In [1]:

```
class D:
    def __init__(self,num):
        self.__num = num
        self.value = 0
    def disp(self):
        return self.__num+self.value
    def set_num(self,num):
        self.__num = num
```

In [2]:

```
d = D(100)
d.value = 200
d.set_num(d.value)
d.disp()
```

Out[2]:

```
400
```

In [8]:

```
class A:
    def __init__(self,a,b):
        self.__a = a
        self.__b = b
        self.c = "Dj"
    def f(self):
        return self.__a
    def g(self):
        return self.__b

q = A(10,20)
#print(q.__a, q.__b, q.__c)
print(q.f(),q.g(),q.c)
```

```
10 20 Dj
```

In [9]:

```
from abc import ABCMeta
class A(m = ABCMeta):
    def __init__(self):
        print("Debanik")
    @abstractmethod
    def d(self):
        pass
    def d1(self):
        pass

class C(A):
    def a(self):
        pass

q = C()
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-9-404776a03d1e> in <module>
      1 from abc import ABCMeta
----> 2 class A(m = ABCMeta):
      3     def __init__(self):
      4         print("Debanik")
      5     @abstractmethod

<ipython-input-9-404776a03d1e> in A()
      3     def __init__(self):
      4         print("Debanik")
----> 5     @abstractmethod
      6     def d(self):
      7         pass

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

In [10]:

```
class A:
    def m1(self):
        return 10
    def m2(self):
        return 20
class B:
    def m3(self):
        return 60
    def m4(self):
        return 70

class C(A):
    def m1(self):
        return 30
    def m4(self):
        return 40
class D(B):
    def m1(self):
        return 50
    def m2(self):
        return 60
```

In [11]:

```
c = C()
d = D()
```

In [12]:

```
print(c.m2()+d.m4())
print(c.m1()+d.m1())
print(c.m4()+d.m3())
```

```
90
80
100
```

In [24]:

```
class A:
    def __init__(self):
        self.__var = 100
    def m1(self):
        return self.__var
class B(A):
    def __init__(self,num):
        super().__init__()
        self.__num = num
    def m2(self):
        return self.__num + self.m1()
```

In [25]:

```
d = B(50)
d.m2()
```

Out[25]:

```
150
```

In [30]:

```
class A:
    def __init__(self):
        self.__var = 200
    def m1(self):
        return self.__var + 1
class B(A):
    def m2(self):
        return 100
class C(B):
    def __init__(self):
        super().__init__()
        self.num = 200
```

In [32]:

```
b= B()
c = C()
print(c.m1())
```

```
201
```

In [43]:

```
class E:
    __c = 100
    def __init__(self,name):
        self.name = name
        self.id = E.__c
        E.__c += 1

    @staticmethod
    def total():
        print("Emp Count",E.__c - 100)
```

In [44]:

```
e1  =E("Debanik")
print("e1 id:",e1.id)
e2  =E("Payel")
print("e1 id:",e2.id)
E.total()
```

```
e1 id: 100
e1 id: 101
Emp Count 2
```

In [47]:

```
class Sample:
    n = 100
    def __init__(self, num):
        self.val = None
    def set_val(self, num):
        val = num
    def get_val(self):
        self.val = n
        return self.val
```

In [48]:

```
obj = Sample(200)
obj.set_val(10)
print(obj.get_val())
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-48-1996548f7877> in <module>
      1 obj = Sample(200)
      2 obj.set_val(10)
----> 3 print(obj.get_val())

<ipython-input-47-2d8ac1b6e6e3> in get_val(self)
      6         val = num
      7     def get_val(self):
----> 8         self.val = n
      9         return self.val

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

In [ ]:

```

```