI

Ask an assistant: *"I'm a beginner. Give me a 7-day practice plan for Python with 20 minutes a day, each day with three small exercises and answers I can check."* Use it to **practise**, not to skip the struggle: type the code, break it on purpose, and read the errors.

Summary

KEY TAKEAWAYS

- A program is a set of instructions; Python is a **high-level, interpreted, readable** language with a huge data ecosystem (NumPy, pandas, scikit-learn, PyTorch).
- CPython compiles source to bytecode and runs it on a virtual machine; speed-critical libraries use C underneath.
- Start with **Google Colab** or install **Anaconda** or **Python + VS Code**; isolate projects with **virtual environments**.
- A Jupyter notebook mixes code, Markdown and output; the **kernel** holds state, so *Restart and Run All* before sharing.
- Read error messages from the bottom; learn the common error types.
- Use clear comments, relative paths, seeds and version control for reproducible work.

Quick quiz

6 questions · tap an answer to check it

1. Which best describes Python?

- A A low-level compiled language
- B A high-level, interpreted, general-purpose language with a rich data ecosystem**
- C A database language
- D A spreadsheet tool

Answer: B. Python is readable, dynamically typed and widely used for data.

2. What does the kernel do in a notebook?

- A Formats Markdown
- B Runs your code and keeps variables in memory**
- C Stores the file
- D Draws charts only

Answer: B. The kernel is the Python process behind the notebook.

3. Why run "Restart kernel and Run All" before sharing a notebook?

- A To make it smaller
- B To confirm the notebook works from top to bottom without hidden state**
- C To delete output
- D To change the Python version

Answer: B. Cells can be run out of order; this test removes that risk.

4. What does `NameError: name 'x' is not defined` mean?

A x is a bad name

B Python has no variable called x yet (typo or the defining cell was not run)

C x must be a number

D A syntax mistake

Answer: B. You used a name before creating it.

5. Which command installs a package from inside a notebook cell?

A `install pandas`

B `!pip install pandas`

C `%pip.exe pandas`

D `import install`

Answer: B. A leading `!` runs a shell command.

6. What is a virtual environment for?

A Faster internet

B Keeping the packages of one project separate from others

C Running Python 2

D Backing up data

Answer: B. Isolated environments avoid version conflicts.

EXERCISE Set up and run

(1) Open Colab or Jupyter and create a notebook called `day1.ipynb`. (2) Add a Markdown cell with a title and your goals. (3) In code cells, print your name, compute the area of a rectangle 12×7.5 , and print the Python version (`import sys; print(sys.version)`). (4) Cause a `NameError` on purpose and fix it. (5) Restart and Run All.

Show solution (worked answer)

Python

```
print("Asha Rao")
length, width = 12, 7.5
print("Area:", length * width)           # 90.0
import sys; print(sys.version)
print(undefined_name)                   # NameError: name 'undefined_name' is not
defined
undefined_name = 5; print(undefined_name)
```

INTERVIEW CORNER Questions you may be asked

Q: Why is Python popular for data science? Readable syntax, a huge ecosystem (NumPy, pandas, scikit-learn, PyTorch), strong community, easy integration with SQL, APIs and cloud, and notebooks for exploration.

Q: Is Python compiled or interpreted? Both: CPython compiles source to bytecode, which the Python virtual machine interprets. It is called interpreted because there is no separate manual compile step.

Q: What is a virtual environment? An isolated Python installation with its own packages for one project (created with `venv` or `conda`), preventing dependency conflicts.

Q: Notebook vs script? Notebooks are interactive and great for exploration and communication; scripts (`.py`) are better for production code, testing and automation.

Practise more online

Free, trusted places to test yourself on this topic. Links open the original site in a new tab.