In [7]:
```
class MyClass:
    def fun(self):  #self is a keyword which says this fun belongs to this partic
        pass
    def display(self,name):
        print("name is:",name)

mc=MyClass()
mc.fun()
mc.display(name="amulya")
```

```
name is: amulya
```

In [12]:
```
class demo:
    def des(self,model):
        print("brand of car:",model)
    def cost(self,cost):
        print("cost for the car:",cost)
d=demo()
d.des(model="Toyota")
d.cost(cost=1000000)
```

```
brand of car: Toyota
cost for the car: 1000000
```

In [35]:
```
class Demo:
    def m1(self):
        print("instance method")
    @staticmethod
    def m2(name):
        print("static method:",name)
d=Demo()
d.m1()
d.m2(name="yes")
```

```
instance method
static method: yes
```

In [37]:
```
#declaring variables inside the class
class myclass:
    a=20
    b=10  #class variables
    def add(self):
        print(self.a+self.b)
    def multi(self):
        print(self.a*self.b)
mc=myclass()
mc.add()
mc.multi()
```

```
30
200
```

In [41]:
```
#local variable,global variable, class variable
i,j=15,25
class LocalVar:
    a,b=10,20  #class variables

    def add(self,x,y):  #local variables
        print(x+y)     #accessing local variables
        print(self.a+self.b) #accessing class variables
        print(i+j) #accessing global variable
lv=LocalVar()
lv.add(2,4)
```

```
6
30
40
```

In [45]:
```
#local variable,global variable, class variable with same name
a,b=2,4
class MyClass:
    a,b=10,20
    def add(self,a,b):
        print("sum of local variable:", a+b)    #local variables
        print("sum of class variales:", self.a+self.b)  #class variables
        print("sum of global variables:", globals()['a']+globals()['b'])   #globa
mc=MyClass()
mc.add(10,20)
```

```
sum of local variable: 30
sum of class variales: 30
sum of global variables: 6
```

In [48]:
```
#Accessing local,class,global variables
x,y=4,2
class myCls:
    a,b=10,20

    def add(self,i,j):
        print(i+j)
        print(self.a*self.b)
        print(globals()['x']-globals()['y'])
mc=myCls()
mc.add(10,5)
```

```
15
200
2
```

In [50]:
```
#creating multiple objects to one class
class MyClass:
    def display(self):
        print("good morning!")
m1=MyClass()
m1.display()

m2=MyClass()
m2.display()
```

```
good morning!
good morning!
```

In [51]:
```
#named and nameless objects
class Myclass:
    def display(self):
        print("good moring!")
mc=Myclass()
mc.display()  #named Object

MyClass().display()  #nameless objects
```

```
good moring!
good morning!
```

In [58]:
```
#check the memory locations of the object
class Myclass:
    def display(self):
        pass
c1=Myclass()
c2=Myclass()
c3=c1
print(id(c1))  #checking the location
print(id(c2))
print(id(c3))
print(c1 is c2)
print(c2 is c3)
print(c1 is c3)

print(c1 is not c2)
print(c2 is not c3)
print(c1 is not c3)
```

```
1420982852336
1420982850368
1420982852336
False
False
True
True
True
False
```

In [1]:
```
#constructor
class mycls:
    def m1(self):
        print("good morning!")
    def __init__(self):
        print("this is constrctor")
m=mycls()
m.m1()
```

```
this is constrctor
good morning!
```

In [7]:
```
#accesing local variables into class level
class myclass():
    def m1(self,var1,var2):
        print(var1)
        print(var2)
        self.var1=var1
        self.var2=var2
    def add(self):
        print(self.var1+self.var2)
mc=myclass()
mc.m1(10,20)
mc.add()
```

```
10
20
30
```

In [18]:
```
class myclass():
    def __init__(self,var1,var2):
        print(var1)
        print(var2)
        self.var1=var1
        self.var2=var2
    def add(self):
        print(self.var1+self.var2)
mc=myclass(10,20)
mc.add()
```

```
10
20
30
```

In [22]:
```
#call current class method in another method
class myclass():
    def m1(self):
        print("first method")
        self.m2(200)
    def m2(self,b):
        print("second method",b)
mc=myclass()
mc.m1()
```

```
first method
second method 200
```

In [24]:
```
#pass arguments to the constructor
class myclass():
    def __init__(self,name):
        print(name)
mc=myclass("roshan")
```

```
roshan
```

In [25]:
```
#pass arguments to the constructor
class myclass():
    name="radha"
    def __init__(self,name):
        print(name) #local variables
        print(self.name) #class variables
mc=myclass("roshan")
```

```
roshan
radha
```

In [31]:
```
class emp:
    def __init__(self,ename,eid,esal):
        self.ename=ename
        self.eid=eid
        self.esal=esal
    def display(self):
        print("ename:{},eid:{},esal:{}".format(self.ename,self.eid,self.esal))
        print("ename:%s,eid:%d,esal:%f"%(self.ename,self.eid,self.esal))
e=emp('Amulya',101,100000)
e.display()
```

```
ename:Amulya,eid:101,esal:100000
ename:Amulya,eid:101,esal:100000.000000
```

In [32]:
```
class myclass:
    pass
c=myclass()
print(c)
```

```
<__main__.myclass object at 0x00000159296D1BB0>
```

In [35]:
```
class dclass:
    def __str__(self):
        return "hello"
c=dclass()
print(c)
```

```
hello
```

In [38]:
```
#print using refernce  __str__
class mydemo:
    def __init__(self,var1,var2):
        print(var1)
        print(var2)
        self.var1=var1
        self.var2=var2
    def __str__(self):
        return("var1:{},var2:{}".format(self.var1,self.var2))
md=mydemo(10,20)
print(md)
```

```
10
20
var1:10,var2:20
```

In [41]:
```
#__del__
class myclass:
    def __del__(self):
        print("destroyed")
m=myclass()
del m
```

```
destroyed
```

In [ ]:
```
```