Name: Meith Navlakha
SAP : 60004190068
Python Lab

EXPERIMENT 2

AIM: To create functions, classes and objects using python

THEORY:

Python is an object-oriented programming language. An object is simply a collection of data (variables) and methods (functions) that act on those data. Similarly, a class is a blueprint for that object.

Class in Python:

A class is a user-defined blueprint or prototype from which objects are created. Classes provide a means of bundling data and functionality together. Each class instance can have attributes attached to it for maintaining its state. Class instances can also have methods (defined by their class) for modifying their state.

Class creates a user-defined data structure, which holds its own data members and member functions, which can be accessed and used by creating an instance of that class.

In Python, Class is defined by “def ” keyword.

Attributes are always public and can be accessed using the dot (.) operator. Eg.: Myclass.Myattribute. The values of the attributes of the objects of the class can also be modified.

Objects in Python:

An Object is an instance of a Class. When an object of a class is created, the class is said to be instantiated. All the instances share the attributes and the behaviour of the class. But the values of those attributes, i.e. the state are unique for each object. A single class may have any number of instances.

An object consists of:

- **State:** It is represented by the attributes of an object. It also reflects the properties of an object.
- **Behaviour:** It is represented by the methods of an object. It also reflects the response of an object to other objects.
- **Identity:** It gives a unique name to an object and enables one object to interact with other objects.

Name: Meith Navlakha
SAP : 60004190068
Python Lab

Constructors in Python:

The __init__ method is similar to constructors in C++ and Java. Constructors are used to initializing the object's state. Like methods, a constructor also contains a collection of statements(i.e. instructions) that are executed at the time of Object creation. It runs as soon as an object of a class is instantiated.

Use of __init__() function is to assign values to object properties, or other operations that are necessary to do when the object is being created.

"self " in Python:

Class methods must have an extra first parameter in the method definition. We do not give a value for this parameter when we call the method, Python provides it. This is similar to this pointer in C++ and this reference in Java.

The self parameter is a reference to the current instance of the class and is used to access variables that belongs to that class. Even If we have a method that takes no arguments, then we still have to have one argument.

1) Simple Class in Python

```
class Meith:
  m = 5

# instantise object of class
myclass = Meith()
print("Print m variable of object myclass : " , myclass.m)

# Modifiy the values of object
myclass.m = 10
print("Print UPDATED m variable of object myclass : " , myclass.m)

# Add attribute
myclass.msg = "Hello"

print('Added a new attribute to the object : ' , myclass.msg)
```

```
Print m variable of object myclass :  5
Print UPDATED m variable of object myclass :  10
Added a new attribute to the object :  Hello
```

Name: Meith Navlakha
SAP : 60004190068
Python Lab

2) Constructor in Python

```
class Students:

  field = 'Engineering'
  def __init__(self , name, age, course):

    self.name = name
    self.age = age
    self.course= course


# Initialise Student objects
Student1 = Students('Meith' , 20, 'Comps')
Student2 = Students('Python' , 19, 'IT')

# Print object values
print('*****STUDENT DETAILS*****')
print(Student1.name," : ", Student1.field , " " , Student1.course , " " , Student1.age)
print(Student2.name,": ", Student2.field , " " , Student2.course , " " , Student2.age)
```

```
*****STUDENT DETAILS*****
Meith  :  Engineering   Comps   20
Python :  Engineering   IT   19
```

3) Object Method:

```
class Students:

  # field = 'Engineering'
  def __init__(self , name, marks, course):

    self.name = name
    self.marks = marks
    self.course= course

  def calcPass(self):
    #  Method
    if(self.marks >=35):
      return "Pass"
    else:
      return 'Fail'

# Initialise Student objects
Student1 = Students('Meith' , 88, 'Comps')
Student2 = Students('Python' , 19, 'IT')

# Print object values
print('*****STUDENT RESULTS*****\n')
print(Student1.name," : " , Student1.course , " " , Student1.calcPass()) # calling calcPass method of the object
print(Student2.name,": ", Student2.course , "    " , Student2.calcPass())
```

```
*****STUDENT RESULTS*****

Meith  :  Comps   Pass
Python :  IT      Fail
```

4) PASS

```
[ ]  class Person:
       pass
```

CONCLUSION:

In this experiment we learnt about class and object in python. Class is the blueprint and object are instance of the class. Each class can have multiple objects, each have a unique pointer to it. Through objects we can access the variables and the methods of the class. Also, we can modify the values of the attributes of the object. In python, __init__ method is like a constructor and is called by default when an object is instantised. Thus, these are the class and object that I learnt in the experiment.