# Python - *Fundamentals Of Python Language*

# Module 1: Introduction to Python

## 1. Introduction to Python and its Features

Python is a simple, high-level, interpreted programming language known for its readability and wide application in web development, automation, data science, etc.

## 2. History and Evolution of Python

Python was created by Guido van Rossum and released in 1991. It has evolved over time with major releases like Python 2 and Python 3, focusing on code readability and simplicity.

## 3. Advantages of Python Over Other Languages

Python has easy syntax, vast libraries, and community support. It is dynamically typed and supports multiple programming paradigms.

## 4. Installing Python and Setting Up Environment

Python can be installed from python.org. IDEs like Anaconda, PyCharm, or VS Code are used for development.

## 5. Writing and Executing Your First Python Program

Using print('Hello, World!'), you can write and run your first program either in IDLE, terminal, or an IDE.

# Module 2: Programming Style

## 1. PEP 8 Guidelines

PEP 8 is the style guide for writing clean and readable Python code, focusing on naming, indentation, and formatting.

## 2. Indentation, Comments, and Naming Conventions

Python uses indentation to define blocks. Comments use '#'. Variable names should be lowercase\_with\_underscores.

## 3. Writing Readable and Maintainable Code

Follow PEP 8, write modular code using functions, and use clear variable names for readability.

# Module 3: Core Python Concepts

## 1. Python Data Types

Python supports int, float, string, list, tuple, dictionary, and set as common data types.

## 2. Python Variables and Memory Allocation

Variables are created dynamically in Python and memory is managed using reference counting and garbage collection.

## 3. Python Operators

Python includes arithmetic (+, -, \*, /), comparison (==, !=), logical (and, or, not), and bitwise operators (&, |, ^).

# Module 4: Conditional Statements

## 1. Conditional Statements (if, elif, else)

Used to make decisions in code. if checks a condition, elif adds more, and else is the fallback.

## 2. Nested if-else Conditions

An if-else block inside another if-else is called nested, used for multiple level decision-making.

# Module 5: Looping (For, While)

## 1. For and While Loops

For is used to iterate over sequences. While runs while a condition is true.

## 2. Using Loops with Collections

Loops can be used with lists, tuples, and dictionaries to access or modify data.

# Module 6: Generators and Iterators

## 1. Generators in Python

Generators return values one at a time using yield. They are memory-efficient.

## 2. Yield vs Return

Return exits the function and returns a value. Yield pauses the function saving its state.

## 3. Iterators in Python

An iterator has \_\_iter\_\_() and \_\_next\_\_() methods to traverse through elements.