```
In [2]: class account:
            pass

        class emp:
            pass

        a1=account()
        a2=account()
        print(a1)
        print(a2)
        e1=emp()
        print(e1)
```

```
<__main__.account object at 0x0000023DC09AA5D0>
<__main__.account object at 0x0000023DC09A9D90>
<__main__.emp object at 0x0000023DC09A9810>
```

```
In [3]: x=10
        print(x)
```

```
10
```

```
In [4]: x=int(10)
        print(x)
```

```
10
```

```
In [6]: def deposit():
            print("this is deposit fun")

        class account:
            def deposit():
                print("this is deposit method")

        deposit()
        a=account()
        a.deposit() #by interpreter --->a.deposit(a)
```

```
this is deposit fun
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[6], line 10
      8 deposit()
      9 a=account()
---> 10 a.deposit()

TypeError: account.deposit() takes 0 positional arguments but 1 was given
```

```
In [7]: def show():
            print("this is show")

        show(10)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[7], line 4
      1 def show():
      2     print("this is show")
----> 4 show(10)

TypeError: show() takes 0 positional arguments but 1 was given
```

```
In [10]: class account:
             def deposit(x):
```

```
        print("this is deposit method",x)

a=account()
a.deposit()  #by interpreter --->a.deposit(a)

a2=account()
a2.deposit()#by interpreter --->a2.deposit(a2)
```

```
this is deposit method <__main__.account object at 0x0000023DC1A121D0>
this is deposit method <__main__.account object at 0x0000023DC0DA16D0>
```

In [11]:
```
class account:
    def deposit(khud):
        print("this is deposit method")

a=account()
a.deposit()  #by interpreter --->a.deposit(a)

a2=account()
a2.deposit()#by interpreter --->a2.deposit(a2)
```

```
this is deposit method
this is deposit method
```

In [12]:
```
class account:
    def deposit(self):
        print("this is deposit method")

a=account()
a.deposit()  #by interpreter --->a.deposit(a)

a2=account()
a2.deposit()#by interpreter --->a2.deposit(a2)
```

```
this is deposit method
this is deposit method
```

In [13]:
```
class account:
    def deposit(self,x):
        print("this is deposit method")

a=account()
a.deposit(100)  #by interpreter --->a.deposit(a,100)

a2=account()
a2.deposit(200)#by interpreter --->a2.deposit(a2,200)
```

```
this is deposit method
this is deposit method
```

In [15]:
```
class test:
    def show(self):
        print("this is show")

class other:
    def disp(self):
        print("this is disp")

t=test()
t.show()
t.disp()
```

```
this is show
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Cell In[15], line 11
      9 t=test()
```

```
     10 t.show()
---> 11 t.disp()

AttributeError: 'test' object has no attribute 'disp'
```

In [16]:
```python
x=[]
x.upper()
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Cell In[16], line 2
      1 x=[]
----> 2 x.upper()

AttributeError: 'list' object has no attribute 'upper'
```

In [17]:
```python
s="abababa"
print(s.count('a'))
```

```
4
```

In [18]:
```python
x=[1,2,1,2,1,2]
print(x.count(1))
```

```
3
```

In [19]:
```python
x=()
x.append(10)
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Cell In[19], line 2
      1 x=()
----> 2 x.append(10)

AttributeError: 'tuple' object has no attribute 'append'
```

In [20]:
```python
class account:
    def deposit(self,x):
        print("this is deposit method")

a1=account()
a1.deposit(100)
a1.deposit(200)
a1.deposit(300)
```

```
this is deposit method
this is deposit method
this is deposit method
```

In [21]:
```python
class account:
    def deposit(self,x):
        print("this is deposit method")

a1=account()
a1.deposit(100)

a2=account()
a2.deposit(200)

a3=account()
a3.deposit(300)
```

```
this is deposit method
this is deposit method
this is deposit method
```

In [22]:
```
#How to add data members(properties) to object
class account:
    pass

a1=account()
a2=account()
a1.acn=101
a1.bal=2000

a2.acn=102
a2.bal=1000

a1.bal=a1.bal+500
print(a1.acn,a1.bal)
print(a2.acn,a2.bal)

a3=account()
```
```
101 2500
102 1000
```

In [23]:
```
#Constructor method-->used to initialize newly created object

class account:
    def __init__(self):
        print("this is constructor")

a=account()
```
```
this is constructor
```

In [24]:
```
#Constructor method-->used to initialize newly created object

class account:
    def __init__(self):
        self.acn=101
        self.bal=2000

a1=account()
a2=account()
print(a1.acn,a1.bal)
print(a2.acn,a2.bal)
```
```
101 2000
101 2000
```

In [25]:
```
#Constructor method-->used to initialize newly created object

class account:
    def __init__(self,a,b):
        self.acn=a
        self.bal=b

a1=account(101,1000)
a2=account(102,2000)
print(a1.acn,a1.bal)
print(a2.acn,a2.bal)
```
```
101 1000
102 2000
```

Type of data members

- instance data members
    - represent property of individual object

        - separate copy is allocated inside each object
    - class data members
        - represent proprty of class and shared to all objects
        - single copy is allocated

In [8]:
```
class test:
    x=10             #class data member

    def __init__(self):
        self.y=20    #instance data member

print(test.x)
t=test()
t2=test()
print(t.y,t2.y)
t.y=200
print(t.y,t2.y)
print(t.x,t2.x)
test.x=100
print(t.x,t2.x)
```

```
10
20 20
200 20
10 10
100 100
```

Type of Methods

- instance method
- class method
- static method

In [9]:
```
class test:

    def m1(self):           #instance method
        print("this is m1")

    @classmethod
    def m2(cls):            #class method
        print("this is m2")

    @staticmethod
    def m3():               #static method
        print("this is m3")
```

In [14]:
```
t=test()
t.m1()     #t.m1(t)

test.m2()  #test.m2(test)
t.m2()     #test.m2(test)

test.m3()  #test.m3()
t.m3()     #test.m3()
```

```
this is m1
this is m2
this is m2
this is m3
this is m3
```

```
class calc:
```

In [15]:
```
    def __init__(self):
        self.x=4
        self.y=5

    def mul(self):
        print(self.x*self.y)

obj=calc()
obj.mul()
```
```
20
```

In [17]:
```
class calc:
    x=4
    y=5

    @classmethod
    def mul(cls):
        print(cls.x*cls.y)

calc.mul()
```
```
20
```

In [18]:
```
class calc:

    @staticmethod
    def mul(x,y):
        print(x*y)

calc.mul(4,5)
```
```
20
```

In [19]:
```
class test:
    x=3
    y=4
    print(x*y)
```
```
12
```

In [ ]:
```

```