DAY 01

Python is a high-level, versatile, interpreted, Object-Oriented, Dynamic Typing programming language known for its simplicity and readability. Guido van Rossum created Python in the late 1980s and released its first version in 1991. Since then, it has gained widespread popularity and is widely used in various domains such as web development, data analysis, artificial intelligence, scientific computing, and more.

INTERPRETED

Python is an interpreted language, meaning that code is executed line by line, making debugging easier.
Instead of translating the entire source code into machine code before execution (as in compiled languages like C++ or Java), the Python interpreter reads and executes each line of code sequentially, from top to bottom.

Since Python is interpreted, you can see the results of your code immediately after running it. This makes debugging easier, as you can identify and fix errors as you go along without needing to wait for a separate compilation step.

OBJECT-ORIENTED

Python supports object-oriented programming (OOP) principles, allowing for the creation of reusable and modular code through classes and objects.

OOP provides a way to organize and manage code by encapsulating data (attributes) and behaviors (methods) into objects.

DYNAMIC TYPING

Python uses dynamic typing, which means you don't need to declare variable types explicitly. Variables can dynamically change types during runtime, enhancing flexibility and reducing the amount of code needed.

PRINT FUNCTION

In Python, the print( ) function is used to display output to the console or terminal. It takes zero or more arguments and prints them to the standard output device (usually the console). The print() function is a built-in function, meaning you don't need to import any modules to use it.

STRING

In Python, a string is a sequence of characters enclosed within either single quotes (') or double quotes ("). Strings are one of the fundamental data types in Python and are used to represent text data.

VARIABLES

In Python, a variable is a named reference to a value stored in the computer's memory. It serves as a way to label and store data so that it can be referenced and manipulated within a program. Variables in Python are dynamically typed

RULES AND GUIDELINES

- Variable names can contain only letters, numbers, and underscores. They can start with a letter or an underscore, but not with a number. For instance, you can call a variable message_1 but not 1_message.
- Spaces are not allowed in variable names, but underscores can be used to separate words in variable names. For example, greeting_message works, but greeting message will cause errors.

- Avoid using Python keywords and function names as variable names; that is, do not use words that Python has reserved for a particular programmatic purpose, such as the word print.
- Variable names should be short but descriptive. For example, name is better than n, student_name is better than s_n, and name_length is better than length_of_persons_name.
- Be careful when using the lowercase letter l and the uppercase letter O because they could be confused with the numbers 1 and 0

DATA TYPES

In Python, data types represent the type of data that can be stored and manipulated within a program. Python is a dynamically typed language, meaning you don't need to declare the data type of a variable explicitly. Instead, the interpreter automatically assigns the appropriate data type based on the value assigned to the variable. Python provides several built-in data types, including:

NUMERIC TYPES:

- int: Integer values represent whole numbers without any decimal point.

- float: Floating-point values represent decimal numbers.
- complex: Complex numbers have a real and imaginary part, denoted as a + bj, where a is the real part and b is the imaginary part.

SEQUENCE TYPES:

- **str:** Strings represent a sequence of characters enclosed within single, double, or triple quotes.
- **list:** Lists are ordered collections of items, which can be of different data types, enclosed within square brackets **[ ]**.
- **tuple:** Tuples are similar to lists but are immutable, meaning their elements cannot be modified once defined, enclosed within parentheses **( )**.

MAPPING TYPE:

- **dict:** Dictionaries are unordered collections of key-value pairs, where each key is associated with a value, enclosed within curly braces **{ }**.

SET TYPES:

- set: Sets are unordered collections of unique elements, enclosed within curly braces { }. Duplicates are automatically removed.

BOOLEAN TYPE:

- **bool:** Boolean values represent truth values, **True** or **False**, used in logical operations and conditions.

NONE TYPE:

- **None:** The None type represents the absence of a value or a null value. It is often used to indicate that a variable has no value assigned to it.

OPERATORS:

In Python, operators are special symbols or keywords that perform operations on operands, such as variables, values, or expressions. Python supports various types of operators, including arithmetic, comparison, logical, assignment, bitwise, identity, and membership operators. Here's an explanation of each type:

ARITHMETIC OPERATORS:

- + (Addition): Adds two operands.
- - (Subtraction): Subtracts the second operand from the first.
- * (Multiplication): Multiplies two operands.
- / (Division): Divides the first operand by the second (result is a float).

- // (Floor Division): Divides the first operand by the second and returns the integer quotient.
- % (Modulus): Returns the remainder of the division of the first operand by the second.
- ** (Exponentiation): Raises the first operand to the power of the second.

COMPARISON OPERATORS:

- == (Equal to): Checks if two operands are equal.
- != (Not equal to): Checks if two operands are not equal.
- > (Greater than): Checks if the first operand is greater than the second.
- < (Less than): Checks if the first operand is less than the second.
- >= (Greater than or equal to): Checks if the first operand is greater than or equal to the second.
- <= (Less than or equal to): Checks if the first operand is less than or equal to the second.

LOGICAL OPERATORS:

- and: Returns True if both operands are True.
- or: Returns True if at least one of the operands is True.
- not: Returns True if the operand is False, and False if the operand is True.

ASSIGNMENT OPERATORS:

- = (Assignment): Assigns the value of the right operand to the left operand.
- +=, -=, *=, /=, //=, etc.: Performs arithmetic operation and assigns the result to the left operand.

BITWISE OPERATORS:

- & (Bitwise AND)
- | (Bitwise OR)
- ^ (Bitwise XOR)
- ~ (Bitwise NOT)
- << (Left shift)
- >> (Right shift)

IDENTITY OPERATORS:

- is: Returns True if both operands refer to the same object.
- is not: Returns True if both operands do not refer to the same object.

MEMBERSHIP OPERATORS:

- in: Returns True if the first operand is present in the second operand (e.g., element in sequence).
- not in: Returns True if the first operand is not present in the second operand.