In [8]:

```
#Q1.Implement a basic python program to demonstrate class, object and method.
class person:
    name='Cristiano'
    age=38

    def display(self):
                print("The Name of Person is: ",self.name)
                print("The Age of Person: ",self.age)
p=person()
p.display()
```

```
The Name of Person is:  Cristiano
The Age of Person:  38
```

In [19]:

```
#Q2.Implement a python program to use built in attributes.
class person:
    name= 'Sam'
    age=20

    def display(self):
        print( "The Name of Person:",self.name)
        print("The Age of Person:",self.age)

p=person()
print(getattr(p,'name'))
print(hasattr(p,'age'))
setattr(p,'name','Neel')
print("Person name After modifying:",p.name)
```

```
Sam
True
Person name After modifying: Neel
```

In [27]:

```
#3. Implement a python program to use built in class attributes.
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 :",Employee.empCount)

   def displayEmployee(self):
      print ("Name : ", self.name, ", Salary: ", self.salary)

print("\nEmployee __doc__ :", Employee.__doc__)
print("\nEmployee __name__ :", Employee.__name__)
print("\nEmployee __module__:", Employee.__module__)
print("\nEmployee __dict__ :", Employee.__dict__)
```

```

Employee __doc__ : Common base class for all employees

Employee __name__ : Employee

Employee __module__: __main__

Employee __dict__ : {'__module__': '__main__', '__doc__': 'Common base class
for all employees', 'empCount': 0, '__init__': <function Employee.__init__ a
t 0x000001B86CA017E0>, 'displayCount': <function Employee.displayCount at 0x
000001B86CA02170>, 'displayEmployee': <function Employee.displayEmployee at
0x000001B86CA01EA0>, '__dict__': <attribute '__dict__' of 'Employee' objects
>, '__weakref__': <attribute '__weakref__' of 'Employee' objects>}
```

In [28]:

```
#4. Implement a python program to demonstrate destructor in python.
class Employee:
    def __init__(self):
        print('Employee created.')
    def __del__(self):
        print('Destructor called, Employee deleted.')

obj = Employee()
del obj
```

```
Employee created.
Destructor called, Employee deleted.
```

In [29]:

```
#5.Implement a python program to find area of a rectangle using classes and objects.
class rectangle():
    def __init__(self,breadth,length):
        self.breadth=breadth
        self.length=length
    def area(self):
        return self.breadth*self.length
a=int(input("Enter length of rectangle: "))
b=int(input("Enter breadth of rectangle: "))
obj=rectangle(a,b)
print("Area of rectangle:",obj.area())

print()
```

```
Enter length of rectangle: 5
Enter breadth of rectangle: 5
Area of rectangle: 25
```

In [1]:

```
#6. Define a class to represent a bank account. Include the following details like name of
class Bank_Account:


    def new(self):
        self.name=input("Enter the Name of customer:")
        self.acct=input("Enter the Type of the Account:")
        self.accno=input("Enter the Account Number:")
        self.bal=int(input("Enter the Balance:"))
    def deposit(self):
        amount=float(input("Enter amount to be Deposited:"))
        self.bal += amount
        print("\n Amount Deposited:",amount,"\n\n")
    def withdraw(self):
        amount = float(input("Enter amount to be Withdrawn:"))
        if self.bal>=amount:
                self.bal-=amount
                print("\n You Withdrew:", amount,"\n\n")
        else:
            print("\n Insufficient balance \n\n")
    def display(self):
        print("\n Net Available Balance=",self.bal,"\n\n")
s = Bank_Account()
choice=0
while choice!=5:
    print("Mini Bank\n")
    print("1. New Customer")
    print("2. Deposit Amount")
    print("3. Withdraw Amount")
    print("4. Display Balance")
    print("5. Exit")
    choice=int(input("\nEnter ur Option:"))
    if choice==1:
        s.new()
    if choice==2:
        s.deposit()
    if choice==3:
        s.withdraw()
    if choice==4:
        s.display()
    if choice==5:
        print("EXIT")
```

```
Mini Bank

1. New Customer
2. Deposit Amount
3. Withdraw Amount
4. Display Balance
5. Exit

Enter ur Option:1
Enter the Name of customer:Sumit Kamble
Enter the Type of the Account:Saving
Enter the Account Number:128783039
Enter the Balance:1000
```

```
Mini Bank

1. New Customer
2. Deposit Amount
3. Withdraw Amount
4. Display Balance
5. Exit

Enter ur Option:2
Enter amount to be Deposited:500

 Amount Deposited: 500.0


Mini Bank

1. New Customer
2. Deposit Amount
3. Withdraw Amount
4. Display Balance
5. Exit

Enter ur Option:3
Enter amount to be Withdrawn:100

 You Withdrew: 100.0


Mini Bank

1. New Customer
2. Deposit Amount
3. Withdraw Amount
4. Display Balance
5. Exit

Enter ur Option:4

 Net Available Balance= 1400.0


Mini Bank

1. New Customer
2. Deposit Amount
3. Withdraw Amount
4. Display Balance
5. Exit

Enter ur Option:5
EXIT
```

In [3]:

```
#Problem Statement:- B] Implement programs based on Inheritance and Polymorphism.
```

In [5]:

```
class Person:

    def setPerson(self,name,code):
        self.name=name
        self.code=code

class Account(Person):
    def setAccount(self,pay):
        self.pay=pay

class Admin(Person):
    def setAdmin(self,exp):
        self.exp=exp
class Employee(Account,Admin):
    def data(self):
        print("\nEmployee Name: ",self.name)
        print("Employee Code: " ,self.code)
        print("Employee Salary: ",self.pay)
        print("Employee Experience: ",self.exp)

e=Employee()
e.setPerson("CR7",37)
e.setAccount(5000)
e.setAdmin(5)
e.data()
```

```

Employee Name:  CR7
Employee Code:  37
Employee Salary:  5000
Employee Experience:  5
```

In [ ]:

```
Create a class called Staff with code and name. Create classes Teacher (subject, publicatio
Typist(speed), Officer(grade). Using the typist class as base clas! classes Regular(salary)
Casual (daily wages). Implement a menu driven prograc the same.
```

In [11]:

```
class Staff:
    def setstaff(self,name,code):
        self.name=name
        self.code=code
        print("\nName = ",self.name)
class Teacher(Staff):
    def setTeacher(self,sub,pub):
        self.sub=sub
        self.pub=pub
        print("\nSubject = ",self.sub)


class Typist(Staff):
    def setTypist(self,spd):
        self.spd=spd
        print("\nSpeed= ",self.spd)

class Officer(Staff):
    def setOfficer(self,grade):
        self.grade=grade
        print("\nGrade= ",self.grade)

class Regular(Typist):
    def setRegular(self,sal):
        self.sal=sal
        print("\nSalary =",self.sal)


class Casual(Typist):
    def setCasual(self,wages):
        self.wages=wages
        print("\nDaily Wages:",self.wages)

t = Teacher()
o = Officer()
r = Regular()
c = Casual()
choice=0
while choice!=5:
    print("\n\n1. Teacher Info")
    print("2. Officer Info")
    print("3. Regular Salary Info")
    print("4. Display Wages")
    print("5. Exit")
    choice=int(input("\nEnter ur Option:"))
    if choice==1:
        t.setstaff('abc',1)
        t.setTeacher('Python','JavaTpoint')
    if choice==2:
        o.setstaff('asd',2)
        o.setOfficer('A')
    if choice==3:
        r.setTypist(200)
        r.setRegular(5000)
    if choice==4:
        c.setTypist(600)
        c.setCasual(500)
    if choice==5:
        print('EXIT')
```

```
1. Teacher Info
2. Officer Info
3. Regular Salary Info
4. Display Wages
5. Exit

Enter ur Option:1

Name =  abc

Subject =  Python


1. Teacher Info
2. Officer Info
3. Regular Salary Info
4. Display Wages
5. Exit

Enter ur Option:2

Name =  asd

Grade=  A


1. Teacher Info
2. Officer Info
3. Regular Salary Info
4. Display Wages
5. Exit

Enter ur Option:3

Speed=  200

Salary = 5000


1. Teacher Info
2. Officer Info
3. Regular Salary Info
4. Display Wages
5. Exit

Enter ur Option:4

Speed=  600

Daily Wages: 500


1. Teacher Info
2. Officer Info
```

```
3. Regular Salary Info
4. Display Wages
5. Exit

Enter ur Option:5
EXIT
```

In [5]:

```
class A:
    def __init__ (self, a):
        self.a=a

    def __add__(self, o):
        return self.a + o.a
ob1 = A(1)
ob2 = A(2)
ob3 = A("\nSumit ")
ob4 = A("Kamble\n")

print(ob1 + ob2)
print(ob3 + ob4)
print("Addition of Numbers:",20 + 22)

# concatenate two strings
print("\nWelcome To "+"Sanjay Ghodawat University\n")

# Product two numbers
print("3*4 : ",3 * 4,"\n")

# Repeat the Stn!ng
print ( "Happy " *4)
```

```
3

Sumit Kamble

Addition of Numbers: 42

Welcome To Sanjay Ghodawat University

3*4 :  12

Happy Happy Happy Happy
```

In [ ]:

```

```