Practical Learning #6
Python Classes/Objects

Overview of OOP Terminology

- **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.

Creating Classes

The *class* statement creates a new class definition. The name of the class immediately follows the keyword *class* followed by a colon as follows –

```
class ClassName:
   'Optional class documentation string'
   class_suite
```

- The class has a documentation string, which can be accessed via *ClassName.__doc__*.
- The *class_suite* consists of all the component statements defining class members, data attributes and functions.

1. Create a new python file named PL6_LastnameFirstname
2. Code the following:

```
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)
```

- The variable *empCount* is a class variable whose value is shared among all instances of a this class. This can be accessed as *Employee.empCount* from inside the class or outside the class.
- The first method *__init__()* is a special method, which is called class constructor or initialization method that Python calls when you create a new instance of this class.
- You declare other class methods like normal functions with the exception that the first argument to each method is *self*. Python adds the *self* argument to the list for you; you do not need to include it when you call the methods.

Creating Instance Objects

To create instances of a class, you call the class using class name and pass in whatever arguments its *__init__* method accepts.

Accessing Attributes

You access the object's attributes using the dot operator with object. Class variable would be accessed using class name

3. Add the following codes:

```
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)
```

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

```
Name :  Zara ,Salary:  2000
Name :  Manni ,Salary:  5000
Total Employee 2
```

4. You can modify or delete attributes of classes and objects at any time –

```
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)
```

```
emp1.name = "Zoraida"  # Modify 'name' attribute.
emp1.displayEmployee()
del emp1.name  # Delete 'name' attribute.
emp1.displayEmployee()
```

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

```
Name :  Zara , Salary:  2000
Name :  Manni , Salary:  5000
Total Employee 2
Name :  Zoraida , Salary:  2000
Traceback (most recent call last):
  File "C:/Users/Iya/AppData/Local/Programs/Python/Python37-32/PL6.py", line 27,
 in <module>
    emp1.displayEmployee()
  File "C:/Users/Iya/AppData/Local/Programs/Python/Python37-32/PL6.py", line 14,
 in displayEmployee
    print ("Name : ", self.name,  ", Salary: ", self.salary)
AttributeError: 'Employee' object has no attribute 'name'
```

Instead of using the normal statements to access attributes, you can use the following functions –

- 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.

```
hasattr(emp2, 'name')    # Returns true if 'age' attribute exists
getattr(emp2, 'name')    # Returns value of 'age' attribute
setattr(emp2, 'name', "Manilyn") # Set attribute 'age' at 8
delattr(emp2, 'name')    # Delete attribute 'age'
```

5. Upload your work.

Online Reference:
https://www.tutorialspoint.com/python/python_classes_objects.htm