Access Modifiers

Access modifiers define the accessibility of the members of the class.
Python provides 3 access modifiers

1. Private
2. Public
3. Protected

Private

If the members of the class are private, these members are accessible within class but cannot accessible outside the class.

Private members are defined by prefixing with __ (double) underscore.

class A:
  def m1(self): # public method
  print("m1 of A class")
  def __m2(self): # private instance method
  print("private m2 method")
 	
obja=A()
obja.m1()
obja.__m2()
	m1 of A class
	Traceback (most recent call last):
	  File "E:/python5pmjun/ooptest1.py", line 10, in <module>
	  obja.__m2()
	AttributeError: 'A' object has no attribute '__m2'. Did you mean: '_A__m2'?

class A:
  def __init__(self):
  self.__x=100
  self.y=200
 	
obj1=A()
print(obj1.__x)
print(obj1.y)
	Traceback (most recent call last):
	  File "E:/python5pmjun/ooptest2.py", line 9, in <module>
	  print(obj1.__x)
	AttributeError: 'A' object has no attribute '__x'

In python data hiding is achieved by declaring instance variables of class as private.

Private members are accessible within class but cannot accessible outside the class.

Example:

```
class List:
    def __init__(self):
        self.__data=None
    def append(self,value):
        self.__data=value
    def get(self):
        return self.__data



list1=List()
list1.append(10)
value=list1.get()
print(value)
```

Output

10

Example

```
class Customer:
    def __init__(self,a,cn,b):
        self.__accno=a
        self.__cname=cn
        self.__balance=b
    def print_account(self):
        print(f'AccountNo {self.__accno}')
        print(f'CustomerName {self.__cname}')
        print(f'Balance {self.__balance}')
    def deposit(self,tamt):
        self.__balance=self.__balance+tamt
    def withdraw(self,tamt):
        if tamt>self.__balance:
            print("Insuff balance")
```

```
        else:
            self.__balance=self.__balance-tamt



cust1=Customer(1,"naresh",5000)
cust1.print_account()
cust1.deposit(5000)
cust1.print_account()
cust1.withdraw(3000)
cust1.print_account()
```

Output

```
AccountNo 1
CustomerName naresh
Balance 5000
AccountNo 1
CustomerName naresh
Balance 10000
AccountNo 1
CustomerName naresh
Balance 7000
```

Example

```
class Marks:
    def __init__(self,r,n,s1,s2,s3):
        self.__rollno=r
        self.__name=n
        self.__sub1=s1
        self.__sub2=s2
        self.__sub3=s3
    def find_result(self):
        if self.__sub1<40 or self.__sub2<40 or self.__sub3<40:
            result="fail"
        else:
            result="pass"
        print(f'{self.__rollno},{self.__name},{self.__sub1},
{self.__sub2},{self.__sub3},{result}')
```

```
stud=Marks(1,"naresh",60,70,90)
stud.find_result()
```

Output

```
1,naresh,60,70,90,pass
```

Public

Public members are not prefix with any underscore
Public members are accessible within class and outside the class

Example:

```
class Employee:
    def __init__(self,n,s):
        self.name=n
        self.__salary=s
    def getSalary(self):
        return self.__salary
    def update_sal(self,s):
        self.__salary=self.__salary+s



emp1=Employee("naresh",6000)
print(emp1.name)
print(emp1.getSalary())
emp1.update_sal(1000)
print(emp1.getSalary())
```

Output

```
naresh
6000
7000
```

Protected

Protected members are prefix with _ (single) underscore.
These members are accessible within class, inherited class but cannot accessible outside the class.

What is difference between private,public and protected?

	Private	**Protected**	**Public**
Within class	YES	YES	YES
Outside the class	NO	NO	YES
Inherited class	NO	YES	YES

Example:

```
# Write a program to read the scores of n players and display

class Player:
    def __init__(self,n,s):
        self.__name=n
        self.__score=s
    def getName(self):
        return self.__name
    def getScore(self):
        return self.__score



n=int(input("How many players ?"))
playerList=[]
for i in range(n):
    name=input("Enter Name :")
    score=int(input("Enter Score :"))
    p=Player(name,score)
    playerList.append(p)

for p in playerList:
    print(p.getName(),p.getScore())
```

Output

```
How many players ?2
Enter Name :rohit
Enter Score :90
Enter Name :virat
```

Enter Score :50
rohit 90
virat 50

Example:

```
# Write a program to find area of triangle

class Triangle:
    def __init__(self,b,h):
        self.__base=b
        self.__height=h
    def find_area(self):
        return self.__base*self.__height*0.5



t1=Triangle(1.5,2.5)
area=t1.find_area()
print(area)
```

Output

1.875

Example:

```
class Person:
    def __init__(self):
        self.__name=None
        self.__age=None
    def set_data(self,n,a):
        self.__name=n
        self.__age=a
    def getName(self):
        return self.__name
    def getAge(self):
        return self.__age

p1=Person()
p1.set_data("naresh",40)
print(p1.getName(),p1.getAge())
```

Output
naresh 40

Class level variables and Methods