**What is Python?**

Python is a popular Interpreted programming language. It was created by Guido van Rossum, and released in 1991.

**It is used for:**

* web development (server-side),
* software development,
* mathematics,
* system scripting.

**Python works on different platforms (Windows, Mac, Linux, Raspberry Pi, etc).**

**What can Python do?**

* Python can be used on a 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.
* Python can be used for rapid prototyping, or for production-ready software development.

**🔃Python setup on vscode in window click on link for setup manual:-**

<https://drive.google.com/file/d/1ZPIsX7jJgepS33Gvjp2Y531pZqnLY64w/view?usp=sharing>

**Python Popular Frameworks and Libraries:-**

* Web development (Server-side) - Django Flask, Pyramid, CherryPy
* GUIs based applications - Tk, PyGTK, PyQt, PyJs, etc.
* Machine Learning - TensorFlow, PyTorch, Scikit-learn, Matplotlib, Scipy, etc.
* Mathematics - Numpy, Pandas, etc.
* BeautifulSoup: a library for web scraping and parsing HTML and XML
* Requests: a library for making HTTP requests
* SQLAlchemy: a library for working with SQL databases
* Kivy: a framework for building multi-touch applications
* Pygame: a library for game development
* Pytest: a testing framework for Python
* Django REST framework: a toolkit for building RESTful APIs
* FastAPI: a modern, fast web framework for building APIs
* Streamlit: a library for building interactive web apps for machine learning and data science
* NLTK: a library for natural language processing

**-----------------------------------------------------------------------------------------------------**

**\*Python Variables:-**

A value-holding Python variable is also known as an identifier.

**Identifier Naming in Variables:-**

* The variable's first character must be an underscore or alphabet (\_).
* Every one of the characters with the exception of the main person might be a letter set of lower-case(a-z), capitalized (A-Z), highlight, or digit (0-9).
* White space and special characters (!, @, #, %, etc.) are not allowed in the identifier name. ^, &, \*).
* Identifier name should not be like any watchword characterized in the language.
* Names of identifiers are case-sensitive; for instance, my name, and MyName isn't something very similar.
* Examples of valid identifiers: a123, \_n, n\_9, etc.
* Examples of invalid identifiers: 1a, n%4, n 9, etc.

**Declaring Variable and Assigning Values:**

* Python doesn't tie us to pronounce a variable prior to involving it in the application. It permits us to make a variable at the necessary time.
* In Python, we don't have to explicitly declare variables. The variable is declared automatically whenever a value is added to it.
* The equal (=) operator is utilized to assign worth to a variable.

**Variable Names:**

name = "Devansh"

age = 20

marks = 80.50

print(name)

print(age)

print(marks)

**Consider the following valid variables name.**

name = "A"

Name = "B"

naMe = "C"

NAME = "D"

n\_a\_m\_e = "E"

\_name = "F"

name\_ = "G"

\_name\_ = "H"

na56me = "I"

**print**(name,Name,naMe,NAME,n\_a\_m\_e, NAME, n\_a\_m\_e, \_name, name\_,\_name, na56me)

**Output:**

A B C D E D E F G F I

**The multi-word keywords can be created by the following method.**

* **Camel Case** - In the camel case, each word or abbreviation in the middle of begins with a capital letter. There is no intervention of whitespace. For example - nameOfStudent, valueOfVaraible, etc.
* **Pascal Case** - It is the same as the Camel Case, but here the first word is also capital. For example - NameOfStudent, etc.
* **Snake Case** - In the snake case, Words are separated by the underscore. For example - name\_of\_student, etc.

**Multiple Assignment:**

Multiple assignments, also known as assigning values to multiple variables in a single statement, is a feature of Python.

**1. Assigning single value to multiple variables**

x=y=z=50

print(x)

print(y)

print(z)

**Output:**

50

50

50

**2. Assigning multiple values to multiple variables:**

a,b,c=5,10,15

print a

print b

print c

**Output:**

5

10

15

**-----------------------------------------------------------------------------------------------------------**

**Python Variable Types:**  Local Variable & Global Variables

**Local Variables:**

The variables that are declared within the function and have scope within the function are known as local variables

# Declaring a function

def add():

# Defining local variables. They has scope only within a function

a = 20

b = 30

c = a + b

print("The sum is:", c)

# Calling a function

add()

**Output:**

The sum is: 50

**Global Variables:**

Global variables can be utilized all through the program, and its extension is in the whole program. Global variables can be used inside or outside the function.

# Declare a variable and initialize it

x = 101

# Global variable in function

def mainFunction():

# printing a global variable

global x

print(x)

# modifying a global variable

x = 'Welcome To Python-program

print(x)

Function() # call

print(x)

**Output:**

101

Welcome To Python-program

Welcome To Python-program

**-----------------------------------------------------------------------------------------------------------------**

**Delete a variable:** We can delete the variable using the del keyword.

**syntax :** del <variable\_name>

# Assigning a value to x

x = 6

print(x)

# deleting a variable.

del x

print(x)

**Output:**

6

Traceback (most recent call last):

File "C:/Users/DEVANSH SHARMA/PycharmProjects/Hello/multiprocessing.py", line 389, in

print(x)

NameError: name 'x' is not defined

------------------------------------------------------------------------------------------------------------

**Some\_Example:-**

name = "Niranjan"

current\_addr = "Noida"

print("myname is" ,name, )

print("i am" ,name, "\n",current\_addr, "is my current address" )

**O/P:-**

myname is Niranjan

I am Niranjan

Noida is my current address

**Example:-**

print(":::::::::::::::::: simple single vairbale ::::::::::::::::::")

a = "niranjan"

print(a)

print("------------------------------------------------------------------------------")

print("\_\_\_\_\_\_\_\_Assigning single value to multiple variables\_\_\_\_\_\_\_\_\_\_")

x=y=z=50

print(x)

print(y)

print(z)

print("----------------------------------------------------------------------------")

print("-----------Assigning multiple values to multiple variables ------------")

a,b,c=5,10,15

print(a)

print(b)

print(c)

print(a , b, c, )

print("\*\*\*\*\*\*\*\*\*\*\* show sum of total variable \*\*\*\*\*\*\*\*\*\*")

print(+ a + b + c ) # show sum of total variable

print("-------------------------------------------------------------------------")

print("<<<<<<<<< VARIABLE AND TYPE OF VARIABLES >>>>>>>>>>>")

print("-------------------- Type of Variable:- local and global ----------------------")

print("\*\*\*\*\* Here we are Defining local variable\*\*\*\*\*")

# local variables- which is define into function

def add():

a = 20

b = 30

c = a + b

print("The sum is:", c)

# Calling a function

add() print("------------------------------------------------------------")

print ("------------ Here we going to define Global variable -------")

# Defining global variables

print(" ----------------------- Here we using global variables ------------------------- ")

# global variables means the variable which is use for entire code we can use it everywhere in code

d = 101;

# Global variable in function

def GLOBALVARIABLE():

print(d);

# modifying a global variable

x = 'Welcome To global variable'

print(d);

GLOBALVARIABLE(); # call

print(d);

print("------------------------------------------------------------------------------------")

print(">>>>>>>>>>>>>>>> To Delete a Vriable <<<<<<<<<<<<<<<<")

t = 3

print(t)

print(" -------------------------- Now Variable is Deleted Here --------------------------- ")

del(t)

print(t)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Basic Fundamentals: (**Tokens and Comments **)**

**Tokens:** The tokens can be defined as a punctuator mark, reserved words, and each word in a statement. The token is the smallest unit inside the given program.

**There are following tokens in Python:**

Keywords.

Identifiers.

Literals.

Operators.

Punctuators

**1. Keywords:**

They are used to define the Python language’s “Syntax” or “Structure.” A keyword is a reserved word. Python keywords form the vocabulary of the python language.

**and, not, or:** logical Operators

**as:** To create an alias

**assert:** For debugging

**break:** To break out of a loop

**if:** To create a conditional statement

**while:** To create a while loop

**2. Identifiers:**

In Python, an identifier is a name given to a Class, Function, or Variable.

**Characteristics of Python Identifier**

The initial letter of the identifier should be any letter or underscore (\_).

Upper and lower case letters have distinct characteristics.

Except for the initial letter, any digit from 0 to 9 can be part of the identification.

It shouldn’t be used as a keyword

Except for the underscore (\_), an identifier cannot contain any special characters.

Identifiers can be as long as you want them to be.

Case matters when it comes to identifier names. Myself and myself, for example, are not the same thing.

**3. Operators:**

Operators are tokens that, when applied to variables and other objects in an expression, cause a computation or action to occur. Operands are the variables and objects to which the computation is applied. There are 7 different operators.

**i)Arithmetic Operators: +**

**%**

**\***

**-**

**÷**

**ii) Relational Operators:**   **==** (check if two operands' values are equal)

**!=**  (check if two operands' values are not equal)

**>**  (check if two operands values are not identical)

**iii) Assignment Operators: +=** It adds the right side input to the left side input and then assigns the result to the left side input.

-= Augmented assignment operator- It takes the right side operand and subtracts it from the left side operand, then assigns the result to the left side operand.

**iv) Logical Operators:**

The logical AND operator makes a condition true if both operands are true or non-zero.

The logical OR operator returns true if one of the two operands is true or non-zero.

**v) Bitwise Operators:** The bitwise operator manipulates individual bits in one or more bit patterns or binary numbers. For example, If a binary XOR operator (^) is set in one input value but not both, it copies the matching binary 1 to the result.

**vi) Membership Operators:**

The membership operator checks for membership in successions, such as a string, list, or tuple. Like in a membership operator that fetches a variable and if the variable is found in the supplied sequence, evaluate to true; otherwise, evaluate to false.

**vii) Identity Operators**

When comparing the memory locations of two objects, identity operators are used. If two variables point to separate objects, it does not return true; otherwise, it returns false.

**4. Literals:**

Literals, tokens in Python, are data elements with a fixed value.

String Literals

Numeric Literals. These are further of three types, integer, float, and complex literals.

Boolean Literals

Literal Collection

**5. Punctuators:**

Punctuators are tokens in python employed to put the grammar and structure of syntax into practice. Punctuators are symbols that are used to structure programming sentences in a computer language. Some commonly used punctuators are: ‘, ‘ ,#, \ ,( ) ,{ },[ ] ,@ ,: , =

**\*Comments:** single-line comments, multi-line comments, documentation strings, and other Python comments.

**Advantages of Using Comments: -** Our code is more comprehensible when we use comments in it. It assists us in recalling why specific sections of code were created by making the program more understandable.

Readability of the Code

Restrict code execution

Provide an overview of the program or project metadata

To add resources to the code

**Single-Line Comments:**

**Example-**

# This code is to show an example of a single-line comment

print( 'This statement does not have a hashtag before it' )

**Output:** This statement does not have a hashtag before it

**Multi-Line Comments:**

We can also use # with multiple line as:-

# it is a

# comment

# extending to multiple lines

we utilize the strings inside triple quotes(""") as multi-line comments.

**Example-**

"""

this is our first program in multiline comment

can u tell me about this

is it right or not

give one line answer

"""

**Using String Literals:-**

'it is a comment extending to multiple lines'

**-----------------------------------------------------------------------------------------------**

**\*Python Data Types:**

Every value has a datatype, and variables can hold values. We can verify the type of the program-used variable thanks to Python. The type() function in Python returns the type of the passed variable.

a=10

b="Hi Python"

print(type(a))

print(type(b))

**Output**

<type 'int'>

<type 'str'>



**Standard data types:** Numbers

Sequence Type

Boolean

Set

Dictionary

A variable can contain a variety of value, a person's id must be stored as an integer, while their name must be stored as a string.

**Numbers -** Numeric values are stored in numbers. The whole number, float, and complex qualities have a place with a Python Numbers datatype.

When a number is assigned to a variable, Python generates Number objects.

**Example**

**a = 5**

print("The type of a", type(a))

**b = 40.5**

print("The type of b", type(b))

**c = 1+3j**

print("The type of c", type(c))

print(" c is a complex number", isinstance(1+3j,complex))

**Output:** The type of a <class 'int'>

The type of b <class 'float'>

The type of c <class 'complex'>

c is complex number: True

**Python supports three kinds of numerical data:-**

**Int:**  Whole number worth can be any length, like numbers 10, 2, 29, - 20, - 150, and so on. An integer can be any length you want in Python. Its worth has a place with int.

**Float:**  Float stores drifting point numbers like 1.9, 9.902, 15.2, etc. It can be accurate to within 15 decimal places.

**Complex:**  An intricate number contains an arranged pair, i.e., x + iy, where x and y signify the genuine and non-existent parts separately. The complex numbers like 2.14j, 2.0 + 2.3j, etc.

**--------------------------------------------------------------------------------------------------------------**

**Sequence Type:-**

**String**

The sequence of characters in the quotation marks can be used to describe the string. A string can be defined in Python **using single, double, or triple quotes.**

String dealing with Python is a direct undertaking since Python gives worked-in capabilities and administrators to perform tasks in the string.

When dealing with strings, the operation "hello"+" python" returns "hello python," and the operator + is used to combine two strings.

Because the operation "Python" \*2 returns "Python," the operator \* is referred to as a repetition operator.

**Example:**

**str = "string using double quotes"**

print(str)

**s = '''''A multiline**

**string'''**

print(s)

**Output:**

string using double quotes

A multiline

string

**List**

Lists in Python are like arrays in C, but lists can contain data of different types. The things put away in the rundown are isolated with a comma (,) and encased inside **square sections []**.

To gain access to the list's data, we can use slice [:] operators. Like how they worked with strings, the list is handled by the concatenation operator (+) and the repetition operator (\*).

**list1 = [1, "hi", "Python", 2]**

#Checking type of given list

print(type(list1))

#Printing the list1

print (list1)

# List slicing

print (list1[3:])

# List slicing

print (list1[0:2])

# List Concatenation using + operator

print (list1 + list1)

# List repetation using \* operator

print (list1 \* 3)

**Output:**

[1, 'hi', 'Python', 2]

[2]

[1, 'hi']

[1, 'hi', 'Python', 2, 1, 'hi', 'Python', 2]

[1, 'hi', 'Python', 2, 1, 'hi', 'Python', 2, 1, 'hi', 'Python', 2]

**Tuple**

In many ways, a tuple is like a list. Tuples, like lists, also contain a collection of items from various data types. A **parenthetical space ()** separates the tuple's components from one another. Because we cannot alter the size or value of the items in a tuple, it is a read-only data structure.

**Example:**

**tup = ("hi", "Python", 2)**

# Checking type of tup

print (type(tup))

#Printing the tuple

print (tup)

# Tuple slicing

print (tup[1:])

print (tup[0:1])

# Tuple concatenation using + operator

print (tup + tup)

# Tuple repatation using \* operator

print (tup \* 3)

**Output:**

<class 'tuple'>

('hi', 'Python', 2)

('Python', 2)

('hi',)

('hi', 'Python', 2, 'hi', 'Python', 2)

('hi', 'Python', 2, 'hi', 'Python', 2, 'hi', 'Python', 2)

Traceback (most recent call last):

File "main.py", line 14, in <module>

t[2] = "hi";

**Dictionary:**

A dictionary is a key-value pair set arranged in any order. It stores a specific value for each key, like an associative array or a hash table. Value is any Python object, while the key can hold any primitive data type.

The comma (,) and the curly braces are used to separate the items in the dictionary.

**d = {1:'Jimmy', 2:'Alex', 3:'john', 4:'mike'}**

# Printing dictionary

print (d)

# Accesing value using keys

print("1st name is "+d[1])

print("2nd name is "+ d[4])

**Output:-**

{1: 'Jimmy', 2: 'Alex', 3: 'john', 4: 'mike'}

1st name is Jimmy

2nd name is mike

**Boolean:-**

True and False are the two default values for the Boolean type. These qualities are utilized to decide the given assertion valid or misleading.

**# Python program to check the boolean type**

**print(type(True))**

**print(type(False))**

**print(false)**

**Output:**

<class 'bool'>

<class 'bool'>

NameError: name 'false' is not defined

**Set:-**

The data type's unordered collection is Python Set. It is iterable, mutable(can change after creation), and has remarkable components. The elements of a set have no set order; It might return the element's altered sequence. Either a sequence of elements is passed through the curly braces and separated by a comma to create the set or the built-in function set() is used to create the set. It can contain different kinds of values.

# Creating Empty set

**set1 = set()**

**set2 = {'James', 2, 3,'Python'}**

#Printing Set value

**print(set2)**

# Adding element to the set

**set2.add(10)**

**print(set2)**

#Removing element from the set

**set2.remove(2)**

**print(set2)**

**Output:**

{3, 'Python', 'James', 2}

{'Python', 'James', 3, 2, 10}

{'Python', 'James', 3, 10}