# Python Class and Object

Python is an object oriented programming language. Unlike procedure oriented programming, where the main emphasis is on functions, object oriented programming stress on objects.

Object is simply a collection of variables and methods (functions) that act on those data. And, class is a blueprint for the object.

We can think of class as a sketch of a house. It contains all the details about the floors, doors, windows etc. Based on these descriptions we build the house. House is the object.

# Class

A class is a blueprint for any functional entity which defines its properties and its functions. Classes are a user-defined data type that is used to encapsulate data and associated functions together. It also helps in binding data together into a single unit. Like Human Being, having body parts, and performing various actions.

In Python, a class is defined by using a keyword class like a function definition begins with the keyword def.

**Class Syntax**

```
class ClassName:
    Statements
```

The Statements consists of all the component statements defining class members, data attributes and functions.

| Example | Output |
|---|---|
| class abc:<br>  a=20<br>  print a | 20 |

# Object

Object is an entity that has state and behavior. It may be anything. It may be physical and logical. We can create new object instances of the classes. The procedure to create an object is similar to a function call.

**Object Syntax**

```
obj=class_name()
```

This will create a new instance object named obj. We can access attributes of objects using the object name prefix.

Attributes may be data or method. Method of an object are corresponding functions of that class. Any function object that is a class attribute defines a method for objects of that class.

| Example | Output |
|---|---|
| class abc:<br>  def fun(self):<br>  print "Class and object"<br>obj=abc()<br>obj.fun() | Class and object |

You may have noticed the self parameter in function definition inside the class but, we called the method simply as obj.fun() without any arguments. This is because, whenever an object calls its method, the object itself is passed as the first argument.

## self Parameter

The self parameter is a reference to the class itself, and is used to access variables that belongs to the class.

It does not have to be named self , you can call it whatever you like, but it has to be the first parameter of any function in the class.

## The __init__() Function

To understand the meaning of classes we have to understand the built-in __init__() function.

All classes have a function called __init__(), which is always executed when the class is being initiated.

Use the __init__() function to assign values to object properties, or other operations that are necessary to do when the object is being created.

| Example | Output |
|---|---|
| class abc:<br>  def __init__(self,a,b): | Value of A is 10 and B is 20 |

```
        self.a=a
        self.b=b

    def fun(self):
        print "Value of A is", self.a,"and B is", self.b
obj=abc(10,20)
obj.fun()
```

| Example | Output |
|---|---|
| class abc:<br>  def __init__(self,a,b):<br>  self.a=a<br>  self.b=b<br><br>  def fun(self):<br>  print "Value of A is", self.a,"and B is", self.b<br>obj=abc(10,20)<br>obj.fun()<br>obj=abc(100,200)<br>obj.fun() | Value of A is 10 and B is 20<br>Value of A is 100 and B is 200 |