OOPs in Python

· Python is a object oriented language from the beginning. So, it is easy to create and use classes and objects.

· Before going to the concepts let us discuss about some basic definition in oops.

Class : Generally, A class is defined as a blueprint of an object. A class is a code template for the object.

Object : An object is the instance of the class. An object contains both class variables and instance variables and methods.

Class Variable : The variable that is used by every instance of a class and these are defined inside a class but outside the class method. These are not used as effectively as Instance variables.

Function Overloading : Assigning different action to a particular function is called function overloading. The action to be performed depends on objects or arguments.

Operator Overloading : Assigning more function to a particular operator is called operator overloading.

Instantiation : The process of creating a instance object for the class.

Method : This is a function which is defined inside a class.

Instance : An object to which the class is instantiated is called instance.

Instance variable : The variable which is defined inside a method and belongs only to that instances of a class.

Data members : A variable that contains the data of class and objects.

Classes and Objects :

Creating a Class : A class is created by using the keyword **class.**

Syntax:

```
class Classname:
        '''class details'''
        pass
```

· Class name should start with Uppercase letter only.

· A class creates a new local namespace where all its attributes are defined. Attributes may be data or functions.

· There are also special attributes in it that begins with double underscores (__). For example, __doc__ gives us the docstring of that class.

· When we define a class, a new class object is created with the same name. This class object allows us to access the different attributes as well as to instantiate new objects of that class.

Example:

```
class Firstclass:
    '''This is the first class example in python'''
    a = "digitallync"
    def func(self):
        print (5+6)
print (Firstclass.a)
print (Firstclass.func)
print (Firstclass.__doc__)
```

Output :

```
digitallync
<function Firstclass.func at 0x000002E40DCDF620>
This is the first class example in python.
```

· In the example we created a class with classname as Firstclass and pass a class variable=”digitallync” and a method func in which addition is performed.

· We can access these variables and methods by using a operator called dot operator(**.**) along with the classname.

· Here when tried to access the function it gives the location of the func.

Self() : In python a class must have an extra arguments in method definition. Value cannot be provided to this arguments as python default provided it.

· Even though a function does not have any arguments but this argument is default as self.

· When a method is called using the object obj.functionname(par1,par2),python automatically converts this into MyClass.method(obj,par1,par2).

<u>Creating an Object</u> : Class object is used to access different attributes.

· It is also used to create new object instances of that class. Creating an object is same as functions.

Syntax:

```
objname=Classname()
```

· This will create a new instance object with objname.

· Attributes may be data or method. Method of an object are corresponding functions of that class.

· We can say **Classname.func** is **function object** whereas **objname.func** will be a **method object.**

Example:

```
class Firstclass:
    '''This is the first class example in python'''
    a = "digitallync"
    def func(self):
        print (5+6)
obj = Firstclass()
print (Firstclass.func)
print (obj.func)
obj.func()
```

Output :

```
<function Firstclass.func at 0x00000247A3D6F620>
<bound method Firstclass.func of <__main__.Firstclass object at 0x00000247A3D6C3C8>>
11
```

· We called the method simply as obj.func() without any arguments it still worked.

· This is because, whenever an object calls its method, the object itself is passed as the first argument. So, obj.func() translates into Firstclass.func(obj).

Constructor : In python the in-built __init__ is called as the class constructor.

· Class functions that begins with double underscore (__) are called special functions as they have special meaning.

· __init__() gets called whenever a new object of that class is instantiated.

Example:

```
class Complex:
    def __init__(self, r=0,i=0):
        self.real = r
        self.imag = i
    def print(self):
        print('{0} + {1}j is complex number'.format(self.real,self.imag))
c=Complex(8,9)
c.print()
c1 = Complex(10)
c1.print()
```

Output:

```
8 + 9j is complex number
10 + 0j is complex number
```

· We define a new class to represent complex numbers. It has two functions, __init__() to initialize the variables and print() to display the complex numbers properly.

· When we provide single value it will take it as real value because in the print statement we used string format to display and order is specified as {0} and {1}.