Object Oriented Programming

OOPS:

- It is a concept designed & developed by some other person and it is implemented in various programming languages. One of the languages is Python.
- This concept is designed from our day-to-day life.
- Because of this concept, language learning and understanding will be easier.
- In Python, Object Oriented Programming (OOPs) is a programming paradigm that uses objects and classes in programming.

Class

A class is a collection of objects. A class contains the blueprints or the prototype from which the objects are being created. It is a logical entity that contains some attributes and methods.

Syntax to create class:

```
1.class class_name:
        pass

2.class class_name():
        pass

3.class class_name(object):
        pass
```

Objects

The object is an entity that has a state and behaviour associated with it. It may be any real-world object like a mouse, keyboard, chair, table, pen, etc. Integers, strings, floating-point numbers, even arrays, and dictionaries, are all objects.

An object consists of:

State: It is represented by the attributes of an object. It also reflects the properties of an object.
Behaviour: It is represented by the methods of an object. It also reflects the response of an object to other objects.

e.g. 1.

```
class A:
        x = 10  #class variable
        def m1(self):
                print('m1 – A')

a = A() #object creation
a.x = 20 #instance variable
print(a.x)
```

```
a.m1()
a1 = A()
print(a1.x)
a1.m1()
```

OUTPUT:

```
20
m1 – A

10
m1 -- A
```

In above program, m1 method has a parameter **self** which is used to hold **current object**. It signifies that which object is calling the method.
Instance variable is a variable which belongs to that object only. **Class variable** is shared among all objects.

2.

```
class Student:
        def display(self):
                print('Roll no : ',self.rollno)
                print('Name : ',self.name)
s1 = Student()
s1.rollno = 1
s1.name = 'abc'
s1.display()
s2 = Student()
s2.rollno = 2
s2.name = 'xyz'
s2.display()
```

OUTPUT:

```
Roll no : 1
Name : abc
Roll no : 2
Name : xyz
```

Constructor in class:

- Constructor is denoted by a magic method named as **__init__**
- It is a magic method as it is called automatically when we create object of a class.
- It is used to initialize the instance variable.

e.g 1.

```
class A:
        def __init__(self):
                print('Constructor of A')

a1 = A()  #__init__ is called
a2 = A()
```

OUTPUT:

```
Constructor of A
Constructor of A
```

2.

```
class Student:
        def __init__(self,rn,nm):
                self.rollno = rn
                self.name = nm
s1 = Student(1,'abc')
print(s1.rollno,s1.name)
s2 = Student(2,'xyz')
print(s2.rollno,s2.name)
```

OUTPUT:

```
1 abc
2 xyz
```

__str__ in class:

This method **returns the string representation of the object**. This method is called when print() is invoked on an object.

```
class Student:
        def __init__(self,rn,nm):
                self.rollno = rn
                self.name = nm
        def __str__(self):
                return 'Rollno : {} Name : {}'.format(self.rollno,self.name}

s1 = Student(1,'abc')
print(s1)
s2 = Student(2,xyz)
print(s2)
```

OUTPUT:

```
Rollno : 1 Name : abc
Rollno : 2 Name : xyz
```