**Introduction to Python – Detailed Notes**

**What is Python?**

**Python** is a **high-level**, **interpreted**, **general-purpose programming language** created by **Guido van Rossum** in **1991**.  
It emphasizes **code readability** and **simplicity**, allowing developers to express concepts in fewer lines of code than many other languages.

Python’s design philosophy focuses on:

* **Readability and simplicity** ("There should be one—and preferably only one—obvious way to do it.")
* **Ease of learning and use**
* **Extensibility and portability**

It is widely used in various fields such as:

* **Data Science & Machine Learning**
* **Web Development**
* **Automation & Scripting**
* **Artificial Intelligence**
* **Game Development**
* **Networking & IoT**
* **Cybersecurity & Cloud Computing**

**History of Python**

| **Year** | **Event** |
| --- | --- |
| 1989 | Guido van Rossum started developing Python at CWI (Netherlands). |
| 1991 | Python 0.9.0 released — included exception handling and functions. |
| 2000 | Python 2.0 released — introduced garbage collection and Unicode support. |
| 2008 | Python 3.0 released — major redesign for consistency and modernization. |
| Present | Python 3.x series continues with vast community and library support. |

**Note:** Python 2 officially ended support in **January 2020**. All new projects should use **Python 3**.

**Key Features of Python**

1. **Simple and Easy to Learn**
   * Syntax similar to English.
   * Ideal for beginners.
2. **Interpreted Language**
   * Code runs line by line; no need for compilation.
   * Makes debugging easier.
3. **High-Level Language**
   * Programmer doesn’t manage memory directly.
   * Focuses on problem-solving rather than system details.
4. **Portable and Cross-Platform**
   * Works on Windows, macOS, Linux, Android, etc.
   * "Write Once, Run Anywhere".
5. **Extensive Standard Library**
   * Built-in modules for file handling, math, networking, etc.
   * Reduces need to rewrite existing functionality.
6. **Object-Oriented Programming (OOP)**
   * Supports classes, inheritance, and encapsulation.
   * Also supports procedural and functional styles.
7. **Dynamically Typed**
   * No need to declare data types explicitly.
   * Type determined during runtime.
8. **Open Source and Free**
   * Available freely for modification and distribution.
9. **Integration Capability**
   * Integrates easily with C, C++, Java, .NET, and APIs.
10. **Huge Community Support**
    * One of the largest developer communities.
    * Abundant online documentation and forums.

**Why is Python So Popular?**

* Minimal code, maximum productivity.
* Extremely versatile — from data science to web apps.
* Supports major frameworks like **TensorFlow**, **Flask**, **Django**, **PyTorch**, **NumPy**, **Pandas**, etc.
* Ideal for **rapid prototyping** and **production-ready applications**.
* Backed by **tech giants** like Google, Meta, Microsoft, and OpenAI.

**How Computers Execute Python Code**

**Step-by-Step Execution Flow**

1. **You write Python code (.py file)**  
   Example:
2. print("Hello, World!")
3. **Python Interpreter Reads the Code**  
   Python is **interpreted**, not compiled into machine code directly.  
   The interpreter executes the code **line by line**.
4. **Conversion to Bytecode**
   * Python first compiles the source code to **bytecode** (low-level platform-independent representation).
   * This bytecode is stored in .pyc files inside the \_\_pycache\_\_ folder.
   * Example:
   * hello.py → hello.cpython-311.pyc
5. **Execution by Python Virtual Machine (PVM)**
   * The **Python Virtual Machine (PVM)** reads and executes the bytecode.
   * The PVM handles:
     + Memory management
     + Garbage collection
     + Exception handling
6. **Result Displayed to User**
   * Output is displayed on the console or passed to the next function.

**Example Visualization**

Source Code (.py)

↓

Compiler (CPython)

↓

Bytecode (.pyc)

↓

Python Virtual Machine (PVM)

↓

Output

**Python Implementations**

There are several implementations of the Python language:

**Advantages of Python**

| **Advantage** | **Description** |
| --- | --- |
| **Readable & Maintainable** | Clean syntax makes it easy to write and maintain. |
| **Extensive Libraries** | Huge ecosystem of packages (NumPy, Pandas, Matplotlib, etc.). |
| **Cross-Platform** | Works seamlessly on multiple operating systems. |
| **Productive Development** | Fewer lines of code → faster development. |
| **Integration Support** | Easily integrates with other languages (C/C++/Java). |
| **Community & Support** | Massive global community and resources. |
| **Ideal for Data Science** | Libraries for ML, AI, visualization, and analytics. |
| **Open Source** | Freely available and customizable. |

**Limitations of Python**

| **Limitation** | **Description** |
| --- | --- |
| **Slower Execution** | Being interpreted, it’s slower than compiled languages like C/C++. |
| **High Memory Usage** | Dynamic typing and large objects can consume more memory. |
| **Weak in Mobile Development** | Rarely used for Android/iOS apps. |
| **Runtime Errors** | Dynamic typing can lead to unexpected runtime bugs. |
| **Not Ideal for Low-Level Programming** | Unsuitable for system-level hardware control. |

**Python vs Other Languages**

| **Feature** | **Python** | **Java** | **C++** |
| --- | --- | --- | --- |
| **Typing** | Dynamic | Static | Static |
| **Execution** | Interpreted | Compiled + JVM | Compiled |
| **Syntax** | Simple, minimal | Verbose | Complex |
| **Speed** | Slower | Fast | Very Fast |
| **Use Case** | Data Science, AI, Web | Enterprise, Android | Systems, Games |
| **Memory Management** | Automatic | Automatic | Manual |

**Applications of Python**

1. **Data Science & Analytics** – Pandas, NumPy, Matplotlib, Scikit-learn
2. **Machine Learning & AI** – TensorFlow, PyTorch, OpenCV
3. **Web Development** – Django, Flask, FastAPI
4. **Automation/Scripting** – File automation, bots, data scraping
5. **Game Development** – Pygame, Panda3D
6. **Cybersecurity** – Network scanning, automation scripts
7. **Desktop Applications** – Tkinter, PyQt
8. **IoT & Embedded Systems** – MicroPython, Raspberry Pi projects