PYTHON PROGRAMMING

Topics covered:

- Real Time Applications developed by using Python Programming
- History of Python Language
- Versions of Python
- Features of Python
- Drawbacks of Python Programming Language

REAL TIME APPLICATIONS DEVELOPED BY USING PYTHON PROGRAMMING

By using python programming, we can develop the following applications:

1. Web Applications
2. Artificial Intelligence Applications (ML, DL)
3. Desktop Applications (GUI applications)
4. Text Processing Applications
5. Web Scraping Applications
6. Software Development
7. Automation
8. Business Application Development (Business Apps – Swiggy)
9. Embedded Applications
10. Scientific Applications (for solving complex mathematical problems)
11. Data Visualization and Analysis Apps

HISTORY OF PYTHON

- The idea of the Python programming language emerged in the 1980s
- Developed by Guido van Rossum, often referred to as the "Father of Python"
- Developed at CWI (Centrum Wiskunde & Informatica), Netherlands
- Python was first officially released in February 1991 (version 0.9.0)
- Managed and maintained by a non-commercial organisation called "Python Software Foundation (PSF)"
- Official website of PSF: www.python.org

Interesting Fact: The name "Python" was inspired by Guido's love for the British comedy series ***Monty Python's Flying Circus***, not the snake.

VERSIONS OF PYTHON

Python programming language contains 3 types of versions:

- Python 1.x (outdated)
  - First official version released as Python 0.9.0, and later evolved into Python 1.0 in January 1994
- Python 2.x (outdated)
  - Python 2.x does not support Backward Compatibility
  - Completely developed from scratch (different from 1.x)
- Python 3.x (current version)
  - Python 3.x does not support Backward Compatibility
  - Completely developed from scratch (different from 2.x)
  - 3 → Major Version and x → Minor Version

FEATURES OF PYTHON

Features of a programming language refer to the services or facilities provided by language developers and are used by language programmers for developing real-time applications.

Python programming language provides 11 core features. They are:

1. Simple
2. Freeware and Open Source
3. Platform Independent
4. Dynamically Typed
5. Interpreted
6. High Level
7. Robust
8. Both Functional and Object Oriented
9. Extensible
10. Embedded
11. Supports third party APIs (such as Numpy and Pandas)

SIMPLE: Python is a SIMPLE programming language because of three technical factors:

- Factor 1: Rich set of Modules (Libraries)
- Factor 2: Inbuilt Garbage Collection Facility (no need to write code for Garbage Collector)
- Factor 3: User-friendly syntaxes

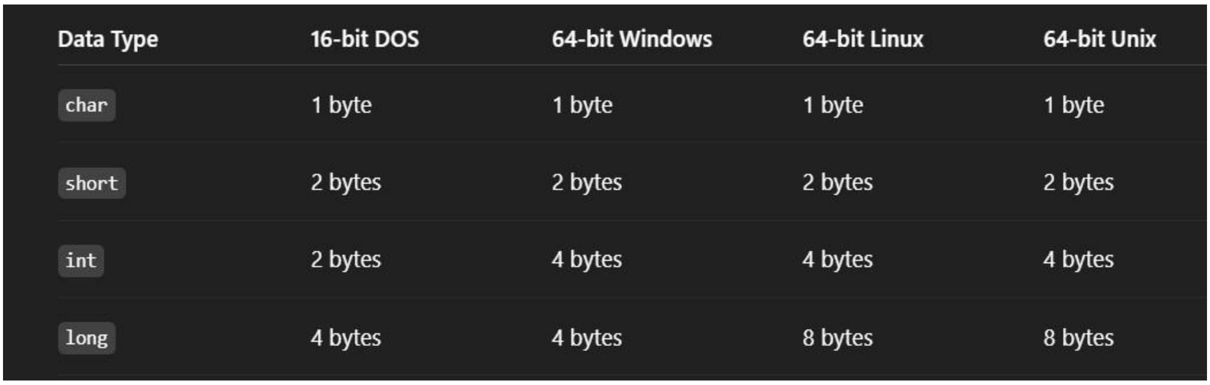
These three technical factors make Python easy to read, write, learn, and debug, contributing significantly to its popularity and widespread adoption.

FREEWARE AND OPEN SOURCE:

- Freeware: Python is free to download, install, and use.
- Open Source: Being Open Source means Python's source code is publicly available.
  - The default and most widely used implementation of Python is called "CPython"
  - The customized version of CPython are called **Python Distributions**
  - Example of Python Distributions: JPython or Jython – used for running Java based apps; Anaconda Python – used for running BigData and Hadoop based apps

PLATFORM INDEPENDENT LANGUAGE:

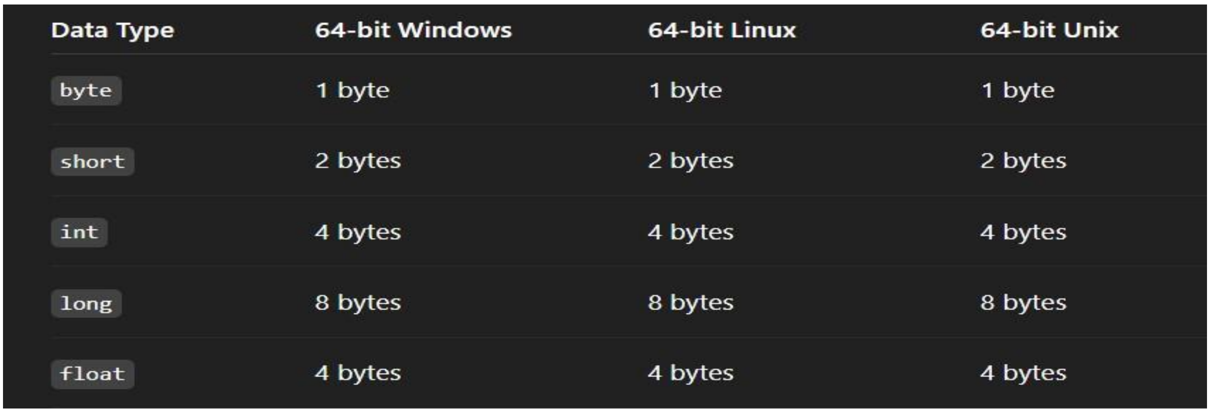
- A platform refers to the type of Operating System (OS) or hardware environment in which a program is compiled and executed. Example: Windows, Linux, etc.
- A platform independent language allows a program to be written once and executed on any operating system without modification.
- **Datatype (General)**: A data type is used to specify the type of data a variable can hold, and it helps the complier/interpreter allocate appropriate memory space and perform valid operations on that data.
- **Datatype (Python):** In Python, a data type is a classification that identifies the type of a value and determines what operations can be performed on that value.



Data Type	16-bit DOS	64-bit Windows	64-bit Linux	64-bit Unix
`char`	1 byte	1 byte	1 byte	1 byte
`short`	2 bytes	2 bytes	2 bytes	2 bytes
`int`	2 bytes	4 bytes	4 bytes	4 bytes
`long`	4 bytes	4 bytes	8 bytes	8 bytes

Platform dependency in C/C++

Conclusion: Since data types in C/C++ occupy different memory sizes on different operating systems and architectures, C/C++ is a Platform Dependent programming language.



Data Type	64-bit Windows	64-bit Linux	64-bit Unix
`byte`	1 byte	1 byte	1 byte
`short`	2 bytes	2 bytes	2 bytes
`int`	4 bytes	4 bytes	4 bytes
`long`	8 bytes	8 bytes	8 bytes
`float`	4 bytes	4 bytes	4 bytes

Platform independency in Java

Conclusion: In the Java programming language, data types always occupy the same memory size on all platforms. Thus, Java is a Platform Independent programming language.

Question: How Platform Independency is achieved?
Answer: Platform Independency is possible when the language abstracts away OS-specific details using an interpreter or virtual machine.

How Python Achieves Platform Independence?

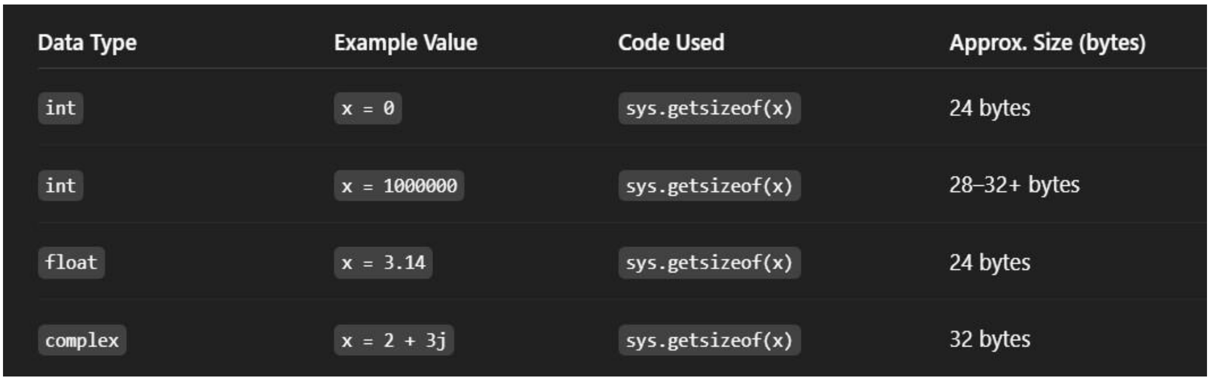
Python code is not compiled to machine code like C/C++ but instead:

1. Python source code (.*py*) is converted into bytecode (.*pyc*)
2. That bytecode is then interpreted by the Python Virtual Machine (PVM)
3. The **PVM (part of the Python interpreter)** is platform-specific but behaves consistently across systems
4. As long as the Python interpreter is available on a platform, you can run the same .*py* file on any OS. The exact same file can run unchanged on any system where Python is installed, **because the interpreter handles OS-level differences**.

Understanding Datatypes in Python:

Python is dynamically typed, and all data types in Python are actually objects, not primitive types like in C/C++. Because of that:

- Their sizes are not fixed.
- The size depends on the value stored in the variable.
- You can find the size in bytes using the *sys.getsizeof()* function from the *sys* module.



Data Type	Example Value	Code Used	Approx. Size (bytes)
`int`	`x = 0`	`sys.getsizeof(x)`	24 bytes
`int`	`x = 1000000`	`sys.getsizeof(x)`	28–32+ bytes
`float`	`x = 3.14`	`sys.getsizeof(x)`	24 bytes
`complex`	`x = 2 + 3j`	`sys.getsizeof(x)`	32 bytes

Python Datatypes

Conclusion:

1) C/C++ is platform dependent, even though it uses object code, because the final executable (.exe, .out) is tied to the OS and hardware. (C/C++ generates machine code)
2) Python is platform independent because Python code (.py files) is first compiled into bytecode (.pyc files) — which is OS-independent (Python only generates Bytecode). This bytecode is then executed by the PVM on any system — as long as the PVM for that platform is installed.

Question: Is PVM platform independent?
Answer: No.
Reason: When you install Python on any operating system (Windows, Linux, macOS), you are installing:

1) The Python Interpreter
2) The Python Standard Library
3) The PVM (Python Virtual Machine)

This PVM is compiled for the specific operating system and architecture. For example: On Windows, the PVM is compiled for Windows and On Linux, it's compiled for Linux.

Question: Why is Python Platform Independent? Explain.
Answer: Python is considered a platform-independent language because the same Python code can run on different operating systems (like Windows, Linux, or macOS) without modification.
When a Python script is run, it is first compiled into bytecode (.pyc files), which is a low-level, platform-independent representation of the code. This bytecode is then executed by the Python Virtual Machine (PVM), which is a part of the Python interpreter.
The PVM is platform-specific, but the Python bytecode is not. As long as a compatible interpreter is available on the system, the same .py or .pyc file will run seamlessly.

Unlike languages like C or C++, which compile directly to machine code and produce platform-specific binaries, Python relies on an interpreter layer that abstracts away the underlying operating system.

DYNAMICALLY TYPED PROGRAMMING LANGUAGE:

There are 2 types of programming languages: (Based on Typing)

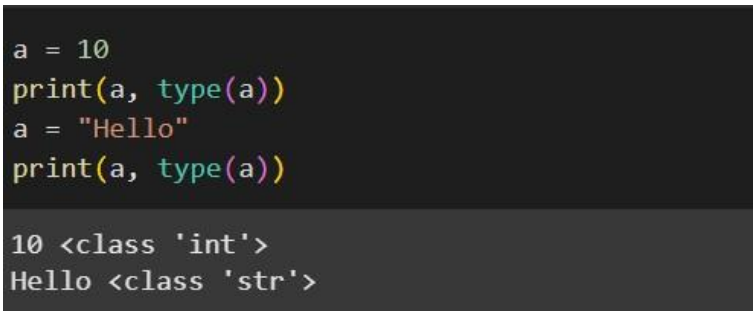
1) Statically Typed Programming Language
2) Dynamically Typed Programming Language

Statically Typed Programming Language:

- In Statically typed languages, the datatype of the variable must be declared explicitly before using it. Examples: C, C++, Java
- int a = 11 (int a → **Variable Declaration**)
- In statically typed programming languages, **datatype of the variable is known at the compile time**
- Memory is allocated based on the declared type

Dynamically Typed Programming Language:

- In dynamically typed languages, there is no need to explicitly declare the datatype of a variable. Examples: Python, Javascript
- In dynamically typed programming language, depending on the value of the variable, datatype is automatically/implicitly assigned by python interpreter.
- **The interpreter determines the datatype at runtime**, based on the value assigned to the variable.
- This means the type of a variable can change during the program's execution, depending on the data it holds.



```
a = 10
print(a, type(a))
a = "Hello"
print(a, type(a))
```

```
10 <class 'int'>
Hello <class 'str'>
```

Dynamic Typing and Dynamic Binding

INTERPRETED PROGRAMMING LANGUAGE:

- INTERPRETATION:
  - The line-by-line conversion of source code into machine code (or intermediate form) at runtime is called Interpretation
  - Each line is translated and executed immediately

  - Execution stops if a runtime error is encountered
- INTERPRETER: An interpreter is a program that performs the interpretation process — it reads the source code one line at a time, translates it, and executes it instantly.
- COMPILATION:
  - The process of converting all lines of source code into machine code (binary/executable) at once, before execution, is called compilation.
  - The entire program is converted before running
  - Errors are caught during the compile phase, before execution
- COMPILER: A compiler is a program that performs the compilation process, converting high-level source code into low-level object code or executables.

Question: Why is Python an Interpretation-Based Programming Language?
Answer: Python is considered an interpreted language because its source code is compiled to bytecode and then executed line-by-line by the Python Virtual Machine, rather than being converted into a platform-specific executable file.

Whenever we execute a Python program, two phases take place:

1) Compilation Phase: Python source code is first compiled into bytecode (*.pyc* files). However, this compilation is not line-by-line — it's done as a whole script and converts the entire code into an intermediate, portable format.

2) Execution Phase: This bytecode is then sent to the Python Virtual Machine (PVM), which executes it line-by-line.

Since Python uses an interpreter (PVM) to execute the bytecode line-by-line at runtime, it is classified as an interpretation-based programming language.

Question: What is PVM?
Answer: PVM is a part of Python interpreter, and its role is to read and execute Python bytecode line-by-line, converting it into machine-level instructions that the underlying operating system and hardware can understand.

Question: What is a Program?
Answer: A program is a well-defined set of instructions written in a programming language to perform a specific task or solve a particular problem.

HIGH-LEVEL PROGRAMMING LANGUAGES:

There are 2 types of programming languages: (Based on Abstraction Level)

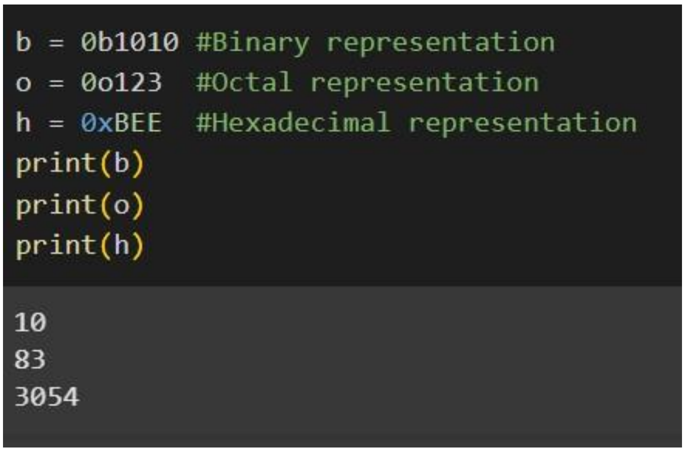
1) Low-Level Programming Language
2) High-Level Programming Language

Low-Level Programming Language:

- Low-level programming languages are closer to hardware and provide little or no abstraction from a computer's instruction set architecture.
- Instructions are written in binary, octal, or hexadecimal formats.
- These languages are machine-dependent.
- Difficult to write, read, and debug for humans.
- Examples: **Machine Language and Assembly Language**

High-Level Programming Language:

- High-level programming languages are closer to human languages and provide a high level of abstraction from hardware.
- Data is usually written and manipulated in decimal format, or readable formats like strings, lists, and objects.
- Easier to learn, write, debug, and maintain
- Programs are portable across platforms (platform-independent).
- Examples: **C++, Java, Python**
- Even when the programmer writes in binary, octal, or hexadecimal, the language runtime handles the conversion and manages memory.



```
b = 0b1010 #Binary representation
o = 0o123  #Octal representation
h = 0xBEE  #Hexadecimal representation
print(b)
print(o)
print(h)
```

```
10
83
3054
```

ROBUST:

- Robustness refers to the ability of a program to handle errors gracefully without crashing, and to provide meaningful, user-friendly error messages instead of technical failures.
- Python is one of the robust programming languages because it provides powerful Exception Handling mechanism.
- Exception Handling: Exception Handling is the process of detecting and managing runtime errors in a program, and converting technical error messages into clear, user-friendly messages.

Question: What are the different types of Errors in Python?

Answer: Python errors are mainly categorized into 3 types:

1) **Syntax Errors**:

- Occurs when the code violates Python's syntax rules
- Detected at compile time (before execution)

2) **Runtime Errors (Exceptions)**:

- These occur during program execution when python encounters something it cannot handle
- The program compiles correctly but crashes while running unless the error is handled. Example: ZeroDivisionError
- These are also called **Exceptions**

3) **Logical Errors**:

- These occur when the program runs without crashing, but the output is incorrect due to a mistake in the logic
- Detected only by carefully checking the output

BOTH FUNCTIONAL AND OBJECT-ORIENTED PROGRAMMING LANGUAGE:

Python supports multiple programming paradigms, including:

- **Functional Programming** — focuses on using functions to build logic
- **Object-Oriented Programming (OOP)** — organizes code into classes and objects

EXTENSIBLE:

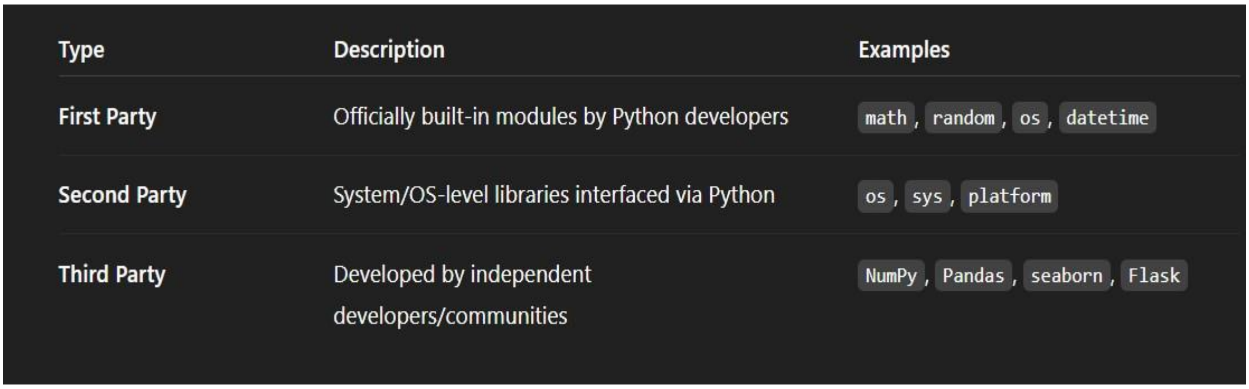
- A programming language is said to be Extensible if it allows its features or functionality to be used or extended by code written in other languages (like C, C++, or Java).
- Python provides its programming facilities to C, C++, Java and hence, python is one of the Extensible programming languages
- In data science, **NumPy and Pandas use C/C++ extensions under the hood for speed, while exposing a Python interface**

EMBEDDED:

- A programming language is said to be Embedded if it can integrate and use the features of other programming languages.
- Python is an embedded language because it embeds the services of C (and sometimes C++) to improve performance and speed
- The CPython interpreter itself is written in C — so Python relies on C at its core.

SUPPORTS THIRD PARTY APIs:

- A third-party API (or library/module) refers to **any software package developed outside the core Python development team**, which can be installed and imported into Python programs to add new functionality.
- Numpy (Numerical Python):
  - Used for solving complex mathematical problems
  - Developed by Travis Oliphant
- Pandas (Panel Data):
  - Used for Data Analysis and Analytics
  - Developed by Wes McKinney
- Matplotlib: used for Data Visualization



Type	Description	Examples
First Party	Officially built-in modules by Python developers	`math`, `random`, `os`, `datetime`
Second Party	System/OS-level libraries interfaced via Python	`os`, `sys`, `platform`
Third Party	Developed by independent developers/communities	`NumPy`, `Pandas`, `seaborn`, `Flask`

Categorization of Modules/Libraries

DRAWBACKS OF PYTHON PROGRAMMING LANGUAGE

- **Slower Execution Speed**: Python is interpreted and dynamically typed, which makes it slower than compiled languages like C, C++, or Java.
- **High Memory Consumption**: Python uses a lot of memory because It stores everything as objects
- **Runtime Errors**: Because Python is dynamically typed, many bugs appear only at runtime, not at compile time.