Experiment No : 11

Aim : Write python programs to understand a) Classes, Objects, Constructors, Inner class and Static method.

Description:

Instance Methods

The first method on MyClass, called method, is a regular instance method. That's the basic, no-frills method type you'll use most of the time. You can see the method takes one parameter, self, which points to an instance of MyClass when the method is called (but of course instance methods can accept more than just one parameter).

Through the self parameter, instance methods can freely access attributes and other methods on the same object. This gives them a lot of power when it comes to modifying an object's state.

Not only can they modify object state, instance methods can also access the class itself through the self.__class__ attribute. This means instance methods can also modify class state.

Class Methods

Let's compare that to the second method, MyClass.classmethod. I marked this method with a @classmethod decorator to flag it as a class method.

Instead of accepting a self parameter, class methods take a cls parameter that points to the class—and not the object instance—when the method is called

Because the class method only has access to this cls argument, it can't modify object instance state. That would require access to self. However, class methods can still modify class state that applies across

all instances of the class.

Static Methods
The third method, MyClass.staticmethod was marked with a @staticmethod decorator to flag it as a static method.

This type of method takes neither a self nor a cls parameter (but of course it's free to accept an arbitrary number of other parameters).

Therefore a static method can neither modify object state nor class state. Static methods are restricted in what data they can access - and they're primarily a way to namespace your methods.

Implementation:
Code :

```
#1) class and object
x=1
y="1"
z=1.03
a= True
#class is a design used to creating the object.
#object is simple instance of class.
# Build in classes.
print(type(x))
print(type(y))
print(type(z))
print(type(a))


#user define classes
class shri:
# if we keep class empty and trying to run empty class , gives error.
```

```
#if you will keep (pass) . no error will generate.
 pass

# s1 is object of class shri.
# or it is sort of constructor.
#s1=shri()

# using __init__ method to create constructor .
#constructor automatically create on called on object creation.
#constructor are used to initialize member variable of class
class school:
 def __init__(self,m,s):
  self.math=m
  self.science=s
c1=school(90,80)

print(c1.math)
print(c1.science)



# also write like . no error will come


class school:
 def __init__(shri,m,s):
# instance variable inside int function
# these are object specific
  shri.m=m
  shri.s=s
c1=school(90,80)
print(c1.m)
print(c1.s)
```

```
# variable in class and object
class school:
 a=5
#  (a) is class variable or static variable. (a) value common to all.
 def __init__(shri,m,s) :
# instance variable inside int function
    shri.m = m
    shri.s = s
c1=school(70,80)
print(c1.m)
print(c1.s)
# create new object
c2=school(100,90)
# accessing class variable.
print(c1.a)
print(c2.a)
print(school.a)
print(school.a)


# 2)types of methods in class and object( static method)
class school:

 a=5
 def __init__(shri,m,s):
    shri.m = m
    shri.s = s
#1) instance methods associated with instance variable(like shri, self
etc.)
 def output1(shri):
    print(shri.a,shri.m)
    print(shri.a,shri.s)

 #2)classmethod associated with class variable (like a)
```

```
 @classmethod
 def output2(cls):
   print(cls.a)
 #3)static method. if you want to add ,extra in respective class.
 @staticmethod
 def output3():
    print("hello this is a static method")
c1=school(70,80)
c2=school(100,90)
# c1, c2 and c3 working behind the scene
c2.output1()
c2.output2()
c1.output3()

# 3)Inner classes
#ython has inner classes. Inner classes are classes that are defined
inside other classes.
#Inner classes have access to all the members of their outer classes.

#Indention is very important when using inner classes. Every inner
class should be indented by 4 spaces.

class college:
    def __init__(self):
        print('outer class constructor')

# student class is inner class
    class student:
        def __init__(self):
            print('inner class constructor')
        def displayS(self):
            print('student method')
#This method belongs to college class
    def displayC(self):
```

```
        print('college method')

# c is object of college class

c=college()
c.displayC()
#generate error if
#c.displayS()

# create inner class object using outer class object
# init method is implicitly called
s=c.student()
s.displayS()

#Generate error if
#.displayC()
# outer class (college) with constructor (student)
s=college().student()
s.displayS()
```

Output:

Class And Object

```
<class 'int'>
<class 'str'>
<class 'float'>
<class 'bool'>
90
80
90
80
70
80
5
5
5
5
```

Static Method

```
5 100
5 90
5
hello this is a static method
```

Inner Class

```
outer class constructor
college method
inner class constructor
student method
outer class constructor
inner class constructor
student method
```

Conclusion: We are successfully implemented the python programs to understand a) Classes, Objects, Constructors, Inner class and Static method.