Chapter 2: Object Oriented Programming in Python

**1. What are different OOP terminology used in python?**

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.

**2. Write a program to demonstrate use of class and object**

class rectangle:

def accept(self):

self.length=int(input('Enter length='))

self.breadth=int(input('Enter breadth='))

def area(self):

a=self.length\*self.breadth

print('Area=',a)

r1=rectangle()

r2=rectangle()

r1.accept()

r1.area()

r2.accept()

r2.area()

**3. Explain \_\_init\_\_() function with example**

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:

class rectangle:

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

self.length=int(input('Enter length='))

self.breadth=int(input('Enter breadth='))

def area(self):

a=self.length\*self.breadth

print('Area=',a)

r1=rectangle()

r2=rectangle()

r1.area()

r2.area()

**4. Explain use of ‘self’ parameter**

The self parameter is a reference to the current instance of the class, 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:

class Person:

def \_\_init\_\_(mysillyobject, name, age):

mysillyobject.name = name

mysillyobject.age = age

def myfunc(abc):

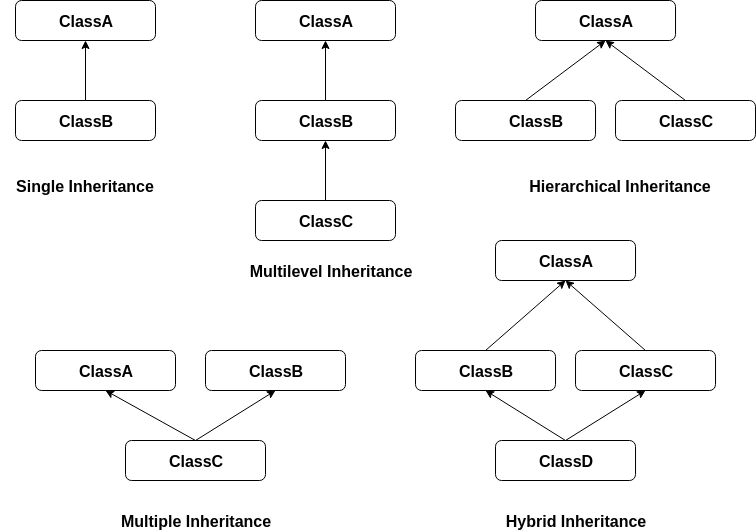
print("Hello my name is " + abc.name)

p1 = Person("John", 36)

p1.myfunc()

**5. Explain inheritance in python**

Inheritance is the capability of one class to derive or inherit the properties from another class. It provides reusability of a code.



Python also has a super() function that will make the child class inherit all the methods and properties from its parent

Multiple functions having same name and same parameter, but one is in base class and another is in derived class, is called as Method Overriding.

When we add the \_\_init\_\_() function in a parent class, the child class will no longer be able to inherit the parent class’s \_\_init\_\_() function. The child’s class \_\_init\_\_() function overrides the parent class’s \_\_init\_\_() function.

We can make some of the instance variables of the parent class private, which won’t be available to the child class. We can make an instance variable by adding double underscores before its name.

Example: self.\_\_d #Here d is private member

**6. Write a program to demonstrate single inheritance**

class person:

def \_\_init\_\_(self,a,b):

self.id=a

self.name=b

def display(self):

print('ID=',self.id)

print('Name=',self.name)

class employee(person):

def \_\_init\_\_(self,a,b,c):

person.\_\_init\_\_(self,a,b) #super().\_\_init\_\_(a,b)

self.salary=c

def display(self):

person.display(self) #super().display()

print('Salary=',self.salary)

e=employee(1,'AAA',2000)

e.display()

**7. Write a program to demonstrate multilevel inheritance**

class person:

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

self.id=int(input('Enter ID='))

self.name=input('Enter name=')

def display(self):

print('ID=',self.id)

print('Name=',self.name)

class employee(person):

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

super().\_\_init\_\_()

self.department=input('Enter department=')

def display(self):

super().display()

print('Department=',self.department)

class manager(employee):

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

super().\_\_init\_\_()

self.salary=int(input('Enter Salary='))

def display(self):

super().display()

print('Salary=',self.salary)

m=manager()

m.display()

**8. Write a program to demonstrate hierarchical inheritance**

class parent:

def f1(self):

print('I am in parent class')

class child1(parent):

def f2(self):

print('I am in child1')

class child2(parent):

def f3(self):

print('I am in child2')

ob1=child1()

ob2=child2()

ob1.f1()

ob1.f2()

ob2.f1()

ob2.f3()

**9. Write a program to demonstrate multiple inheritance**

class base1:

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

print('I am in base1')

class base2:

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

print('I am in base2')

class derived(base1,base2):

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

super().\_\_init\_\_() #base1.\_\_init\_\_(self)

#base2.\_\_init\_\_(self)

print('I am in derived')

ob=derived()

**10. Write a program to demonstrate hybrid inheritance**

class stud:

def setstud(self,sno,sname):

self.sno = sno;

self.sname = sname;

def putstud(self):

print("Student No : " ,self.sno);

print("Student Name : ",self.sname);

class marks(stud):

def setmarks(self, m1,m2):

self.mark1 = m1;

self.mark2 = m2;

def putmarks(self):

print("Mark1 : " , self.mark1);

print("Mark2 : " , self.mark2);

class pratical:

def getpractial(self,p1):

self.p1=p1;

def putpractial(self):

print("Practial mark=",self.p1);

class result(marks,pratical):

def calc(self):

self.total = self.mark1 + self.mark2+self.p1;

def puttotal(self):

print("Total : " , self.total);

r =result();

r.setstud(60,"Ash");

r.setmarks(50,60);

r.getpractial(100);

r.calc();

r.putstud();

r.putmarks();

r.putpractial()

r.puttotal();