**Python**

**Python Introduction:**

🡪Python is a popular programming language. It was created by Guido Van Rossum in the year 1991.

🡪Python is a high-level, interpreted, and general-purpose programming language.

🡪**High-level language:** Uses either compiler or interpreter to translate the code

🡪**Low-level language:** Uses assembler to translate the instructions to machine language.

🡪**Interpreter:** Line by line execution of the program.

**Example:** Python is Interpreted language.

🡪**Compiler:** The whole is executed at once.

**Example:** Java is Compiler language.

🡪Python emphasizes code readability and simplicity, which makes it an excellent language for beginners and experienced developers.

🡪It is used for:

* Web development
* Software development
* AI & ML
* Mathematics,
* System scripting

**What can Python do?**

**🡪**Python can be used to create web applications.

🡪Python can connect to databases. It can also read and modify the files.

🡪Python can be used to handle big data and perform complex mathematics.

**Why Python ?**

🡪Python works on different platforms like Windows, Linux and Mac.

🡪Python has simple syntax. It is easy to understand.

🡪Python has beginner friendly libraries like random, os, re etc.

🡪Python error messages are easy to understand and debug.

**Key features in Python:**

* **Easy to Learn and Use:**Python’s simple and readable syntax makes it beginner-friendly.
* **Cross-Platform Compatibility:**Python runs seamlessly on Windows, macOS, and Linux.
* **Extensive Libraries:** Includes robust libraries for tasks like web development, data analysis, and machine learning.
* **Dynamic Typing:**Variable types are determined automatically at runtime, simplifying code writing.
* **Versatile:** Supports multiple programming paradigms, including object-oriented, functional, and procedural programming.
* **Open Source:** Python is free to use, distribute, and modify.

**Program in Python:**

print (‘Hello World!’) #Output: Hello World!

**Python Syntax:**

Python syntax can be executed by writing directly in the command line:

**Example:**

>>> print(‘Hello World!’)

Hello World!

**Python Indentation:**

🡪Indentation refers to the spaces at the beginning of the code line.

🡪In python programming languages the indentation in code is for reliability only, the indentation in Python is very important.

🡪Python uses indentation to indicate the block of code:

**Example:**

if 5 > 2:  
  print("Five is greater than two!")

Output:

Five is greater than two!

**Python Comments:**

🡪Comments can be used to explain the code.

🡪Comments can be used to make the code more reliable.

🡪Comments can be used to prevent execution when testing the code.

**Creating a comment:**

🡪Comments starts with a **#,** and python will ignore them.

🡪For the multiple line comments we use triple quotes **(””” “””)**.

**Python Variables:**

🡪Variables are containers for storing data values.

🡪Python has no command to create a variable.

🡪A variable is created the moment when you assign a value to it.

🡪Variable do not need to be declared with any particular type, and can even be changes after they have been set.

**Example:**

x=5

y=’Hi’

print(x) #result---5

print(y) #result---Hi

**Casting:**

If you want to specify the data type of a variable, this can be done with casting.

**Example:**

x = str(3)    # x will be '3'  
y = int(3)    # y will be 3  
z = float(3)  # z will be 3.0

**Get the type:**

You can get the data type of a variable with the **type()** function.

**Example:**

x = 5  
y = "John"  
print(type(x)) # int  
print(type(y)) # str

**Variable Names:**

🡪Variable names are case-sensitive.

🡪A variable name can be a short name or a descriptive name.

🡪Rules for Python variables:

* 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 \_)
* Variable names are case-sensitive (age, Age and AGE are three different variables)
* A variable name cannot be any of the [Python keywords](https://www.w3schools.com/python/python_ref_keywords.asp).

**Assign Multiple Values:**

🡪Python allows you to assign multiple values to multiple variables in one line:

**Example:**

x, y, z = "Orange", "Banana", "Cherry"  
print(x) # ‘Orange’  
print(y) # ‘Banana’  
print(z) #’Cherry’

🡪Make sure the number of variables matches the number of values, or else you will get an error.

**Output Variables:**

🡪The Python **print( )** function is often used to output variables.

**Example:**

x = "Python is awesome"  
print(x) # Python is awesome

**Local Variables**

Local variables are declared inside a function and can only be accessed within that function.

**Example:**

def my\_function():

local\_var = "I am local"

print(local\_var)

my\_function() # I am local

**Global Variables:**

🡪Variables that are created outside the function are known as global variables.

🡪Global variables are used by everyone, both inside of functions and outside.

**Example-1:**

Create a variable outside the function, and use it inside the function.

x = "awesome"  
def myfunc():  
  print("Python is " + x)  
myfunc() # Python is awesome

**Example-2:**

Create a variable inside a function, with the same name as the global variable.

x = "awesome"  
def myfunc():  
  x = "fantastic"  
  print("Python is " + x) # Python is fantastic  
myfunc()  
print("Python is " + x) # Python is awesome

**The global keyword:**

🡪To create a global variable inside a function, you can use the **global** keyword.

**Example-1:**

If you use the **global** keyword, the variable belongs to the global scope:

def myfunc():  
  global x  
  x = "fantastic"

myfunc()  
print("Python is " + x) # Python is fantastic

**Example-2:**

To change the value of a global variable inside a function, refer to the variable by using the global keyword:

x = "awesome"  
def myfunc():  
  global x  
  x = "fantastic"  
myfunc()  
print("Python is " + x) # Python is fantastic

**Data Types in Python**

Python provides various built-in data types to handle different types of data. These are categorized as follows:

**1. Numeric Types**

Used to store numerical values. There are 3 numeric types in python:

* **int**: Integer values (e.g., 10, -5)
* Int, or integer, is a whole number, positive or negative, without decimals, of unlimited length.

**Example:**

x=1

print(type(x)) # int

* **float**: Floating-point numbers (e.g., 10.5, -3.14)
* Float, or "floating point number" is a number, positive or negative, containing one or more decimals

**Example:**

x=10.5

print(type(x)) # float

* **complex**: Complex numbers (e.g., 3+4j, -2-5j)
* Complex numbers are written with a "j" as the imaginary part:

**Example:**

x=3+5j

Print(type(x)) # complex

**2. Sequence Types**

Used to store ordered collections:

* **str**: String of characters (e.g., 'hello', "Python")
* Strings are represented in single quotes or double quotes.

**Example:**

Print(‘Hello’) # Hello

* **list**: It is a ordered collection (e.g., [1, 2, 3])
* List is Mutable. It means after creation of the list, we can perform any kind of operation on it.
* List is represented in square brackets [ ].

**Example:**

x=[1,2,3]

print(x) # [1,2,3]

* **tuple**: It is a ordered collection (e.g., (4, 5, 6))
* Tuple is Immutable. It means after creation of the list, we cannot perform any kind of operation on it.
* Tuple is represented in parenthesis ( ).

**Example:**

x=(1,2,3)

print(x) # (1,2,3)

**3. Mapping Type**

Used to store key-value pairs:

* **dict**: Dictionary (e.g., {'name': 'John', 'age': 30})
* It is a collection of key-value pairs.
* It is represented in curly braces { }.

**Example:**

x={'name': 'John', 'age': 30}

Print(x) # {'name': 'John', 'age': 30}

**4. Set Types**

Used to store unordered collections of unique elements:

* **set**:It is an unordered collection of elements(e.g., {1, 2, 3})
* It is represented in curly braces { }.
* Set does not allow duplicate elements.
* Set is Mutable, but the elements in the set are Immutable.

**Example:**

thisset = {"apple", "banana", "cherry"}  
print(thisset) # {‘apple’, ’banana’, ’cherry’)

* **frozenset**: Immutable version of a set (e.g., frozenset([1, 2, 3]))
* It is an unordered and unindexed collection of unique elements.
* It also does not allow duplicate elements.

**Example:**

mylist = ['apple', 'banana', 'cherry']  
x = frozenset(mylist) #frozenset({‘apple’, ’banana’, ’cherry’})

**5. Boolean Type**

Represents truth values:

* **bool**: Can be True or False.
* You can evaluate any expression in Python, and get one of two answers, True or False.
* When you compare two values, the expression is evaluated and Python returns the Boolean answer:

**Example:**

print(10 > 9) # True

print(10 == 9) # False

print(10 < 9) # True

**6. Binary Types**

Used to store binary data:

* **bytes**: Immutable sequence of bytes (e.g., b'hello')

**Example**:

x=b’Hello’

print(x) # b’Hello’

* **bytearray**: Mutable sequence of bytes (e.g., bytearray ([65, 66, 67]))

**Example**:

x=bytearray(5)

print(x) # bytearray(b'\x00\x00\x00\x00\x00')

* **memoryview**: Provides a view of binary data (e.g., memoryview(b'hello'))

**Example**:

x = memoryview(bytes(5))

print(x) # <memory at 0x00B08FA0>

**7. None Type**

Represents the absence of a value:

* **NoneType**: Only one value, None

These data types help in managing and processing data efficiently in Python.

**Operators in Python**

Python provides a rich set of operators to perform various operations on variables and values. These operators are categorized as follows:

**1. Arithmetic Operators**

These are used to perform mathematical operations:

* **+ (Addition): a + b**
* It is used to add two numbers

**Example:**

a=10

b=20

sum\_=a+b

print("Addition of two numbers:", sum\_)

Output:

Addition of two numbers: 30

* **- (Subtraction): a – b**
* It is used to subtract two numbers.

**Example:**

a=20

b=10

sub=a-b

print('Subtraction of two numbers:', sub)

Output:

Subtraction of two numbers: 10

* **\* (Multiplication): a \* b:**
* It is used to multiply two numbers.

**Example:**

a=10

b=20

mul=a\*b

print('Multiplication of two numbers:' ,mul)

Output:

Multiplication of two numbers: 200

* **/ (Division): a / b**
* Here, division is done between two numbers.
* Quotient is taken as the result. The result is a **float** number.

**Example:**a=100

b=20

div=a/b

print('Division of two numbers:', div)

Output:

Division of two numbers: 5.0

* **// (Floor Division): a // b**
* Here, division is done between two numbers.
* Quotient is taken as the result. The result is a **int** number.

**Example:**

a=100

b=20

floor\_div=a/b

print('Division of two numbers:', floor\_div)

Output:

Division of two numbers: 5

* **% (Modulus): a % b**
* Here the division is done between two numbers.
* Remainder is taken as the result.

**Example:**

a=100

b=21

mod\_div=a%b

print('Modulo division of two numbers:', mod\_div)

Output:

Modulo division of two numbers: 16

* **\*\* (Exponentiation): a \*\* b**

**Example:**

a=3

b=2

exp=a\*\*b

print('Exponential of two numbers:', exp)

Output:

Exponential of two numbers: 9