Finding Area of Triangle

```
class Triangle:
    def __init__(self): # constructor
        self.__base=0.0  # instance variable
        self.__height=0.0 # instance variable
    def set_base(self,b): # instance method
        self.__base=b
    def set_height(self,h): # instance method
        self.__height=h
    def find_area(self): # instance method
        area=self.__base*self.__height*0.5
        return area

def main():
    t1=Triangle()
    t1.set_base(1.5)
    t1.set_height(1.7)
    a=t1.find_area()
    print(f'Area of triangle is {a:.2f}')

main()
```

Example of representing Account object in Banking Application

```
class Account:
    def __init__(self,a,cn,bal):
        self.__accno=a
        self.__cname=cn
        self.__balance=bal
    def deposit(self,t):
        self.__balance=self.__balance+t
    def withdraw(self,t):
        if t>self.__balance:
            print("insuff balance")
        else:
            self.__balance=self.__balance-t
    def printAccount(self):
        print(f'AccountNo {self.__accno}')
        print(f'CustomerName {self.__cname}')
        print(f'Balance {self.__balance}')
```

```
def main():
    acc1=Account(101,"naresh",5000.0)
    acc1.printAccount()
    acc1.deposit(1000)
    acc1.printAccount()
    acc1.withdraw(2000)
    acc1.printAccount()
main()
```

Output:

```
======== RESTART: F:/python6pmaug/ooptest16.py ========
AccountNo 101
CustomerName naresh
Balance 5000.0
AccountNo 101
CustomerName naresh
Balance 6000.0
AccountNo 101
CustomerName naresh
Balance 4000.0
```

 # Example of reading details of n studentmarks and find result

```
class StudentMarks:
    def __init__(self):
        self.__rollno=None # I.V or O.L.V
        self.__sub1=None
        self.__sub2=None
    def setMarks(self,rno,s1,s2):
        self.__rollno=rno
        self.__sub1=s1
        self.__sub2=s2
    def findResult(self):
        result="pass" if self.__sub1>=40 and self.__sub2>=40 else "fail"
        print(f"'Rollno {self.__rollno}\t Subject1 {self.__sub1}\tSubject2
{self.__sub2}\tResult {result}'")

def main():
    marksList=[]
    n=int(input("Enter value of n"))
```

```
    for i in range(n):
        r=int(input("enter rollno"))
        s1=int(input("enter subject1"))
        s2=int(input("enter subject2"))
        sm=StudentMarks()
        sm.setMarks(r,s1,s2)
        marksList.append(sm)
    for s in marksList:
        s.findResult()

main()
```

Output

```
Enter value of n2
enter rollno101
enter subject160
enter subject270
enter rollno102
enter subject130
enter subject250
Rollno 101  Subject1 60      Subject2 70      Result pass
Rollno 102  Subject1 30      Subject2 50      Result fail
```

Example

```
class Matrix:
    def __init__(self):
        self.__l=[] # I.V or O.L.V
    def readMatrix(self):
        r=int(input("Enter how many rows"))
        c=int(input("Enter how many cols"))
        for i in range(r):
            row=[]
            for j in range(c):
                value=int(input("enter value"))
                row.append(value)
            self.__l.append(row)
    def printMatrix(self):
        for row in self.__l:
            for col in row:
                print(col,end=' ')
```

```
            print()

def main():
    matrix1=Matrix()
    matrix1.readMatrix()
    matrix1.printMatrix()

main()
```

Output:

```
Enter how many rows2
Enter how many cols2
enter value1
enter value2
enter value3
enter value4
1 2
3 4
```

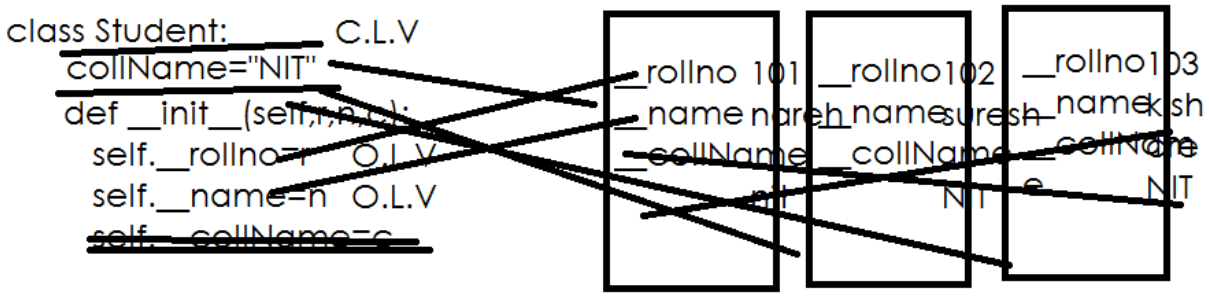
Class level variables

A variable declared inside class without self is called class level variable. Class level variables are global variables, which are global to one or more than one object.

These variables are used to define common properties belongs to objects. This variable is bind with class name and these variables can accessible without creating object.

Syntax:

```
class <class-name>:
        class-level-variables
        def method(self):
                self.variable=value → object level variable
                self.variable=value → object level variable
```




```
stud1=Student(101,"naresh","NIT")
stud2=Student(102,"suresh","NIT")
stud3=Student(103,"kishore","NIT")
```

Example

```
class A:
    x=100 # class level variable
    def __init__(self):
        self.y=200 # object level variable


def main():
    print(A.x)
    obj1=A()
    print(obj1.y)
    obj2=A()
    print(obj2.y)
    print(obj1.x)
    print(obj2.x)

main()
```

Output:

```
100
200
200
100
100
```

Example:

```
class Account:
    minBalance=5000 # C.L.V
    def __init__(self):
        self.__accno=None # OLV
        self.__cname=None # OLV
        self.__balance=None # OLV
    def setAccount(self,a,c,b):
        self.__accno=a
        self.__cname=c
        self.__balance=b
    def deposit(self,t):
        self.__balance=self.__balance+t
    def withdraw(self,t):
        if self.__balance-t<Account.minBalance:
            print("Insuff balance")
        else:
            self.__balance=self.__balance-t
    def printAccount(self):
        print(f'AccountNo {self.__accno}')
        print(f'Balance {self.__balance}')

def main():
    acc1=Account()
    acc1.setAccount(101,"naresh",10000)
    acc1.printAccount()
    acc1.deposit(5000)
    acc1.printAccount()
    acc1.withdraw(12000)

main()
```

Output:

```
AccountNo 101
Balance 10000
AccountNo 101
Balance 15000
insuff balance
```

class level method

A method defined inside class with first argument as “cls” is called class level method.
Class level method is used to perform class level operation.
Class level method is declared using @classmethod decorator
Class Level method access class level data/variables

Syntax:
@classmethod
def method-name(cls,arg1,arg2,arg3,…):
        statement-1
        statement-2

class level method is bind with class name. This method called without creating object.

Example:

```
class A:
    def m1(self):
        print("Object Level Method")
    @classmethod
    def m2(cls):
        print("Class Level Method")


A.m2()
obj1=A()
obj1.m1()
```

Output:

```
======== RESTART: F:/python6pmaug/ooptest22.py ========
Class Level Method
Object Level Method
```

Example:

```
import datetime
class Person:
    def __init__(self,n,a):
        self.__name=n
        self.__age=a
    def printPerson(self):
```

```
        print(f'Name :{self.__name} Age:{self.__age}')
    @classmethod
    def createPerson(cls,name,dob):
        cd=datetime.date.today()
        age=cd.year-dob.year
        p=Person(name,age)
        return p
def main():
    p1=Person("naresh",40)
    p1.printPerson()
    p2=Person.createPerson("suresh",datetime.date(2000,12,12))
    p2.printPerson()
main()
```

Output:

```
======== RESTART: F:/python6pmaug/ooptest23.py ========
Name :naresh Age:40
Name :suresh Age:22
```

Static method