Module 1

Introduction to everything

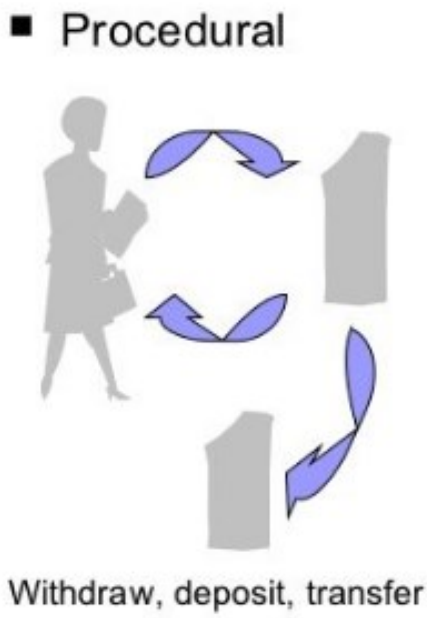
Section

Python programming: object-oriented programming

Object oriented programming

Main ideas in Object-Oriented Programming (OOPs) are -

- Class
- Objects
- Encapsulation
- Inheritance
- Polymorphism



Classes and objects

- Python, being an object-oriented programming language, considers everything as an object.
- An object has its properties and methods.
- A Class is like a blueprint. Classes represent real-world things and situations and these classes are used to construct objects.
- When we write a class, we specify the general behavior of the class, that will be exhibited by all the objects created from the class.
- Instantiation is the process by which an object is created from a class and it is with these instances of classes that we work with.

Classes are created by keyword class.

Attributes are the variables that belong to a class.

Attributes are always public and can be accessed using the dot (.) operator.

01

self represents the instance of the class.

02

Class methods must have an extra first parameter in the method definition. We don't have to pass a value for this parameter when we call the method.

03

By using the "self" keyword we can access the attributes and methods of the class in python. It binds the attributes with the given arguments.

04

This is similar to this pointer in C++ and this reference in Java.

Example Code:

```
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)
```

```
e1=Employee("Devi",100000)
e2=Employee("Nur",50000)
e3=Employee("Riya",80000)
e3.displayEmployee()
e3.displayCount()
```

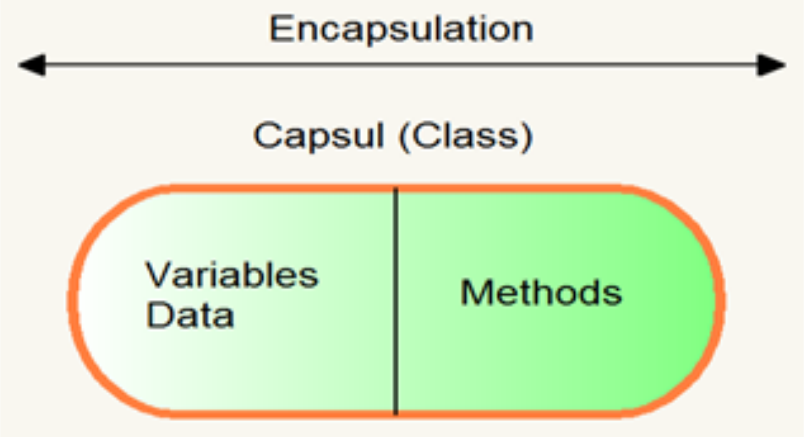
Output:

```
Name:  Riya, Salary:  80000
Total Employee:  6
```

The method **__init__() i**s a special method, which is called class constructor or initialization method that Python calls when you create a new instance of this class.

Encapsulation

Encapsulation refers to the idea of wrapping data and the methods that operate on data into one single unit. **Encapsulation** helps in isolating implementation details from clients that could expose hidden implementation details or violate state invariance maintained by the methods. A class is an example of encapsulation as it encapsulates all the data that is member functions, variables, etc.



Inheritance

Inheritance in object-oriented programming is a concept where one class can derive or inherit properties from another class. The class that derives properties is called the derived class and the class from which the properties are being derived is called the base class or parent class. The benefits of inheritance are-

- It provides the reusability of a code. We don't have to write the same code again and again. Also, it allows us to add more features to a class without modifying it.
- It is transitive in nature, which means that if class B inherits from another class A, then all the subclasses of B would automatically inherit from class A.

Example Code:

```
class Animal:
   def speak(self):
      print("Animal Speaking")

#child class Dog inherits the base class Animal
class Dog(Animal):
   def bark(self):
      print("dog barking")

d = Dog()
d.bark()
d.speak()
```

Output:

```
dog barking
Animal Speaking
```

Packages and modules in python

In Python, modules are simple files saved with .py extension. It contains collections of functions and global variables. It is an executable file. A module allows you to logically organize your Python code. Grouping related code into a module makes the code easier to understand and use. You can use any Python source file as a module by executing an import statement in some other Python source file. Python common modules are Numpy, Scipy, Matplotlib etc. A package is a simple directory having collections of modules. For example, Pandas is a python package.

Python has many built-in functions, additionally many functions are available as part of libraries bundled with python distributions, In python these are known as built-in modules

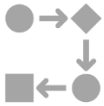
Majority of python modules are written in C language and integrated with python shell

To seea list of all available modules, we can use command **help('modules')**

Advantages of modules in python

Reusability

Using python modules, improves reusability of code in python, same functions/class implementations can be imported in multiple codes.



Simplicity

The module focuses on a small proportion of the problem, rather than focusing on the entire problem.

Scoping

A separate namespace is defined by a module that helps to avoid collisions between identifiers.

Example Code:

```
# This piece of code is written in demo.py file

def callme():
   print("Module Demo Called")

import demo
demo.callme()
```

Output:

```
Module Demo Called
```

Summary

- Object oriented programming enables modular and structural approach of programming which in turn improves code simplicity, eases out code maintenance
- Python modules are .py files containing class implementations, attributes, and functions, can be loaded using import keyword
- Python has many useful built-in modules such as math, os, time, statistics etc.