# Day-1

# Python Introduction

Python is a general-purpose, High-level programming language that is used for :

* Web development,
* Software development,
* Machine learning, and
* Data science
* Scripting etc

**Features/Advantages of Python:**

* 1. Simple and easy to learn
  2. Free and Open Source
  3. High Level Programming language
  4. Platform Independent
  5. Portable
  6. Dynamically Typed
  7. Both Procedure Oriented and Object Oriented
  8. Interpreted as well as compiler based
  9. Extensive Library(like numpy, panda, matplotlib etc…)

**Limitations of Python:**

**Performance and Speed:** Python, being an interpreted language, it is slower as compared to compiled languages like C or Java. This can pose challenges for applications requiring high performance, such as real-time systems or intensive computations.

**Dynamic Typing:** As a dynamically typed language, Python does not check variable types at compile time, which can result in runtime errors.

**Web Development Support:** Python lacks built-in support for web development. Developers need to rely on third-party frameworks and libraries to create web applications in Python.

**Mobile Computing:** Python is not as well-suited for mobile computing as languages like Java or Kotlin, partly due to its high memory consumption.

**Memory Management:** The dynamic typing in Python allows variable data types to change at any time, which can lead to memory leaks and other memory usage issues.

**Application of Pythons:**

1. Healthcare

2. Web development

3. Data Science and Analytic

4. Scientific Computing

5. Automation

6. Scripting

7. Desktop GUI Application

8. Game Development

9. Networking etc

**Python can take almost all programming features from different languages:--**

1. Functional Programming Features from C

2. Object Oriented Programming Features from C++

3. Scripting Language Features from Perl and Shell Script

4. Modular Programming Features from Modula-3(Programming Language)

**Flavors of Python or types of python compilers:**

**1. CPython:** It is the standard flavor of Python. It can be used to work with C

lanugage Applications.

**2. Jython or JPython:** It is for Java Applications. It can run on JVM

**3. IronPython:** It is for C#.Net platform

**4. PyPy:** The main advantage of PyPy is performance will be improved because

JIT (just in time)compiler is available inside PVM.

**5. RPython:** For Ruby Platforms.

**6. AnacondaPython:** It is specifically designed to handle large volumes of data processing, enabling Python code to run efficiently in web browsers.

**Day2**

**Python Versions:**

1. **Python 1.x:- Python 1.0 (January 1994)**
2. **Python 2.x:- Python 2.0 (October 2000)**
3. **Python 3.x:- Python 3.0 (December 2008)**

**Why you chosen python :-**

1. **Ease to Learn**
2. **Strong Demand in the Job Market**
3. **Versatility**
4. **Rich Libraries and Frameworks**
5. **Huge Community and Support**

**Internal working of python**

**PVM**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Source Code/Program** |  | **Byte\_Code** |  | **PVM** |  | **Binary\_Code to Machine\_Code** |  | **O/P** |

**first.py first.pyc**

**Python Compiler**

**Convert python code into byte-code:----**

python -m py\_compile first.py

python first.cpython-310.pyc

**Day-3**

**Python Variable/Identifier**

Variable/identifier

**x y**

object Value

1000

10

**10101010 10101010**

Memory Location

**Note:- In python, memory allocated to the object, not variable.**

**What is identifier?**

Identifier is basically used to assign a name for any variables, functions, classes, modules etc.

**What is python variable?**

A variable is basically used to assign a name to a memory location that holds the corresponding value.

**To summarize, a variable is a**

1. Name assigned to memory location
2. Refers to a object
3. Hold some data

**Properties of variable:**

1. Data **Type**
2. **Scope**
3. **Value**
4. **Memory Location**

##### **Multiple values assign in multiple variables in a single line in Python.**

##### **Single values assign in multiple variables in a single line in Python.**

##### **Re-initialize the variables in Python**

**Note: Python doesn’t support const type variable. But if you want to create use capital letter for that.**

**Rules for creating variables/** **Identifiers in Python:**

* 1. A variable name must start with a letter or the underscore character.

( myvar, my\_var, \_my\_var, myVar, MYVAR, myvar2 )

* 1. A variable name cannot start with a number.

(2myvar,)

* 1. A variable name can only contain alpha-numeric characters and underscores (A-z, 0-9, and \_ ).

( myvar, my\_var, \_my\_var, myVar, MYVAR, myvar2 )

* 1. Variable names are case-sensitive.

(age, Age and AGE are three different variables)

* 1. A variable name cannot be any of the python keywords.(if , else, for ,while, try, etc)

# how to fetch all python-keywords

import keyword

print("The list of keywords is : ")

print(keyword.kwlist)

**Recommendation:**

**Use Descriptive Variable Names:** Ensure that your variable names clearly indicate what they store. For example, age is an appropriate name for a variable that holds a person's age.

**Use Snake Case for Variable Names:** Adopt snake\_case for naming variables, using lowercase letters with underscores to separate words. For instance, my\_variable is a good example of snake\_case.

**Avoid Abbreviations in Variable Names:** Abbreviations can make your code harder to read and understand. For example, num is not a good name for a variable that stores the number of students in a classroom. A better name would be number\_of\_students.

# Assign Multiple Values in multiple variables in single line:-

## Many Values to Multiple Variables

### Example:-

x, y, z = "Neeraj", "Ravi", "Rahul"  
print(x)  
print(y)  
print(z)

## One Value to Multiple Variables in single line:

### Example:-

x = y = z = "Neeraj Kumar"  
print(x)  
print(y)  
print(z)

1. **Advance examples:-**

Example:-

city = ["Bhopal", "Indore", "Jabalpur"]  
x, y, z = city  
print(x)  
print(y)  
print(z)

**Python Comments:-**

1. single line comments:--- ( # ---------------) ctrl+/
2. Multi-line comments:---(‘‘‘ ------------

-----------’’’)

**Day-4**

**Data Type:**

Data-type is a property/attribute of variable that tell us, what kind of value stored in it.

|  |  |
| --- | --- |
| **Data Types in python** | |
| **Numeric** | int, float, complex |
| **String** | str |
| **Sequence** | list, tuple, range |
| **Mapping** | dict |
| **Set** | set, frozenset |
| **Boolean** | bool |
| **Binary** | bytes, bytearray |
| **None** | NoneType |

**1. Numeric data-type:----**

# --------------------------------------Numeric data-type------------------------------------

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

x=20

y=10

z1=x+y

z2=x\*y

z3=x/y

print(z1)

print(type(z1))

print(id(z1))

print(z2)

print(type(z2))

print(id(z2))

print(z3)

print(type(z3))

print(id(z3))

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

x=20.0

y=10.0

z1=x+y

z2=x\*y

z3=x/y

print(z1)

print(type(z1))

print(z2)

print(type(z2))

print(z3)

print(type(z3))

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

x=20+10j

y=10+20j

z1=x+y

z2=x\*y

z3=x/y

print(z1)

print(type(z1))

print(z2)

print(type(z2))

print(z3)

print(type(z3))

O/P:-

------------------- integer ------------------------

30

<class 'int'>

2611473312976

200

<class 'int'>

2611473506960

2.0

<class 'float'>

2611479354288

--------------------- float ------------------------

30.0

<class 'float'>

200.0

<class 'float'>

2.0

<class 'float'>

--------------------- complex ------------------------

(30+30j)

<class 'complex'>

500j

<class 'complex'>

(0.8-0.6j)

<class 'complex'>

**2. String data-type:----**

# collection of charactors are called string..........

# representation of string............................

my\_str = 'Neeraj kumar'

my\_str1 = "Neeraj Kumar"

my\_str2 = '''Neeraj kumar

    patel '''

print(my\_str)

print( my\_str1)

print( my\_str2)

print(type(my\_str))

print(type(my\_str1))

print(type(my\_str2))

O/P:--

Neeraj kumar

Neeraj Kumar

Neeraj kumar

    patel

<class 'str'>

<class 'str'>

<class 'str'>

**3. Sequential/collection data-type:----**

print("----------------List data-type-----------------")

my\_list = ["Neeraj",10,20,40,30,10.5,"Nee123"]

print(my\_list)

print(type(my\_list))

print("----------------Tuple data-type-----------------")

my\_tuple = ("Neeraj",10,20,40,30,10.5,"Nee123")

print(my\_tuple)

print(type(my\_tuple))

print("----------------Range data-type-----------------")

my\_range = range(10)

print(list(my\_range))

print(tuple(my\_range))

print(type(my\_range))

print(type(list(my\_range)))

print(type(tuple(my\_range)))

O/P:--

----------------List data-type-----------------

['Neeraj', 10, 20, 40, 30, 10.5, 'Nee123']

<class 'list'>

----------------Tuple data-type-----------------

('Neeraj', 10, 20, 40, 30, 10.5, 'Nee123')

<class 'tuple'>

----------------Range data-type-----------------

[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

(0, 1, 2, 3, 4, 5, 6, 7, 8, 9)

<class 'range'>

<class 'list'>

<class 'tuple'>

**4. Mapped data-type:----**

my\_dict = {'name':"Neeraj",'age':37,"quali":'M.Tech'}

print(my\_dict)

print(type(my\_dict))

O/P:--

{'name': 'Neeraj', 'age': 37, 'quali': 'M.Tech'}

<class 'dict'>

**5. Set data-type:----**

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

my\_data = {10,20,30,20,30,40,30}

print(set(my\_data))

print(type(my\_data))

my\_data1 = {"Neeraj",20,30,20,30,"Rahul",30}

print(set(my\_data1))

print(type(my\_data1))

# Frozen sets are immutable, meaning that their elements cannot be changed after they have been created.

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

my\_data2 = {10,20,"Neeraj",20,"Rahul",30,"Raj",40}

print(set(my\_data2))

print(frozenset(my\_data2))

O/P:--

-----------------------------set()---------------------------------

{40, 10, 20, 30}

<class 'set'>

{'Neeraj', 20, 30, 'Rahul'}

<class 'set'>

-----------frozenset()----------------

{20, 'Rahul', 40, 10, 'Neeraj', 'Raj', 30}

frozenset({20, 'Rahul', 40, 10, 'Neeraj', 'Raj', 30})

**6. Boolean data-type:----**

# ------------------------ bool() -----------------

my\_data = [10,20,30]

my\_data1 = []

print(bool(my\_data))

print(bool(my\_data1))

O/P:--

True

False

**7. Binary data-type:----**

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

my\_data = [10,20,25,32,35]

x = bytes(my\_data)

print(type(x))

print(x)

# 'bytes' object does not support item assignment

# x[0]=100

# print(x)

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

my\_data = [10,20,25,32,35]

x = bytearray(my\_data)

print(type(x))

print(x)

# 'bytearray' object supports item assignment

x[0]=100

print(x)

O/P:--

-----------------Bytes---------------

<class 'bytes'>

b'\n\x14\x19 #'

-----------------bytearray---------------

<class 'bytearray'>

bytearray(b'\n\x14\x19 #')

bytearray(b'd\x14\x19 #')

**8. None data-type:----**

x=None

print(type(x))

print(id(x))

O/P:--

<class 'NoneType'>

140721558756568

**Some python in-built functions:**

* 1. **print():** show output on terminal.
  2. **input():** take input from user at runtime.
  3. **id():** Use to find memory location of an variable.
  4. **type():** Use to find-out type of data which variable have.