Types of Methods

Tuesday, July 15, 2025 7:28 PM

There are 3 types of methods

1. Object/Instance Method
2. Class method
3. Static Method

1. Object Method: It is a method that is used to access and modify the members of an object.

   Passing self is mandatory.

   Syntax:
   Class Cname:

   Def Mname(self,args):
   S B

   Obj=Cname(args)
   Cname.mname(obj,values)
   Or
   Obj.Mname(values)

   Eg:

```
class School:
   sname='DPS'
   loc='Delhi'
   def __init__(self,sname,sid):
      self.sname=sname
      self.sid=sid
   def disp(self,sname,sid):
      print(self.sname,self.sid)
s1=School('A',1)
School.disp(s1,'A',1)
s1.disp('A',1)
```

2. Class Method: It is used to access or modify the members of a class.

   Passing 'cls' is mandatory
   @classmethod is used here

   Syntax:
   Class Cname:

   @classmethod
   Def mnane(cls,args):
   S B
   Obj=Cname(values)

```
Cname.mname(values)
        Or
Obj.mname(values)


Creating objects is optional

Eg:
'''class Comp:
   cname='TCS'
   loc='Banglore'
   ctype='Service based'
   pcompany='Tata'
   ceo='K. Krithivasan'

   @classmethod
   def disp(cls):
      print(cls.cname,cls.loc,cls.ctype,cls.pcompany,cls.ceo)
   @classmethod
   def ch_loc(cls,new):
      cls.loc=new

Comp.disp()
Comp.ch_loc('Pune')
Comp.disp()'''


Eg for both class and object method:
class School:
   name='DPS'
   loc='Delhi'
   prin='Mr. A K Datta'
   scid=5545

   def __init__(self,sname,sid,sclass,grade):
      self.sname=sname
      self.sid=sid
      self.sclass=sclass
      self.grade=grade

   def dis_st(self):
      print(self.sname,self.sid,self.sclass,self.grade)

   def ch_cls(self,new):
      self.sclass=new

   @classmethod
   def dis_sc(cls):
      print(cls.name,cls.loc,cls.prin,cls.scid)

   @classmethod
   def ch_prin(cls,new):
      cls.prin=new
```

```
s1=School('Aman',501,5,'A+')
s1.dis_st()
School.dis_sc()
s1.ch_cls(6)
s1.dis_st()
School.ch_prin('S Roy')
School.dis_sc()
```

3. Static Method: It is the class which is neither related to the class nor the object but works as the supportive for both class and object.

@staticmethod is used.

Syntax:

```
Class Cname:
        --------
        @staticmethod
        Def mname(args):
                S B
Ob=Cname(values)
Self.mname(values) #if want to work with respect to object method
        Or
Cls.mname(values) #if we want to work with respect to class method
```

Eg:

```
class Calculator:
   @staticmethod
   def add(a, b):
      return a + b

print(Calculator.add(5, 3))
```

Eg2:

```
class NumberCheck:
   @staticmethod
   def is_even(number):
      return number % 2 == 0

print(NumberCheck.is_even(4))
print(NumberCheck.is_even(7))
```