# Introduction to python

Thursday, 15 February 2024 6:40 PM

Python has been an open Source programming language from the very beginning, encouraging contributions from across the globe. to diversify the features of language itself.

Python 1.0 Released 2004 January → designed to target Mobile Operating system

Python 2.0 2000 → Two significant enhancement
① Provisioning a garbage collector
② support of unicode.

Python 3.0 → 2008

Currently 2.X & 3.X is maintained simultaneously and can be used by programmers.

Python 3 Installation

① download & install from official python.org website.

check the installation by command

python -- version
or
python3--version.

pip the package manager for python.

pip is used to install all the python libraries.

check the pip version

pip --version
or
pip 3 -- version

① Don't have pip installed in your system

download and install from

https://pypi.org/project/pip/:

⇒(Installing Python Packages)

fewBasic modules that come pre installed on your system.

Rest can be installed as follows

① Install package

① pip install <package-name>
② pip 3 install <package-name>

To remove the package

① pip uninstall <package-name>
② pip3 uninstall <package-name>

look at help menu
pip install --help

Ways to run Python Program

① Using the python Interpreter

The python Interpreter comes installed with the Python installation

typing the following command on your terminal will open the python interpreter.

① Python
② Python 3.

```
>>> 2+3
 5
>>> a=8
>>> b=9
>>> a+b
17
>>> a="hello World"
>>> a
  hello World
```

(Using a Text Editor)

Python program containing more than six lines of code are generally written in text editor.

python is an Interpreted language, In case of an Error the execution will stop and the error will be thrown with the line number.

① The program written in text editor is saved by .py extension.

② Open terminal in the folder where program is saved
The command to run the python program
python file name.py
python 3 file name.py

Imp.

## Basics of python programming

① The variables in python are case sensitive

Obj and OBJ are two different variables

Python has help integrated to every step to understand about particular object

① help (Obj)

directory structe of an object can be viewed by following command

dir(Obj)

## Data types

In python programming, you do not have to mention the data type with each variable that you define, as it is dynamically typed.

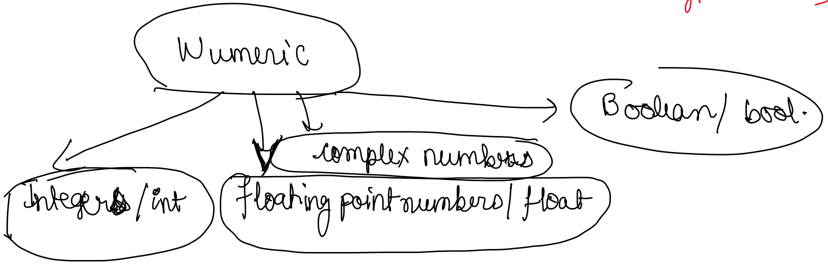
python has several datatype as follows.

None: The none variable is similar to null variable in other language

If variable is not assigned any values, it automatically assumes the value of None

for Example:

```
var_none = None
 if var_none is None:
        print ("The variable is type None")
 else
        print ("The variable is not type None")
```



① Integers/int : It can be of any length and is only limited by the memory available for your program

② Floating-point number/float :- It can indicates real numbers and can be accurate upto 15 decimal places.

③ complex number: a + bj
a → real part
b → complicated part
a and b → float values.

The complex numbers are not commonly used in python language or programming.

Boolean / bool : It implements true or false.

1 → true
0 → false

```
>> a = 5
>> b = 6.5
>> c = int(b)              >>> f = float(a)
>> d = complex(a, b)
>> e = true
>> type(a)                 < class 'int' >
>> type(b)                 < class 'float' >
>> type(c)                 < class 'int' >
>> c                         6
>> d                       (5 + 6.5j)
>> type(e)                 < class 'bool' >
>>> f                        5.0
>>  type(f)                  float
```

converting from one type to another.

List

lists are comma-separated values, that are initialized using square bracket.

All the items in an list need not to be of same type. list are mutable

Example

```
a = [1, 2, 3, 4]
b = [1, 'a', 2, 'b']
c = ['Hello', 'world', 2]
d = [3, 2.2, 'Python']
```

Accessing the elements of list

| Command | Output |
|---|---|
| a[1] | 2 |
| a[:2] | [1, 2] |
| a[2:] | [3, 4] |
| a[1:3] | [2, 3, 4] |
| a[1] = 5 | |
| a | [1, 5, 3, 4] |

mutable

Tuple

tuple are similar to lists Except the fact that they are not mutable.

A tuple can be declared using round Bracket ('( )')

The elements of a can also be of different datatypes.

| | Output |
|---|---|
| a = (1, 2, 3) | |
| a | (1, 2, 3) |
| a [1] = 5 | Error |
| a [2] | 3 |

## Sets

Set is similar to list or a tupple with some constraints

① → The sets can not have duplicate Elements
② → The whole set can be reassigned, individual elements in set are not mutable.
③ no way to access an individual element.

Output

```
        fruits1 = set(["mango", "Apple", "Banana"])
        fruits2 = set(["Grapes", "mango"])
union ← fruits3 = fruits1 | fruits2          ← ({"Mango", "Apple", "Banana", "Grapes"})  ← Set has only one mango.
        fruits3
add ←   fruit1.add("Jackfruit")
delete ← fruit1.discard("mango")
```

## Dictionary

Dictionaries are used to store data values in key : value pairs.
A dictionary is a collection which is ordered, changeable and do not allow duplicates.

Example :

```
Dict = {
   "brand" : "ford"
   "model" : "Mustang"
   "year" : 1996;
}
print (Dict)
print ( len( Dict)) →          → ③
print ( Dict ["Brand"])        → ford
print (type ( dict))           → <Class 'dict'>
```

Output

```
{'brand' : 'ford',
 'model' : 'mustang'
 'year' : 1996 }
```

## Strings

Strings can be declared using single quotes(' ') or double quotes (" ")
Strings are immutable
individual characters of strings can be accessed using square brackets ("[]")

```
a = "Hello"
b = 'World'
print(a)
print(b)
print (len (a))
print( a [0])
print(a.lower())
print(a.upper())
```

Output

```
Hello
World
5
H
hello
HELLO
```

Concatenate

```
c = a+b
print(c)
```

```
Hello World
```

Replace

```
a = "Hey World"
b = a.replace('y', 'llo')

print(b)                    Hello World
```

Slice

```
a = "Message to Slice"
b = a[4:16]
```

```
print(b)        → age to Slice
```

Output

Split

```
a = " Mango Fruit"
b = a.split(' ')
print(b)  →     ('mango', 'fruit')
```

output

Join

```
a = ['apple', 'and', 'mango')
b = ' '.join(a)

print(b)  →     ("apple and mango"))
```

Output

Reverse

```
a = "string"
print( a[::-1])  →   gnirts
```

Output

Flow control statements

Refers to conditional statements and looping statements

If Else

logical conditions that returns true or false value.

Example

```
a = 10
b = 20
if (a == b)
    print(" a is Equal to b")
Else
    print("b is not Equal to b")
```

Looping statements

written to reduce excessive writing number of line

For loop

Syntax

```
① for var in arr:
  or
② for var in range(x):
```

| Example | Output |
|---|---|
| ① `arr = [1,2,3,4]` `for x in arr:` `print(x)` | 1, 2, 3, 4 |
| ② `for i in range(10):` `print(i)` | 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 |

While loop syntax

```
while Condition :
```

Example :

```
while i < 6 :
    print(i)
    i++
```

Functions

A function can Execute Bunch of Statements; it can take a set of parameters and also be able to return a value.

A function in Python can be defined using the keyword def.

Syntax:

```
def function_name ( function-parameters):
    function-body
```

Example:

①
```
def add(a,b):
    c = a+b
    return c
d = add(5,6)
print(d)
```
$\longrightarrow$

```
output
11
```

②
```
def loop (arr):
    var = 0
    for i in arr:
        if (i%2 ==0):
            var += i
        else:
            continue
    return var
varArr = [1,2,3,4]
Print(loop (varArr) )
```
$\longrightarrow (2+4) =$

```
Print/ Output
6
```

Advanced python programming

Libraries

modules/library allow you to use certain functions repeatedly.

Modules can be used in your program by importing the libraries at the start of the program.

```
import math
from math import pi
 r=5
 a= r*r*pi
 print ('Area of circle is', a)
```

## classes

A class is like a template for creating object.
You can create classes in python using the keyword class

```
class    class Name:
          <statement1>
          <statement2>
           :
           :
          <statement N>
```

Example 1:

```
class Fruits:
    fruit1 = "mango"
    def printFruit():
        return "apple"
  Fruits fruit;
  isApple = fruit.printFruit();   <----   O/P
  print(is Apple)                          apple
```

Example 2:

```
class ml:
    ''' this is doc'''
    var = 1
    def printhello(self):
        print('Hello world)
 a= ml()
```

documentation in python is declared using ''' or " " " "

The keyword self refers to the instance of the class.

a= ml() ⟶ object of a class

|  | output |
|---|---|
| ① print (ml. var) | 1 |
| ② print (ml.__doc__) | this is doc |
| ③ print ( ml. printhello) | <function ml.printhello at 0X76---> |
| ④ print (a. var) | 1 |
| ⑤ print (a.printhello()) | <function ml.printhello at 0X76---> |

## Constructor in Python

A constructor is a special type of method in Python that is called when an object is created. It is used to initialize the object attributes. The name of constructor method is always __init__.

Example

```
    Class Person:
        def __init__ (self, name, age):
            self. name = name
            self. age = age
  person = Person ("Alice", 25)
```

```
print(person.name)
print(person.age)
```

Output:

```
Alice
25
```

## Exception handling

The Exception handling needs to be written in the code to handle any Exceptions that we encounter while Executing our python program.

- Try: The try keyword lets you write specific lines of the code and test them for errors as one single block.
- except: The except keyword lets you handle the error.
- finally: The finally block let you execute the code that you would like excute irrespective of the result of the try or except block.

Example

```
try:
    print(var)
except:
    print("var not defined")
Finally:
    print("define var")
```

Output

```
var not defined

define var
```