1. Write a program illustrating class definition and accessing class members.

In [15]:
```
class Myclass:
    cl_var=100   # cl_var is class variable.
    def myfun(self,obx):  # instance method of class My
        self.obx = obx    # obx is the object variable
    def display(self):
        print("Value =",self.obx)
# Decalartion of object of class Myclass
myobj = Myclass()
# Accessing class members---using dot operator
print('Using class name ',Myclass.cl_var)
print('Using object name ',myobj.cl_var)
Myclass.cl_var = 200
print('After Updation using class name ',Myclass.cl_var)
print('After Updation using object name ',myobj.cl_var)
val = int(input('Enter some value '))
# Calling instance method using object myobj
myobj.myfun(val)
myobj.display()
# Calling instance method using class name
Myclass.myfun(myobj,val+100)
Myclass.display(myobj)
```

```
Using class name  100
Using object name  100
After Updation using class name  200
After Updation using object name  200
Enter some value 300
Value = 300
Value = 400
```

2. Write a program to implement default constructor, parameterized constructor, and destructor.

Constructor

- It is the instance method used to construct the members of the object.
- Special method or magic method $__init__()$ is called the constructor method.

In [16]:
```
# Example 1: Default constructor.
# It is the constructor without any argument other than self.
class Myclass:
    def __init__(self):
        print('Inside default consturtor.')
myobj = Myclass()
```

```
Inside default consturtor.
```

In [17]:
```
# Example 2: Parameterized constructor.
# It is the constructor with arguments other than self.
class Student:
    def __init__(self,rollno,name):
        print('Inside Parameterized consturtor.')
        self.rollno = rollno
        self.name = name
    def displayStudent(self):
        print('Roll Number =',self.rollno)
        print('Student Name =',self.name)
r = int(input('Enter roll number '))
n = input('Enter Name ')
s1 = Student(r,n)
s1.displayStudent()
```

```
Enter roll number 101
Enter Name Aman
Inside Parameterized consturtor.
Roll Number = 101
Student Name = Aman
```

In [18]:

```
# Example 3: Destructor.
class Myclass:
    def __init__(self):
        print('Inside default consturtor.')
    def __del__(self):
        print('Destructing object.')
myobj = Myclass()
```

```
Inside default consturtor.
```

3. Create a Python class named Rectangle constructed by a length and width.
a) Create a method called area which will compute the area of a rectangle.

In [2]:

```
class Rectangle:
    def __init__(self,l,w):
        self.length=l
        self.width=w
    def area(self):
        return self.length*self.width
l = int(input('Enter length '))
w = int(input('Enter width '))
rect = Rectangle(l,w)
print('Area = ',rect.area())
```

```
Enter length 5
Enter width 4
Area =  20
```

4. Create a class called Numbers, which has a single class attribute called MULTIPLIER, and a constructor which takes the parameters x and y (these should all be numbers).
 a) Write an instance method called add which returns the sum of the attributes x and y.
 b) Write a class method called multiply, which takes a single number parameter a and returns the product of a and MULTIPLIER.
 c) Write a static method called subtract, which takes two number parameters, b and c, and returns b - c.
 d) Write a method called value which returns a tuple containing the values of x and y.

In [1]:

```
# Program 4.
class Numbers:
    MULTIPLIER = 10   # class attributes
    def __init__(self,x,y):
        self.x = x
        self.y = y
    def add(self):
        return self.x + self.y
    @classmethod
    def multiply(cls,a):
        return a*cls.MULTIPLIER
    @staticmethod
    def subtract(b,c):
        return b-c
    def value(self):
        return self.x,self.y
#driver code
m = int(input("Enter multiplier "))
Numbers.MULTIPLIER = m

f1 = int(input("Enter first parameter "))
f2 = int(input("Enter second parameter "))
N = Numbers(f1,f2)
print("SUM :",N.add())    # Numbers.add(N)

a = int(input("Enter any number "))
print("Product :",Numbers.multiply(a))

b = int(input("Enter value of b "))
c = int(input("Enter value of c "))
print("b-c =",Numbers.subtract(b,c))
val = N.value()
print("Pair :",val)
```

```
Enter multiplier 100
Enter first parameter 2
Enter second parameter 4
SUM : 6
Enter any number 5
Product : 500
Enter value of b 6
Enter value of c 3
b-c = 3
Pair : (2, 4)
```

5. Create a class named as Student to store the rollno, name and marks in three subjects. Use List to store the marks.
a) Write an instance method called compute() to compute total marks and average marks of a student.
b) Write a method called display() to display student information.

In [3]:
```
class Student:
    def __init__(self,rollno,name):
        self.rollno=rollno
        self.name=name
        self.marks=[]
    def setmarks(self):
        m = input('Enter marks of three subjects ').split()  #'20 30 25'.....['20','30'
        marks = [int(i) for i in m]
        self.marks = marks
    def compute(self):
        sum1 = 0
        for m1 in self.marks:     # self.marks=[20,30,25]
            sum1 = sum1 + m1
        print('Total Marks =',sum1)
        print('Average Marks =',sum1/len(self.marks))
    def display(self):
        print('Roll Number =',self.rollno)
        print('Name =',self.name)
        print('Marks =',self.marks)
r = int(input('Enter roll Number '))
n = input('Enter Name ')
s1 = Student(r,n)
s1.setmarks()
s1.compute()
s1.display()
```

```
Enter roll Number 101
Enter Name Aman
Enter marks of three subjects 12 34 56
Total Marks = 102
Average Marks = 34.0
Roll Number = 101
Name = Aman
Marks = [12, 34, 56]
```

6. Create a class Employee that keeps a track of the number of employees in an organization and also stores their name, designation and salary details.
 a) Write a method called getdata to take input (name, designation, salary) from user.
 b) Write a method called average to find average salary of all the employees in the organization.
 c) Write a method called display to print all the information of an employee.

In [3]:
```
class Employee:
    count = 0
    def __init__(self):
        Employee.count = Employee.count + 1
    def getdata(self):
        name = input("Enter name ")
        designation = input("Enter Designation ")
        salary = int(input("Enter salary "))
        self.name = name
        self.designation = designation
        self.salary = salary
    def display(self):
        print("Name :",self.name)
        print("Designation :",self.designation)
        print("Salary :",self.salary)
        print("Number of employees :",Employee.count)
    @staticmethod
    def average(mylist):
        sum1 = 0
        for e in mylist:
            sum1 = sum1 + e.salary
        return sum1/Employee.count
Emplist = []
num = int(input("Enter number of employees ")) #num=3
for i in range(num):  #i=0,1,2
```

```
    E = Employee()
    Emplist.append(E)     # List of objects
    E.getdata()
for e in Emplist:
    e.display()
print("Average Salary :",Employee.average(Emplist))
```

```
Enter number of employees 3
Enter name ABC
Enter Designation SW DEV
Enter salary 1000
Enter name DEF
Enter Designation TL
Enter salary 1500
Enter name WQR
Enter Designation EXE
Enter salary 500
Name : ABC
Designation : SW DEV
Salary : 1000
Number of employees : 3
Name : DEF
Designation : TL
Salary : 1500
Number of employees : 3
Name : WQR
Designation : EXE
Salary : 500
Number of employees : 3
Average Salary : 1000.0
```

7. Create a Python class named Circle constructed by a radius. Use a class variable to define the value of constant PI.
a) Write two methods to be named as area and circumference to compute the area and the circumference of a circle respectively by using class variable PI.
c) Write a method called display to print area and perimeter.

In [4]:
```
class Circle:
    __PI = 3.1415
    def __init__(self,radius):
        self.radius = radius
    def area(self):
        return Circle.__PI*self.radius*self.radius
    def circumference(self):
        return 2*Circle.__PI*self.radius
    def display(self):
        print("Area =",round(self.area(),2))
        print("Perimeter =",round(self.circumference(),2))
r = float(input("Enter radius of circle "))
C = Circle(r)
C.display()
```

```
Enter radius of circle 5
Area = 78.54
Perimeter = 31.42
```

8. Create a class called String that stores a string and all its status details such as number of uppercase letters, lowercase letters, vowels ,consonants and space in instance variables.

a) Write methods named as count_uppercase, count_lowercase, count_vowels, count_consonants and count_space to count corresponding values.
b) Write a method called display to print string along with all the values computed by methods in (a).

In [1]:
```
class String:
    def __init__(self):
        self.uppercase = 0
        self.lowercase = 0
        self.vowels = 0
        self.consonants = 0
        self.space = 0
        msg = input("Enter message ")
        self.msg = msg
    def count_uppercase(self):
        for letter in self.msg:   # msg = 'Hello Students'
            if letter.isupper():
                self.uppercase += 1   #self.uppercase = self.uppercase + 1
    def count_lowercase(self):
        for letter in self.msg:
            if letter.islower():
                self.lowercase += 1
    def count_vowels(self):
        for letter in self.msg: # msg='Hello Students'
            if letter in 'aeiouAEIOU':
                self.vowels += 1
    def count_consonants(self):
        for letter in self.msg:  # msg='Hello @ students'
            if 'a'<=letter<='z' or 'A'<=letter<='Z':
                if letter not in 'aeiouAEIOU':
                    self.consonants += 1
    def count_space(self):
        for letter in self.msg:
            if letter == ' ':
                self.space += 1
    def display(self):
        print("Message")
        print(self.msg)
        print("Uppercase letters :",self.uppercase)
        print("Lowercase letters :",self.lowercase)
        print("Vowels :",self.vowels)
        print("Consonants :",self.consonants)
        print("Spaces :",self.space)
S = String()
S.count_uppercase()
S.count_lowercase()
S.count_vowels()
S.count_consonants()
S.count_space()
S.display()
```

```
Enter message Hello Students
Message
Hello Students
Uppercase letters : 2
Lowercase letters : 11
Vowels : 4
Consonants : 9
Spaces : 1
```

9. Write a program that has a class called Fraction with attributes numerator and denominator.

a) Write a method called getdata to enter the values of the attributes.

b) Write a method show to print the fraction in simplified form.

In [2]:
```
def gcd(a,b):
    if b == 0:
        return a
    else:
        return gcd(b,a%b)
class Fraction:
    def __init__(self,N=None,D=None):
        self.N = N
        self.D = D
    def getdata(self):
        N = int(input("Enter Numerator "))
        D = int(input("Enter Denominator "))
        self.N = N
        self.D = D
    def show(self):
        g = gcd(self.N,self.D)
        self.N = self.N//g
        self.D = self.D//g
        print(self.N,"/",self.D)
F1 = Fraction()
F1.getdata()
F1.show()
```

```
Enter Numerator 12
Enter Denominator 18
2 / 3
```

10. Write a program that has a class Numbers with a list as an instance variable.

a) Write a method called insert_element that takes values from user.

b) Write a class method called find_max to find and print largest value in the list.

In [4]:
```
class Numbers:
    def __init__(self):
        self.list1 = []
    def insert_element(self):
        val = int(input('Enter value '))
        self.list1.append(val)
    def find_max(self):
        if len(self.list1)>0:
            max1 = self.list1[0]
            for i in range(1,len(self.list1)):
                if max1<self.list1[i]:
                    max1 = self.list1[i]
            return max1
        else:
            return 'Empty List!'
num = Numbers()
num.insert_element()
num.insert_element()
num.insert_element()
num.insert_element()
```

```
print(num.list1)
print('Maximum Value =',num.find_max())
```

```
Enter value 4
Enter value 19
Enter value 12
Enter value 7
[4, 19, 12, 7]
Maximum Value = 19
```

```
11. Write a program that has a class Point with attributes x and y.
a) Write a method called midpoint that returns a midpoint of a line
joining two points.
 b) Write a method called length that returns the length of a line
joining two points.
```

In [31]:

```
class Point:
    def __init__(self,x=0,y=0):
        self.x = x
        self.y = y
    def midpoint(self,p):
        temp=Point(0,0)
        temp.x = (self.x + p.x)/2
        temp.y = (self.y + p.y)/2
        return temp.x,temp.y
    def length(self,p):
        d = ((p.x-self.x)**2+(p.y-self.y)**2)**0.5
        return d
print('Enter coordinates of first point ')
x1 = int(input('Enter x-coordinate '))
y1 = int(input('Enter y-coordinate '))
p1 = Point(x1,y1)
print('Enter coordinates of second point ')
x2 = int(input('Enter x-coordinate '))
y2 = int(input('Enter y-coordinate '))
p2 = Point(x2,y2)
print('Mid point =',p1.midpoint(p2))
print('Distance =',p1.length(p2))
```

```
Enter coordinates of first point
Enter x-coordinate 0
Enter y-coordinate 0
Enter coordinates of second point
Enter x-coordinate 3
Enter y-coordinate 4
Mid point = (1.5, 2.0)
Distance = 5.0
```

```
12. Create a class called Complex. Write a menu driven program to read,
display, add and subtract two complex numbers by creating corresponding
instance methods.
```

In [1]:

```
class Complex:
    def __init__(self,r,i):
        self.real = r
        self.imag = i
    def read(self,r,i):
        self.real=r
        self.imag=i
    def display(self):
        if self.imag<=0:
            print(str(self.real)+str(self.imag)+'j')
        else:
            print(str(self.real)+'+'+str(self.imag)+'j')
    def add(self,c):
        temp = Complex(0,0)
        temp.real = self.real+c.real
        temp.imag = self.imag+c.imag
        return temp
    def subtract(self,c):
        temp = Complex(0,0)
        temp.real = self.real-c.real
        temp.imag = self.imag-c.imag
        return temp
while(True):
    print('Enter your Choice:')
    print('1. Read')
    print('2. Display')
    print('3. Add')
    print('4. Subtract')
    ch=input()
    if ch=='1':
        print('Enter first complex number ')
        r1 = int(input('Enter real part '))
        i1 = int(input('Enter imaginary part '))
        c1 = Complex(r1,i1)
        c1.display()
        print('Enter second complex number ')
        r2 = int(input('Enter real part '))
        i2 = int(input('Enter imaginary part '))
        c2 = Complex(r2,i2)
        c2.display()
    elif ch=='2':
        c1.display()
        c2.display()
    elif ch=='3':
        c3 = c1.add(c2)
        c3.display()
    elif ch=='4':
        c4 = c1.subtract(c2)
        c4.display()
    else:
        print('INVALID CHOICE!')
    print('Do want to continue (y/Y) ')
    ch1=input()
    if ch1!='y' and ch1!='Y':
        break
```

```
Enter your Choice:
1. Read
2. Display
3. Add
4. Subtract
2

---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-1-89fe6a0f4769> in <module>
     40         c2.display()
     41     elif ch=='2':
---> 42         c1.display()
```

```
         c2.display()
     elif ch=='3':

NameError: name 'c1' is not defined
```

13. Write a Program to illustrate the use of $__str__()$, $__repr__()$, and $__new__()$ methods.

In [35]:
```
# new magic method
class Myclass:
    def __new__(cls):
        print ("__new__ magic method is called")
        inst = object.__new__(cls)
        return inst
    def __init__(self):
        print ("__init__ magic method is called")
        self.name='Python'
myobj = Myclass()
print(myobj.name)
```

```
__new__ magic method is called
__init__ magic method is called
Python
```

In [40]:
```
class Employee:
    def __init__(self):
        self.name='Shreya'
        self.salary=60000
    def __str__(self):
        return 'Name = '  + self.name + ', Salary=₹'+str(self.salary)
    def __repr__(self):
        return 'Hello = '  + self.name
emp = Employee()
print(emp)
print(repr(emp))
```

```
Name = Shreya, Salary=₹60000
Hello = Shreya
```

14. Create a BankAccount class. Your class should support the following methods:
a) $__init__(self, account_no)$
b) deposit (self, amount)
c) withdraw (self, amount)
d) get_balance (self)

In [12]:
```
class BankAccount:
    def __init__(self,account_no):
        self.account_no = account_no
        self.balance = 0
    def deposit(self,amount):
        self.balance = self.balance + amount
    def withdraw(self,amount):
        if amount > self.balance:
            print("Alert! Insuffcient balance")
        else:
            self.balance = self.balance - amount
    def get_balance(self):
        print("Account Number :",self.account_no)
        print("Balance Amount :",self.balance)
a = int(input('Enter Account Number '))
c1 = BankAccount(a)
c1.get_balance()
amount = int(input('Enter amount to be deposited '))
c1.deposit(amount)
c1.get_balance()
amount = int(input('Enter amount to be withrawn '))
c1.withdraw(amount)
c1.get_balance()
```

```
Enter Account Number 101
Account Number : 101
Balance Amount : 0
Enter amount to be deposited 3000
Account Number : 101
Balance Amount : 3000
Enter amount to be withrawn 4000
Alert! Insuffcient balance
Account Number : 101
Balance Amount : 3000
```

```
15. Write a program to illustrate the use of following built-in methods:
a) hasattr(obj,attr)
b) getattr(object, attribute_name [, default])
c) setattr(object, name, value)
d) delattr(class_name, name)
```

In [32]:
```
class ABC:
    def __init__(self,var):
        self.var = var
    def __str__(self):
        return 'Value is ' + str(self.var)
obj = ABC(100)
print(obj)
print("Getting Attribute",getattr(obj,'var'))
print("Checking for attribute",hasattr(obj,'var'))
setattr(obj,'var',200)
print("After setting value",obj.var)
setattr(obj,'nvar',500)
print("After setting new attribute",obj.nvar)
delattr(obj,'var')
print("After deleting attribute",obj.var)
```

```
Value is 100
Getting Attribute 100
Checking for attribute True
After setting value 200
After setting new attribute 500
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-32-63f2853be222> in <module>
     13 print("After setting new attribute",obj.nvar)
     14 delattr(obj,'var')
---> 15 print("After deleting attribute",obj.var)

AttributeError: 'ABC' object has no attribute 'var'
```

In [ ]: