**Class** − A user-defined prototype for an object that defines a set of attributes that characterize any object of the class. The attributes are data members (class variables and instance variables) and methods, accessed via dot notation.

**Class variable** − A variable that is shared by all instances of a class. Class variables are defined within a class but outside any of the class's methods. Class variables are not used as frequently as instance variables are.

**Data member** − A class variable or instance variable that holds data associated with a class and its objects.

**Function overloading** − The assignment of more than one behavior to a particular function. The operation performed varies by the types of objects or arguments involved.

**Instance variable** − A variable that is defined inside a method and belongs only to the current instance of a class.

**Inheritance** − The transfer of the characteristics of a class to other classes that are derived from it.

**Instance** − An individual object of a certain class. An object obj that belongs to a class Circle, for example, is an instance of the class Circle.

**Instantiation** − The creation of an instance of a class.

**Method** − A special kind of function that is defined in a class definition.

**Object** − A unique instance of a data structure that's defined by its class. An object comprises both data members (class variables and instance variables) and methods.

**Operator overloading** − The assignment of more than one function to a particular operator.

The **getattr(obj, name[, default])** − to access the attribute of object.

The **hasattr(obj,name)** − to check if an attribute exists or not.

The **setattr(obj,name,value)** − to set an attribute. If attribute does not exist, then it would be created.

The **delattr(obj, name)** − to delete an attribute.

class Employee:

'Common base class for all employees'

empCount = 0

def \_\_init\_\_(self, name, salary):

self.name = name

self.salary = salary

Employee.empCount += 1

def displayCount(self):

print "Total Employee %d" % Employee.empCount

def displayEmployee(self):

print "Name : ", self.name, ", Salary: ", self.salary

class Employee:

'Common base class for all employees'

empCount = 0

def \_\_init\_\_(self, name, salary):

self.name = name

self.salary = salary

Employee.empCount += 1

def displayCount(self):

print "Total Employee %d" % Employee.empCount

def displayEmployee(self):

print "Name : ", self.name, ", Salary: ", self.salary

"This would create first object of Employee class"

emp1 = Employee("Zara", 2000)"This would create second object of Employee class"

emp2 = Employee("Manni", 5000)

emp1.displayEmployee()

emp2.displayEmployee()print "Total Employee %d" % Employee.empCount

The **getattr(obj, name[, default])** − to access the attribute of object.

The **hasattr(obj,name)** − to check if an attribute exists or not.

The **setattr(obj,name,value)** − to set an attribute. If attribute does not exist, then it would be created.

The **delattr(obj, name)** − to delete an attribute.

**\_\_dict\_\_** − Dictionary containing the class's namespace.

**\_\_doc\_\_** − Class documentation string or none, if undefined.

**\_\_name\_\_** − Class name.

**\_\_module\_\_** − Module name in which the class is defined. This attribute is "\_\_main\_\_" in interactive mode.

**\_\_bases\_\_** − A possibly empty tuple containing the base classes, in the order of their occurrence in the base class list.

class Parent: # define parent class

parentAttr = 100

def \_\_init\_\_(self):

print "Calling parent constructor"

def parentMethod(self):

print 'Calling parent method'

def setAttr(self, attr):

Parent.parentAttr = attr

def getAttr(self):

print "Parent attribute :", Parent.parentAttr

class Child(Parent): # define child class

def \_\_init\_\_(self):

print "Calling child constructor"

def childMethod(self):

print 'Calling child method'

**issubclass(sub, sup)** boolean function returns true if the given subclass **sub** is indeed a subclass of the superclass **sup**.

The **isinstance(obj, Class)** boolean function returns true if *obj* is an instance of class *Class* or is an instance of a subclass of Class

class Parent: # define parent class

def myMethod(self):

print 'Calling parent method'

class Child(Parent): # define child class

def myMethod(self):

print 'Calling child method'

c = Child() # instance of child

c.myMethod() # child calls overridden method

When the above code is executed, it produces the following result −

Calling child method