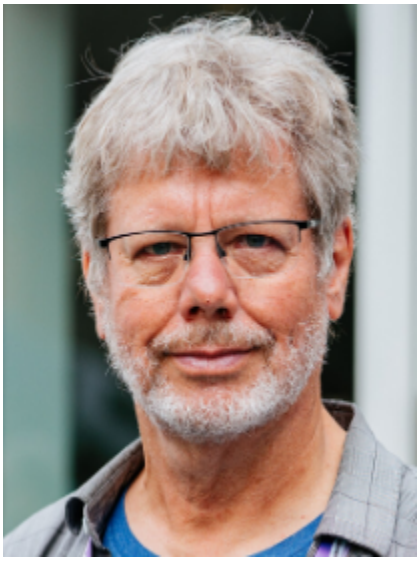
❖ What Is Python?



Python is a **high-level, general-purpose programming language** known for its simplicity, readability, and versatility. It was created by **Guido van Rossum** and first released in **1991**. Python is managed by the **Python Software Foundation** and is open source, meaning it's free to use, modify, and distribute.

🧠 **Design Philosophy**

Python emphasizes:

- **Code readability**: Uses indentation instead of braces or keywords to define blocks.
- **Simplicity and elegance**: Encourages writing clean and understandable code.
- **Versatility**: Supports multiple programming paradigms—procedural, object-oriented, and functional.

Major Implementations

- **CPython**: The default and most widely used implementation.
- **PyPy**: Focuses on speed using Just-In-Time (JIT) compilation.
- **Jython**: Integrates with Java.
- **IronPython**: Integrates with .NET framework.
- **MicroPython**: Designed for microcontrollers.

Brief History

- **1980s**: Guido van Rossum began developing Python as a successor to the ABC language.
- **1991**: Python 0.9.0 released.
- **2008**: Python 3.0 introduced major changes, not backward compatible with Python 2.
- **2025**: Latest stable release is Python 3.13.7.

Real-World Applications

- **Web Development**: Django, Flask
- **Data Science & AI**: Pandas, NumPy, Scikit-learn, TensorFlow

- **Automation & Scripting**: Automate tasks, scrape websites, manage files
- **Game Development**: Pygame
- **Education**: Widely used as a first programming language due to its simplicity

Famous Companies Using Python

- **YouTube**: Backend services
- **Instagram**: Scalability and simplicity
- **Spotify**: Machine learning and backend
- **Google**: Internal tools and APIs

❖ Key Features of Python

1. Simple and Readable Syntax

- Python code is easy to write and understand.
- It resembles plain English, making it ideal for beginners.

2. Interpreted Language

- Python executes code line-by-line, which simplifies debugging and testing.

3. Dynamically Typed

- You don't need to declare variable types explicitly.

python

```
x = 10  # Python automatically knows x is an integer
```

4. High-Level Language

- Abstracts away complex details like memory management, making development faster and easier.

5. Object-Oriented Programming (OOP)

- Supports classes, inheritance, and encapsulation.
- Encourages modular and reusable code.

6. Extensive Standard Library

- It comes with built-in modules for file handling, regular expressions, networking, and more.

7. Cross-Platform Compatibility

- Python runs on Windows, macOS, Linux, and even mobile platforms.

8. Large Ecosystem of Third-Party Libraries

- Libraries like NumPy, Pandas, Matplotlib, TensorFlow, Flask, and Django extend Python's capabilities.
- Rich ecosystem for web development (Django, Flask), data science (NumPy, Pandas), machine learning (TensorFlow, PyTorch), and automation (Selenium, OpenCV).

9. Embeddable and Extensible

- You can embed Python in other languages like C/C++ or extend it with custom modules.

10. Automatic Memory Management

- Python uses garbage collection to manage memory efficiently.

11. Support for Multiple Programming Paradigms

- Procedural, object-oriented, and functional programming styles are all supported.
    - **Procedural (Recipe):** A recipe is a list of steps to follow in order to bake a cake. The recipe and ingredients are separate.
        - **OOP (Smart Robot Chef):** You create a "Chef" object that knows how to do everything itself, like a person. The chef has its own ingredients and knows how to use them. You just tell the chef to "bake cake".
        - **Functional (Assembly Line):** An assembly line of specialized machines. One machine preps ingredients, another mixes them, and so on. Each machine does one specific job perfectly and passes the result to the next machine, without changing anything along the way.

12. Strong Community Support

- Massive global community, with tons of tutorials, forums, and documentation.

❖How to Install Python

✅ Step 1: Download Python

- Go to the official website: https://www.python.org/downloads
- Click on the **Download Python 3.x.x** button (latest version recommended).

✅ Step 2: Install Python

On Windows:

- Run the downloaded .exe installer.
- **Important**: Check the box that says, **"Add Python to PATH"**.

  "C:\Users\admin\AppData\Local\Programs\Python\Python313"
- Click **Install Now**.

On macOS:

- Run the .pkg installer.
- Follow the installation steps.

On Linux:

- Use your package manager:

```bash
sudo apt update
sudo apt install python3
```

✅ Step 3: Verify Installation

- Open **Command Prompt** (Windows) or **Terminal** (macOS/Linux).
- Type:

bash

python --version

or

bash

python3 --version

- You should see the installed version number.

▶️ Running a Python Program

Method 1: Using the Python Shell (Interactive Mode)

- Open Terminal or Command Prompt.
- Type python or python3.
- You'll enter the interactive shell (>>>).
- Try:

print("Hello, World!")

Method 2: Using a Script File

1. Open a text editor (like Notepad, VS Code, or PyCharm).
2. Write your code:

print("Hello from a Python script!")

3. Save the file as hello.py.
4. Run it from the terminal:

python hello.py

or

python3 hello.py

Using IDLE (Python's Built-in Editor)

- After installation, search for **IDLE** in your system.
- Open it and write your code in the editor.
- Press **F5** to run the program.

❖ Disadvantages of Python

1. Slower Execution Speed

- Python is **interpreted**, not compiled, which makes it slower than languages like C++ or Java.

C, C++ vs Python: Machine Code Execution

✅ **C & C++**

- C code is **compiled** using a compiler (like GCC).
- The compiler translates the source code into **machine code** — binary instructions that the CPU can execute directly.
- This results in a standalone executable (.exe, .out, etc.) that runs **natively on the hardware**.
- 🔥 Fast, efficient, and minimal runtime overhead.

❌ **Python**

- Python code is **not compiled into machine code** by default.
- Instead, it's **interpreted**:
  - Python source code (.py) is first compiled into **bytecode** (.pyc).
  - This bytecode is executed by the **Python Virtual Machine (PVM)** — a software layer that interprets instructions.
- So Python runs **on top of a runtime**, not directly on the hardware.

Language	Compilation	Execution
C	Source → Machine Code	Runs directly on CPU
Python	Source → Bytecode → PVM	Runs inside interpreter

2. High Memory Consumption

- Python's dynamic typing and flexibility come at the cost of **higher memory usage**.
- This can be a limitation for mobile apps or embedded systems.

3. Weak in Mobile Development

- Python is rarely used for mobile app development.
- Frameworks like Kivy and BeeWare exist, but they're not as mature or popular as Swift (iOS) or Kotlin (Android).

4. Runtime Errors

- Because Python is dynamically typed, type-related bugs often appear **at runtime**, not during development.
- This can lead to unexpected crashes if not properly handled.

5. Limited Multithreading

- Python's **Global Interpreter Lock (GIL)** restricts true parallel execution of threads.
- This makes Python less efficient for CPU-bound multithreaded tasks.

6. Not Ideal for Enterprise-Grade Applications

- While Python is great for startups and prototypes, some large-scale enterprise systems prefer statically typed languages like Java or C# for better performance and maintainability.

Feature	Statically Typed	Dynamically Typed
Type known at	Compile time	Runtime
Type declaration	Required	Optional
Examples	C, Java, Rust	Python, JavaScript, Ruby
Flexibility	Less flexible	More flexible
Error detection	Early (compile time)	Late (runtime)

7. Design Limitations

- Python's simplicity can lead to **less strict coding practices**, which may affect large team projects without proper discipline.
- **No enforced type declarations** You don't have to specify variable types (e.g., Python, JavaScript).
- **Loose syntax rules** You can write code with fewer formal requirements (e.g., semicolons optional, indentation flexible).
- **Dynamic behavior** Variables can change type during execution, and functions can accept varying inputs.
- **Minimal error checking at compile time** Errors are often caught only when the program runs.

- **Flexible structure** You can write code without strict class hierarchies or rigid design patterns.

```
# Python (less strict)
x = 10
x = "hello"  # Allowed - x changes from int to str

# Java (strict)
int x = 10;
x = "hello";  // ❌ Compile-time error - type mismatch
```

8. Indentation in Python: A Feature and a Flaw

✅ Why It's a Feature

- Python uses indentation to define code blocks (like loops, functions, and conditionals), instead of braces {} or keywords.
- This enforces clean and readable code, which is great for collaboration and learning.

❌ Why It Can Be a Disadvantage

1. **Sensitive to Whitespace**
    - A single misplaced space or tab can break your code or cause unexpected behavior.
    - Example:

```
if True:
    print("Hello")
  print("World")  # This will raise an IndentationError
```

2. **Hard to Spot Errors**
    - Indentation errors are often visually subtle, especially in large codebases or when mixing tabs and spaces.
3. **Copy-Paste Issues**
    - Copying code from websites or documents can introduce invisible formatting problems.
4. **Inconsistent Editors**

- Different text editors may handle tabs and spaces differently, leading to confusion.

❖ Interactive Mode

- **What It Is:**
  - A live Python shell where you can type and execute code **line-by-line**.
  - Great for testing small code snippets, learning, or debugging.
- **How to Use:**
  - Open your terminal or command prompt.
  - Type python or python3 and press Enter.
  - You'll see the >>> prompt where you can start typing Python code.
- **Example:**

```
>>> print("Hello")
Hello
>>> 2 + 3
5
```

- **Pros:**
  - Instant feedback.
  - Ideal for quick experiments or learning.
- **Cons:**
  - Code isn't saved unless you copy it manually.
  - Not suitable for large programs.

❖ Script Mode

- **What It Is:**
  - You write Python code in a **.py file** and run the entire script at once.
  - Best for building full applications or saving reusable code.
- **How to Use:**

1. Open a text editor (like VS Code, Notepad, or PyCharm).
2. Write your code:

```
print("Hello from script mode!")
```

3. Save it as hello.py.
4. Run it from terminal:

```
python hello.py
```

- **Pros:**
  - Code is saved and reusable.
  - Easier to organize and debug large programs.
- **Cons:**
  - Requires saving and running files—less immediate than interactive mode.

Summary Table

Feature	**Interactive Mode**	**Script Mode**
Execution Style	Line-by-line	Entire file
Best For	Testing, learning	Full programs, projects
Code Persistence	Not saved	Saved in .py file
Speed	Instant feedback	Requires saving and running
Tools Needed	Terminal only	Text editor + terminal

Simple Example

you often need to **type cast** when you're working with input() because it always returns a string—even if the user types a number.

```
a = int(input("Enter first number: "))
b = int(input("Enter second number: "))
print("Sum is:", a + b)
```

Without it, Python would try to add two strings, which results in concatenation:

```
a = input("Enter first number: ")  # Let's say user enters "2"
b = input("Enter second number: ") # User enters "3"
print("Sum is:", a + b)           # Output: "Sum is: 23"
```

slightly improved version of the simple calculator that uses float() for decimal support and includes basic error handling:

```
try:
    a = float(input("Enter first number: "))
    b = float(input("Enter second number: "))
    print("Sum is:", a + b)
except ValueError:
    print("Invalid input! Please enter numeric values.")
```

The try-except block catches cases where someone enters text instead of a number.