**Python**

**Introduction to python:**

**What is python?**

* Python is a popular programming language. It was created by Guido van Rossum, and released in 1991.
* Python is a high-level, interpreted programming language known for its simplicity and readability.
* It is widely used for various applications, including web development, data analysis, artificial intelligence, machine learning, automation, and more.
* Python emphasizes code readability with its clean syntax and indentation-based structure.

**Why python?**

* Python works on different platforms such as windows, mac, linux, etc.
* Python has a simple syntax similar to the English language.
* Python has syntax that allows developers to write programs with fewer lines than some other programming languages.
* Python runs on an interpreter system, meaning that code can be executed as soon as it is written. This means that prototyping can be very quick.

**What can python do?**

* Python can be used on server to create web applications.
* Python can be used alongside software to create workflows.
* Python can connect to database systems. It can also read and modify files.
* Python can be used to handle big data and perform complex mathematics.

**Variables:**

* In python, variables are created when you assign a value to it.
* Variables are containers for storing data values.
* Python has no command for declaring a variable.
* Variables do not need to be declared with any particular type, and can even change type after they have been set.
* Ex: x = 5  
   y = "John"  
   print(x) o/p: 5  
   print(y) o/p: John

**Variable Naming Conventions:**

* Naming conventions in python refer to rules and guidelines for naming variables, functions, classes and other entities in our code.
* A variable name must start with a letter or the underscore character.
* A variable name cannot start with a number.
* A variable name can only contain alpha-numeric characters and underscores (A-z, 0-9, and \_).
* A variable name cannot be any of the python keywords.
* Python is case-sensitive (name and NAME are different variables).
* EX: user\_name = “Ben”

Total\_score = 95

**Assigning multiple variables:**

* Python allows you to assign values to multiple variables in one line.
* Make sure the number of variables matches the number of values, or else you will get an error.
* Ex: x, y, z = “orange”, “banana”, “cherry”

print(x) #orange

print(y) #banana

print(z) #cherry

**One value to multiple variables:**

* We can also assign the same value to multiple variables in one line.
* Ex: x = y = z = "Orange"  
   print(x) #Orange  
   print(y) #Orange  
   print(z) #Orange

**Local Variables:**

* A local variable is a variable declared inside a function and can only be accessed within that function.
* Ex: def greet():

message = "Hello, World!" # Local variable

print(message)

greet()

print(message)

**Global Variables:**

* A global variable is a variable declared outside any function and can be accessed anywhere in the program.
* Global variables can be used by everyone, both inside of functions and outside.
* Ex: global\_message = "Hello, Python!" # Global variable

def greet():

print(global\_message) # Accessible inside the function

greet()

print(global\_message) # Accessible outside the function too

**Casting:**

* If you want to specify the data type of a variable, this can be done with casting
* Ex: x = str (3) # x will be ‘3’.

y = int (3) # y will be 3

**Data Types:**

* Python has several built-in data types that allow us to store and manipulate different kinds of values.
* Variables can store data of different types, and different types can do different things.
* Python has various data types they are:

1. Numeric Types: int, float, complex
2. Text Type: str
3. Sequences types: list, tuple, range
4. Mapping type: dict
5. Set types: set, frozenset
6. Boolean type: bool
7. Binary types: bytes, bytearray, memoryview
8. None type: NoneType

**Numeric Types:**

* Used for storing numerical values.
* Example: a = 100 # int

b = 3.14 # float

c = 2 + 5j # complex

print(type(a)) # <class 'int'>

print(type(b)) # <class 'float'>

print(type(c)) # <class 'complex'>

**Text Type:**

* In Python, strings (str) are used to store and manipulate text data. A string is a sequence of characters enclosed in single ('), double ("), or triple quotes (''' or """ """`).
* Ex: x=”hello”

**Sequence type:**

* Used to store collections of items.
* Ex: name = "Python" # String

numbers = [1, 2, 3] # List

coordinates = (10, 20) # Tuple

r = range (5) # Range

print(type(name)) # <class 'str'>

print(type(numbers)) # <class 'list'>

print(type(coordinates)) # <class 'tuple'>

print(list(r)) # Converts range to list: [0, 1, 2, 3, 4]

**Set types:**

* Used for storing unique, unordered collections of items.
* Ex: s = {1, 2, 3, 4, 4} # Set (duplicates are removed)

fs = frozenset ([10, 20, 30]) # Frozenset (immutable)

print(s) # {1, 2, 3, 4}

print(type(fs)) # <class 'frozenset'>

**Mapping Type:**

* Used for key-value pairs.
* Ex: person = {"name": "Alice", "age": 25, "city": "New York"}

print(person["name"]) # Alice

print(type(person)) # <class 'dict'>

**Boolean Type:**

* Represents True or False values.
* Ex: x = True

y = False

print (type(x)) # <class 'bool'>

print (10 > 5) # True

**Binary Types:**

* Used for handling binary data.
* Ex: b = bytes ("Hello", "utf-8")

ba = bytearray ([65, 66, 67]) # ASCII values

mv = memoryview(b)

print(b) # b'Hello'

print(ba) # bytearray(b'ABC')

print(mv) # <memory at 0x...>

**None Type:**

* Represents the absence of a value.
* Ex: x = None

print(type(x)) # <class 'NoneType'>

**Operations in Python:**

Python provides various operations that can be performed on different data types, including arithmetic, comparison, logical, bitwise, assignment, identity, and membership operations.

1. **Arithmetic operators**

Used for mathematical calculations.

* +(Addition): a + b
* -(Subtraction): a – b
* \*(Multiplication): a \* b
* /(Division): a / b
* //(Floor Division): a // b
* %(Modulus): a % b

1. **Comparison Operators**

Used to compare two values:

* == (Equal to): a == b
* != (Not equal to): a != b
* > (Greater than): a > b
* < (Less than): a < b
* >= (Greater than or equal to): a >= b
* <= (Less than or equal to): a <= b

1. **Logical Operators**

Used to perform logical operations:

* and: Returns True if both conditions are true.
* or: Returns True if at least one condition is true.
* not: Negates the truth value.

1. **Bitwise Operators**

Operate on binary numbers:

* & (AND): a & b
* | (OR): a | b
* ^ (XOR): a ^ b
* ~ (NOT): ~a
* << (Left Shift): a << b
* >> (Right Shift): a >> b

1. **Assignment Operators**

Used to assign values to variables:

* =: a = b
* +=: a += b
* -=: a -= b
* \*=: a \*= b
* /=: a /= b
* //=: a //= b
* %=: a %= b
* \*=: a \*= b

1. **Identity Operators**

Check if two objects are the same:

* is: a is b
* is not: a is not b

1. **Membership Operators**

Check for membership in sequences:

* in: a in b
* not in: a not in b

**Conditional Statements in Python**

Conditional statements allow programs to make decisions and execute different code blocks based on conditions.

1. **if statement:**

* The if statement executes a block of code only if the condition is True.
* Ex: age = 18

if age >= 18:

print ("You are eligible to vote.") # This runs if the condition is True

1. **if-else statement:**

* The else block executes when the if condition is False.
* Ex: age = 16

if age >= 18:

print ("You are eligible to vote.")

else:

print ("You are not eligible to vote.") # Executes if age < 18

1. **if-elif-else statement:**

* Used when multiple conditions need to be checked.
* Ex: score = 85

if score >= 90:

print ("Grade: A")

elif score >= 80:

print ("Grade: B")

elif score >= 70:

print ("Grade: C")

else:

print ("Grade: F")

1. **Nested if statements:**

* An if statement inside another if statement.
* Ex: age = 20

citizen = True

if age >= 18:

if citizen:

print ("You can vote.")

else:

print ("Only citizens can vote.")

else:

print ("You are too young to vote.")

**Loops in python:**

Loops in Python are used to execute a block of code multiple times. Python supports **two** types of loops:

1. for loop (Iterates over a sequence)
2. while loop (Runs while a condition is True)

**for loop:**

A for loop iterates over an iterable (list, tuple, string, range, etc.).

Ex: fruits = ["apple", "banana", "cherry"]

for fruit in fruits:

print(fruit)

output:

apple

banana

cherry

**while loop:**

A while loop runs as long as a condition is True.

Ex: x = 1

while x <= 5:

print(x)

x += 1

output:

1

2

3

4

5

**break statement:**

The break statement exits the loop immediately when the condition is met.

Ex:

for num in range(10):

if num == 5:

break # Stops the loop at 5

print(num)

output:

0

1

2

3

4

**Continue statement**:

The continue statement skips the rest of the current iteration and moves to the next one.

Ex:

for num in range(5):

if num == 2:

continue # Skips when num is 2

print(num)

output:

0

1

3

4

**Set in python:**

A set is an unordered collection of unique elements. It does not allow duplicates and supports mathematical operations like union, intersection, and difference.

Example:

my\_set = {1, 2, 3,3, 4, 5}

print(my\_set) # {1, 2, 3, 4, 5}

**Set properties:**

* **Unordered**: The elements don’t have a fixed order.
* **Unique**: Duplicate values are automatically removed.

**Dictionary in Python:**

A dictionary is an unordered collection of key-value pairs. Each key must be unique and immutable (strings, numbers, or tuples), but values can be of any type.

**Dictionary Properties;**

* Keys are unique.
* Values can be duplicated.

Example:

student = {

"name": "Alice",

"age": 20,

"grade": "A"

}

print(student["name"]) # Output: Alice

**12-02-2025**

**Interpreter in Python:**

The python interpreter reads and executes python code line by line. This makes python an interpreted language, unlike compiled languages like C or Java.

**Compiler:**

A compiler is a program that translates source code written in a high-level programming language (like C, C++, or Java) into machine code (binary code) that a computer's processor can execute directly.

**How a compiler works:**

1. **Lexical Analysis** – Breaks code into tokens.
2. **Syntax Analysis (Parsing)** – Checks grammar and builds a syntax tree.
3. **Semantic Analysis** – Ensures logical correctness.
4. **Optimization** – Improves performance and efficiency.
5. **Code Generation** – Converts code into machine language (binary).
6. **Linking** – Combines multiple code files into an executable program.

**High-level language:**

* High-level languages are programming languages that are used for writing programs or software that can be understood by humans and computers.
* High-level languages are easier to understand for humans because they use a lot of symbols letters phrases to represent logic and instructions in a program.
* Python is a high-level language, it means that Python is designed to be easier for humans toread, write, and understand, rather than being focused on how the computer hardware works.

**Low-level language:**

* A low-level language is a programming language that is closer to machine code and interacts directly with the computer's hardware. It requires knowledge of the computer’s architecture and memory management.
* Uses assembler to translate the instructions to machine language.

**Syntax:**

Syntax refers to the set of rules that define how python programs are written and interpreted. Python has a simple and easy-to-read syntax, making it beginner-friendly.

**Indentation in python**

Indentation in Python refers to the spaces or tabs at the beginning of a line to define the structure of the code. Unlike other languages that use {} (curly braces) to indicate code blocks, Python requires indentation to organize code properly.

**Why is Indentation Important?**

* Defines code blocks (functions, loops, conditions).
* Improves readability and structure.
* Mandatory in Python (incorrect indentation causes errors).

**Comments in python:**

Comments in Python are used to explain code and make it more readable. Comments are ignored by the Python interpreter and do not affect program execution.

**Types of Comments in Python:**

1. Single-Line Comments (#):

A single-line comment starts with a # symbol.

Example:

# This is a comment

print ("Hello, World!") # This prints a message

1. Multi-Line Comments **(""" """ or ''' '''):**

Python does not have a specific syntax for multi-line comments, but you can use triple quotes (""" or ''') to create multi-line documentation-style comments.

Example:

"""

This is a multi-line comment.

It can span multiple lines.

"""

print ("Python supports multi-line comments!")

**Operations:**

Operators are used to perform operations on variables and values.

Python divides the operators in the following groups:

* Arithmetic operators
* Assignment operators
* Comparison operators
* Logical operators
* Identity operators
* Membership operators
* Bitwise operators

1. Arithmetic Operations:

Arithmetic operators are used with numeric values to perform common mathematical operations such as addition, subtraction, multiplication and division, modulus, exponential, floor division.

Addition (+):

Adding of two numbers

Ex:

a=20

b=15

add = a+b

print ("addition of two numbers is:", add)

output: addition of two numbers is: 35

Subtraction (-):

Difference of two numbers

Ex:

a=20

b=15

sub=a-b

print ("subtraction of two numbers is:", sub)

output: subtraction of two numbers is: 5

Multiplication (\*):

Product of two numbers

Ex:

a=20

b=15

mult=a\*b

print ("multiplication of two numbers is:", mult)

output: multiplication of two numbers is: 300

Division (/):

Dividing the two numbers

Ex:

a=20

b=15

div=a/b

print ("division of two numbers is:", div)

output: division of two numbers is: 1.3333333333333333

Modulus (%):

It returns the remainder.

Ex:

a=20

b=5

mod=a%b

print ("modulus of two numbers is:", mod)

output: modulus of two numbers is: 0

Exponential:

Raises a number to the power of another.

Ex:

a=2

b=3

expon=a\*\*b

print ("exponential of two numbers is:", expon)

output: exponential of two numbers is: 8

Floor Division:

Returns the quotient without the decimal part.

Ex:

a=20

b=15

fdiv=a//b

print ("floor division of two numbers is:", fdiv)

output: floor division of two numbers is: 1