static Variables :

Adding the static variable from outside of the class

- Syn: <classname>.<variablename>=<value>

Deleting the static variable from outside of the class

- Syn: del <className>.<variablename>

Example:

```
class Sample:
   x=111 #static variable

#Access static variable from outside of the class
print("From outside of class")
print("static variable x : ",Sample.x)

#Adding new static variable the class Sample
#<ClassName>.<variablename>=value
Sample.y=222
print("static variable y : ",Sample.y)

#Deleting static variable from the class Sample
#del  <ClassName>.<staticvariablename>
del Sample.x
print("static variable x : ",Sample.x)
```

Variables in the class 2 types

- Instance fields or instance variable
- Static fields or static variable

Methods in the class 3 types

- Instance fields
    - Mutable methods
    - Immutable methods
    - Initializer methods [constructors]

Class methods

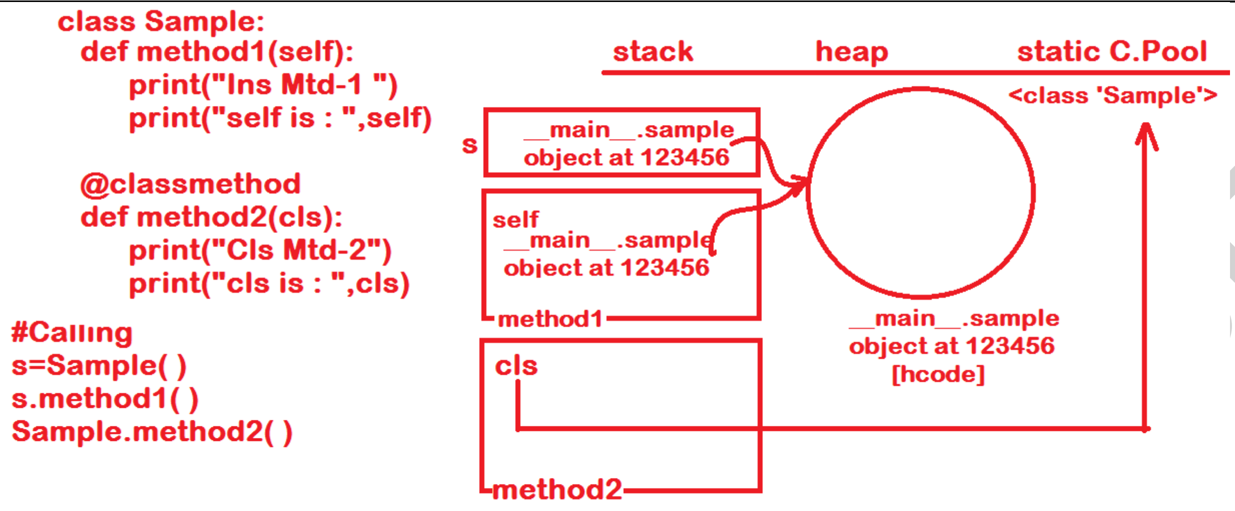
- The methods which are defined by using "cls" as the first argument and it should be defined by using a predefined decorator "@classmethod"

- "cls" argument of the class method can hold "Classname"

- In the "class methods" we can use only "class variable [static variables]"

- Instance methods can perform the operations on both "static variable and instance fields"

- "classmethods" can be referred by either classname or object reference whenever you want access it from outside of the class



```
class Sample:
  def method1(self):
      print("Ins Mtd-1 ")
      print("self is : ",self)

  @classmethod
  def method2(cls):
      print("Cls Mtd-2")
      print("cls is : ",cls)

#Calling
s=Sample( )
s.method1( )
Sample.method2( )
```

stack
heap
static C.Pool
<class 'Sample'>
s
__main__.sample object at 123456
self
__main__.sample object at 123456
method1
cls
method2
__main__.sample object at 123456 [hcode]

Example 2:

```
class Sample:
   def method1(self):
      print("Instance mtd-1")
      print("self is : ",self)

   @classmethod
   def method2(cls):
      print("Class mtd-2")
      print("cls is : ",cls)

#Calling
s=Sample()
s.method1()
Sample.method2()
```

Example 3:

```
class Sample:
   x=222 #static variable

   def method1(self): #instance method
      self.y=111     #instance field

   def method2(self):  #instance mtd
      self.y=self.y+Sample.x
      print("Result is : ",self.y)

   @classmethod
   def method3(cls):
      print("static variable x : ",cls.x)
      #print("instance y is : ",self.y) NameError

#calling
s=Sample()
s.method1()  #calling instance mtd
s.method2()  #calling an instance mtd
Sample.method3()
```

Static Methods

- The methods which are defined with in the class by using predefined decorator “@staticmethod”
- The static methods should not defined with “cls” or “self” as first argument

- **In static methods we can’t access static variable by cls and we can’t access instance fields by using “self”**
- **Function are defined outside of the class , but if define group of functions associated with a particular class then we have to user static methods**
- **Static methods also known utility methods**
- **Static methods can be referred by using either by the name or by the object reference whenever you want access it from outside of the class**

Example:

```
class Maths:
  @staticmethod
  def add(x,y):
    return x+y

  @staticmethod
  def sub(x,y):
    return x-y

#Calling
a=Maths.add(10,20)
print("Add of two is : ",a)
m=Maths()
s=m.sub(10,20)
print("Sub is :",s)
```

#When to use an instance methods ?

'''Whenever you want perform the operations on both
static variables and instance variable then we have define instance mtd
2.instance method must be defined by "self" as first argument
'''

#When to use class methods ?

'''Whenever you want perform the operations only on static variables then we have to define class methods
2.class method must be defined by using predefine dec
"@classmethod"
and "cls" as the first argument
'''

#When to use static methods | utility methods ?

'''Whenever you don't want perform any operations on static variable or instance variable then we have to define static methods
3.static methods must be defined by using "@staticmethod"
without "self" or "cls" as first arguments
'''

```
class MyMaths:
   @staticmethod
   def add(self,cls):  #here self and cls are the formal parameter
      print("self val is : ",self)
                        #formal parameter or acts as local variable
      print("cls val is : ",cls)
      print("Sum is : ",self+cls)

#calling
MyMaths.add(10,20)
```