In [ ]:
```
#creating a class
class first:
  pass
obj1=first()
print(obj1)
```

```
<__main__.first object at 0x7f3e8a73fad0>
```

In [ ]:
```
class add1:
  def addition(self,a,b):
    self.a=a
    self.b=b
    self.sum=self.a+self.b
    print("afterr addition",self.sum)
obj1=add1()
obj1.addition(2,4)
```

```
afterr addition 6
```

In [ ]:
```
#simple class
class computer:
    def features(self):
       print("this is a new model system")
comp1=computer()
comp1.features()
computer.features(comp1)
```

```
this is a new model system
this is a new model system
```

In [ ]:
```
class student:
  pass
s1=student()
s2=student()
s1.name='vidhya'
s2.name='ann'
print(s1.name)
```

```
vidhya
```

In [ ]:
```
#init method
class computer:
    def __init__(self):
       print("you are in init method")
    def features(self):
       print("this is a new model system")
comp1=computer()
comp1.features()
computer.features(comp1)
```

```
you are in init method
this is a new model system
this is a new model system
```

In [1]:
```
#instance variable
class computer:

    def features(self):
        self.name="mary"
        print(self.name)
    def nwfeatures(self):

        print(self.name)
comp1=computer()
comp1.features()
comp1.nwfeatures()
```

```
mary
mary
```

In [ ]:
```
#passing variabls to method
class computer:
    def __init__(self,cpu,ram):
       self.processor=cpu
       self.memory=ram
    def features(self):
       print("this is a new model system")
       print(self.processor,self.memory)
comp1=computer("i5","16gb")
comp2=computer("i4","18gb")

comp2.features()
```

```
this is a new model system
i4 18gb
```

In [ ]:
```
class computer:
      def features(self,processor,ram):
       self.processor=processor
       self.ram=ram
       print("this is a new model system")
       print(self.processor,self.ram)
      def new(self):
        print(self.processor)
comp1=computer()
comp1.features("i5","16gb")
comp1.new()
```

```
this is a new model system
i5 16gb
i5
```

In [9]:
```
#instance and class variables
class Person:
  def __init__(self, name):
    self.name = name
class Employee(Person):
  def isEmployee(self):
    return True
  def isEmployee(self):
    return False
  def getName(self):
    return self.name

e = Employee("Ammu")
print(e.getName(),  e.isEmployee())
p = Person("Anu")
print(p.getName(),  p.isEmployee())
```

```
Ammu False

---------------------------------------------------------------------------
----
AttributeError                            Traceback (most recent call l
ast)
<ipython-input-9-f23ee7e2f6be> in <module>()
     14 print(e.getName(),  e.isEmployee())
     15 p = Person("Anu")
---> 16 print(p.getName(),  p.isEmployee())

AttributeError: 'Person' object has no attribute 'getName'
```

In [ ]:
```
#display complex numbers
class complex1:
    def __init__(self,i,j):
       self.real=i
       self.imaginary=j
    def number(self):
       print("{}+{}j".format(self.real,self.imaginary))
comp1=complex1(5,6)
comp1.number()
```

```
5+6j
```

In [ ]:
```
#add two complex numbers
class complexnw:
    def __init__(self,i,j):
       self.real=i
       self.imaginary=j
    def add(self,obj):
       print(self.real+obj.real)
       print(self.imaginary+obj.imaginary)

complex1=complexnw(5,6)

complex2=complex(7,8)
complex1.add(complex2)
```

```
12
14
```

In [ ]:
```
class Rectangle:
    def __init__(self,length=0,breadth=0):
        self.length=length
        self.breadth=breadth
    def area(self):
        print("area=",self.length*self.breadth)
R1=Rectangle(10,20)
R1.area()
R2=Rectangle(12,13)
R2.area()
R3=Rectangle()
R3.area()
```

```
area= 200
area= 156
area= 0
```

In [ ]:
```
#instance variable and class variable
class Rectangle:
    perimeter=15
    def __init__(self,length=0,breadth=0):
        self.length=length
        self.breadth=breadth
    def area(self):
        print("area=",self.length*self.breadth)
R1=Rectangle(10,20)
R2=Rectangle()
print(R1.length)

print(R2.length)
print(R1.perimeter)

print(R2.perimeter)
Rectangle.perimeter=13
print(R1.perimeter)

print(R2.perimeter)
```

```
10
0
15
15
13
13
```

In [14]:
```
class Rectangle:

    def __init__(self,length=0,breadth=0):
        self.length=length
        self.breadth=breadth
    def area(self):
        print("area=",self.length)
    def classmethod(cls):
        print("this is a class method", self.breadth)
R1=Rectangle(10,20)
R2=Rectangle()
print(R1.length)
```

```
10
```

In [ ]:
```
#Create a class car with attributes model,
 #year and price and a method cost() for displaying the prize.
 #Create two instance of the class and call the method for each instan
ce.(university question)"""
class Car:
     def __init__(self,model,year,prize):
        self.model=model
        self.year=year
        self.prize=prize
     def cost(self):
        print ("Prize of the car=",self.prize)
C1=Car("Maruti",2004,200000)
C2=Car("Ford",2014,5000000)
C1.cost()
C2.cost()
```

```
Prize of the car= 200000
Prize of the car= 5000000
```

In [ ]:
```
#Create a class student with attribute #name and roll number and a
#method dataprint() for displaying the same.
#Create two instance of the class and call the method for each instanc
e.( university question)
class Student:
    def __init__(self,name,rno):
        self.name=name
        self.rno=rno
    def dataprint(self):
        print ("Name=",self.name)
        print ("Rno=",self.rno)
s1=Student("devi",101)
s2=Student("anjana",102)
sq1.dataprint()
s2.dataprint()
```

```
Name= devi
Rno= 101
Name= anjana
Rno= 102
```

In [10]:
```
#Define a class in Python to
#store the details of students( rollno, mark1,mark2)
#with the following methods
#readData()- to assign values to class attributes
#computeTotal()-to find the total marks
#printDetails()- to print the attribute values and total marks.
#Create an object of this class and invoke the methods. ( Univesrsity
 question)
class Student:
    def readData(self):
        self.rollno=input("enter roll number...")
        self.mark1=int(input("enter mark1.."))
        self.mark2=int(input("enter mark2.."))
    def computeTotal(self):
        self.total=self.mark1+self.mark2
    def printDetails(self):
        print ("roll number-->",self.rollno)
        print ("Mark1-------->",self.mark1)
        print( "Mark2-------->",self.mark2)
        print( "Total Marks---",self.total)
S=Student()
S.readData()
S.computeTotal()
S.printDetails()
```

```
enter roll number...2
enter mark1..34
enter mark2..33
roll number--> 2
Mark1--------> 34
Mark2--------> 33
Total Marks--- 67
```

In [17]:
```
#mutator and accessor
class Fruit:
    def   __init__(self, name):
       self.name = name

    def setFruitName(self, name):
       self.name = name

    def getFruitName(self):
        return self.name

f1 = Fruit("Apple")
print("First fruit name: ", f1.getFruitName())
f1.setFruitName("Grape")
print("Second fruit name: ", f1.getFruitName())
```

```
First fruit name:  Apple
Second fruit name:  Grape
```

In [18]:
```
class Student:
    def __init__(self, name):
      print("This is parameterized constructor")
      self.name = name
    def show(self):
       print("Hello ", self.name)
s1 = Student("Rahul")
s1.show()
```

```
This is parameterized constructor
Hello  Rahul
```

In [ ]: