

Python – Introduction


What is Python?

Python is a **high-level, interpreted, general-purpose programming language** that emphasizes code readability and simplicity.

It was designed to be easy to learn and use, making it a top choice for beginners while still being powerful enough for experts. Python supports multiple programming paradigms, including **procedural, object-oriented, and functional programming**.

- It is often called a “**batteries-included**” language because of its vast standard library.
- Python is used in diverse fields like **web development, data science, machine learning, AI, scientific computing, automation, scripting, and cloud development**.

Why is it called Python?

The name **Python** does not come from the snake .

- Guido van Rossum, Python’s creator, was a fan of the **British comedy series “Monty Python’s Flying Circus”**.
- While searching for a name, he wanted something **short, unique, and a little mysterious** — so he chose *Python*.
- Even today, many Python tutorials contain Monty Python jokes as a tribute.

Python History

- **1980s:** Guido van Rossum, working at CWI (Centrum Wiskunde & Informatica) in the Netherlands, started developing Python as a successor to the ABC language.
- **1991:** First official release, Python 0.9.0, included features like functions, exception handling, and core data types (str, list, dict).
- **2000:** Release of Python 2.0 — introduced list comprehensions, garbage collection.
- **2008:** Python 3.0 released (major shift; not backward compatible).
- **2020:** Python 2 officially discontinued.
- **Today:** Python 3.x dominates, with regular updates improving speed, libraries, and typing features.

Why Learn Python?

- **Beginner-friendly:** Simple syntax close to English.
- **Versatile:** Used in web dev, AI, ML, data science, automation, scripting.
- **Huge community:** Millions of developers contribute tutorials, forums, and libraries.
- **Rich libraries/frameworks:** Django, Flask, NumPy, Pandas, TensorFlow, PyTorch.
- **Cross-platform:** Works on Windows, Linux, macOS, even mobile.

- **High demand in jobs:** Python is one of the most sought-after skills in IT and data-related fields.

Features of Python

- **Simple & Easy to Learn** – Syntax is clear and close to English.
- **Interpreted Language** – No compilation step; code runs line by line.
- **Cross-platform** – Works on Windows, Linux, macOS.
- **Open Source** – Free to use and modify.
- **High-level Language** – Programmer-friendly, abstracts low-level details.
- **Extensive Standard Library** – “Batteries included.”
- **Object-Oriented** – Supports classes, inheritance, encapsulation.
- **Dynamic Typing** – No need to declare variable types explicitly.
- **Automatic Memory Management** – Garbage collection is built-in.
- **Supports Multiple Paradigms** – Procedural, functional, object-oriented.
- **Embeddable & Extensible** – Can be combined with C/C++/Java.
- **Huge Ecosystem** – Third-party packages available via PyPI.

Applications of Python

- **Web Development:** Django, Flask, FastAPI.
- **Data Science & Machine Learning:** NumPy, Pandas, Scikit-learn, TensorFlow.
- **Artificial Intelligence:** Deep learning frameworks (PyTorch, Keras).
- **Automation & Scripting:** Automating repetitive tasks, bots.
- **Game Development:** Pygame.
- **Desktop GUI:** Tkinter, PyQt.
- **Cybersecurity:** Pen-testing scripts, security tools.
- **Cloud & DevOps:** AWS SDKs, automation tools.
- **IoT & Embedded Systems:** MicroPython.
- **Scientific Computing:** SciPy, Matplotlib.

Architecture and Working of Python

Let's now talk about Python architecture and its usual flow –

a. Parser

It uses the source code to generate an abstract syntax tree.

b. Compiler

It turns the abstract syntax tree into Python bytecode.

c. Interpreter

It executes the code line by line in a REPL (*Read-Evaluate-Print-Loop*) fashion.

- **Python Interpreter** executes code line by line.

- **Python Execution Flow:**
 1. Source Code (.py file)
 2. Python Compiler → Bytecode (.pyc)
 3. Python Virtual Machine (PVM) executes bytecode.
 4. If needed, C libraries are called.
- **Memory Management:** Uses **heap space + garbage collector**.
- **CPython:** The most widely used reference implementation.

Pros and Cons of Python

✓ Advantages (Why Python Dominates)

- **Easy to learn and read** – The syntax is simple, close to English, and beginner-friendly, which makes it great for fast learning.
- **Rich ecosystem of libraries** – From NumPy for data science to Django for web development, Python has libraries for almost everything.
- **High productivity (fewer lines of code)** – Developers can solve problems with fewer lines of code compared to Java or C++, saving time.
- **Cross-domain applications** – Python is widely used in web development, AI, ML, data science, automation, game dev, and more.
- **Huge community support** – A massive global community provides tutorials, forums, and open-source contributions, making problem-solving easier.

✗ Disadvantages

- **Slower compared to compiled languages (like C/Java)** – Because Python is interpreted, execution time is slower than compiled languages.
- **High memory usage** – Python's flexibility and dynamic nature use more memory, which can be a drawback in resource-limited environments.
- **Not great for mobile development** – Few frameworks exist for mobile apps, so languages like Kotlin/Swift are preferred in this domain.
- **Dynamic typing may cause runtime errors** – Since variables don't require explicit types, unexpected errors may occur during program execution.

Python Versions

- **Python 1.x (1991–2000):** Early versions, basic features.
- **Python 2.x (2000–2020):** Introduced list comprehensions, Unicode support.
- **Python 3.x (2008–present):** Not backward compatible, modernized language, async support, type hints.

Comparison Point:

- `print "Hello"` works in Python 2, but in Python 3 it must be `print("Hello")`.

- Python 3 uses Unicode by default, making it more globalized.
- Python 2 is obsolete, Python 3 is the standard.

Python Coding Environments

Python can be installed and accessed in multiple ways depending on your use case. Some options are beginner-friendly, while others are more suited for advanced development and data science.

1. Official Python Installation (python.org)

- Go to the [official Python website](https://www.python.org).
- Download the installer for your operating system (Windows, macOS, or Linux).
- During installation, make sure to **check the box “Add Python to PATH”** so you can run Python from the terminal/command prompt.
- Once installed, you can type `python` or `python3` in your terminal to start coding.

Points:

- Direct way to install the latest Python version.
- Lightweight and flexible.
- Good for learning core Python or when you want a clean setup.

2. IDLE (Integrated Development and Learning Environment)

- IDLE comes bundled with Python when you install it from python.org.
- After installation, search for “IDLE” in your system and open it.
- It provides a simple editor and an interactive shell where you can write and run Python code immediately.

Points:

- No extra installation needed.
- Great for beginners who want a simple editor.
- Not ideal for large projects due to limited features.

3. Jupyter Notebook

- Install using `pip install notebook` or through Anaconda (recommended for data science).
- Launch with `jupyter notebook` command, which opens in your browser.
- Code is written in **cells**, making it easy to test small chunks of code and visualize results instantly.

Points:

- Perfect for data science, ML, and interactive coding.
- Supports inline visualizations (graphs, plots).
- Great for documenting code with Markdown + code in the same file.

4. Anaconda Distribution

- Download Anaconda from the official Anaconda website.
- During setup, it installs Python, Jupyter, and many scientific libraries (NumPy, Pandas, Matplotlib, etc.) in one go.
- Anaconda Navigator (GUI) makes it easy to manage environments and launch tools.

✓ Points:

- Best for data science and ML users.
- Comes with preinstalled packages, saving time.
- Large in size (needs more storage).

5. Google Colab (Cloud-based)

- Go to Google Colab.
- Sign in with your Google account.
- Start a new notebook and write Python code directly in your browser.
- Runs on Google's cloud servers, so no local setup is required.

✓ Points:

- Free cloud-based environment.
- Provides GPU/TPU support for ML tasks.
- Great for quick experiments and sharing notebooks.

6. VS Code / PyCharm (Professional IDEs)

- **VS Code:** Download from Visual Studio Code website. Install the **Python extension** to enable debugging, auto-complete, and environment management.
- **PyCharm:** Download from [JetBrains](https://www.jetbrains.com/pycharm/). It has a free Community Edition and a paid Professional Edition.

✓ Points:

- Feature-rich editors suitable for real-world projects.
- Debugging, testing, and Git integration available.
- Ideal for intermediate and advanced developers.

👉 This way, learners can pick **the method that suits their goals**:

- **Beginners:** python.org + IDLE
- **Students/Data Scientists:** Jupyter / Anaconda / Colab
- **Professional developers:** VS Code / PyCharm