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. The methods of the object are basically the functions written in the class.

Functions in Python

A function is a group of related statements that performs a specific task. Python Functions is a block of related statements designed to perform a computational, logical, or evaluative task. Functions help break our program into smaller and modular chunks. As our program grows larger and larger, functions make it more organized and manageable. Furthermore, it avoids repetition and makes the code reusable.

Functions can be both **built-in** or **user-defined**.

In Python a function is defined using the “def ” keyword.

The functions are defined and then called upon. To call a function, use the function name followed by parenthesis

Syntax of Function

```
def function_name(parameters):
    """ docstring """
    statement(s)
```

Function definition consists of the following components:

- Keyword ‘def ’ that marks the start of the function header.
- A function name to uniquely identify the function. Function naming follows the same rules of writing identifiers in Python.
- A colon (:) to mark the end of the function header.
- One or more valid python statements that make up the function body. Statements must have the same indentation level.
- Parameters (arguments) through which we pass values to a function. They are optional.
- Optional documentation string (docstring) to describe what the function does.
- An optional return statement to return a value from the function.

Name: Meith Navlakha
SAP : 60004190068
Python Lab

Arguments in Functions

Any data types of argument to a function (string, number, list, dictionary etc.), and it will be treated as the same data type inside the function.

Arbitrary Arguments (*args)

If the number of arguments that will be passed into your function is undecided then add a * before the parameter name in the function definition. This way the function will receive a tuple of arguments which can be accessed using index.

The default name is args, but we can use any other name e.g. *names.

Arbitrary Keyword Arguments (**kwargs)

If the number of arguments that will be passed into your function is undecided then add a * before the parameter name in the function definition. This way the function will receive a *dictionary* of arguments, and can access the items by using keys of the values.

Recursion Function

Python also accepts function recursion, which means a defined function can call itself. Recursion is a common mathematical and programming concept. It means that a function calls itself. This has the benefit of meaning that you can loop through data to reach a result.

A complicated function can be split down into smaller sub-problems utilizing recursion. Sequence creation is simpler through recursion than utilizing any nested iteration. Recursive functions render the code look simple and effective.

Lambda Functions in Python

Lambda functions can have any number of arguments but only one expression. The expression is evaluated and returned. Lambda functions can be used wherever function objects are required. Lambda functions are used when we require a nameless function for a short period of time. Hence, are known as **Anonymous Functions** as well.

Use: In Python, we generally use it as an argument to a higher-order function (a function that takes in other functions as arguments). Lambda functions are used along with built-in functions like filter(), map() etc.

Syntax

lambda arguments: expression

Name: Meith Navlakha
SAP : 60004190068
Python Lab

Inner Functions in Python

A function which is defined inside another function is known as inner function or nested function. Nested functions are able to access variables of the enclosing scope. Inner functions are used so that they can be protected from everything happening outside the function. This process is also known as Encapsulation.

Built-In Functions in Python

The Python built-in functions are defined as the functions whose functionality is pre-defined in Python. The python interpreter has several functions that are always present for use. These functions are known as Built-in Functions. There are several built-in functions in Python which are listed below:

abs(x) : Return the absolute value of a number. The argument may be an integer, a floating point number, or an object implementing __abs__(). If the argument is a complex number, its magnitude is returned.

bin(x) : Convert an integer number to a binary string prefixed with “0b”. The result is a valid Python expression. If x is not a Python int object, it has to define an __index__() method that returns an integer.

chr(i) : Return the string representing a character whose Unicode code point is the integer i. For example, chr(97) returns the string 'a', while chr(8364) returns the string '€'. The valid range for the argument is from 0 through 1,114,111 (0x10FFFF in base 16). ValueError will be raised if i is outside that range.

all(iterable) : Return True if all elements of the iterable are true (or if the iterable is empty).

len(s) : Return the length (the number of items) of an object. The argument may be a sequence (such as a string, bytes, tuple, list, or range) or a collection (such as a dictionary, set, or frozen set).

Name: Meith Navlakha
SAP : 60004190068
Python Lab

Built-in Functions

A	E	L	R
abs()	enumerate()	len()	range()
aiter()	eval()	list()	repr()
all()	exec()	locals()	reversed()
any()			round()
anext()	F	M	
ascii()	filter()	map()	S
	float()	max()	set()
B	format()	memoryview()	setattr()
bin()	frozenset()	min()	slice()
bool()			sorted()
breakpoint()	G	N	staticmethod()
bytearray()	getattr()	next()	str()
bytes()	globals()		sum()
		O	super()
C	H	object()	
callable()	hasattr()	oct()	T
chr()	hash()	open()	tuple()
classmethod()	help()	ord()	type()
compile()	hex()		
complex()		P	V
	I	pow()	vars()
D	id()	print()	
delattr()	input()	property()	Z
dict()	int()		zip()
dir()	isinstance()		
divmod()	issubclass()		_
	iter()		__import__()

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.

Name: Meith Navlakha
SAP : 60004190068
Python Lab

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.

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.

Name: Meith Navlakha
SAP : 60004190068
Python Lab

1) Simple Function

```
# Function Defination
def my_func():
  print("Hello this is Meith")

# call function
my_func()
```

```
Hello this is Meith
```

2) Passing Arguments

```
# Function Definition
def calcArea(l):
  a= l*l
  print("Area of Square : " ,a)

# call function
calcArea(5)
```

```
Area of Square :  25
```

```
[35] def func(list):
       for x in list:
         print(x)

     func(['Apple' , 'Banana' , 'Mango' , 'Orange'])
```

```
Apple
Banana
Mango
Orange
```

```
# Passsing arguements

def my_func2(name, age):
  print("Hello this is ", name)
  print("I am ", age , "years old.")

# call function
my_func2('Meith' , 20)
```

```
Hello this is  Meith
I am  20 years old.
```

3) Arbitrary Arguments

```
def smallestNum(*args):

  a =list(args)
  a.sort()
  print('Smallest Number : ' ,a[0])

print('The Numbers : 10, 5, 8, 20 , 6' )
smallestNum(10, 5, 8, 20 , 6)
```

```
The Numbers : 10, 5, 8, 20 , 6
Smallest Number :  5
```

4) Keyword Arguments

```
def my_function(child3, child2, child1):
  print("The chosen child is " + child3)

my_function(child1 = "Tom", child2 = "Tobias", child3 = "Meith")
```

```
The chosen child is Meith
```

5) Return

```
def isNum(num):
  if (num>0):
    return 'Positive'
  elif (num<0):
    return 'Negative'
  else:
    return 'Neither Positive nor Negative'


# Calling Function
print( '15 is :', isNum(15))
print( '-8 is :', isNum(-8))
print( ' 0 is :', isNum(0))
```

```
15 is : Positive
-8 is : Negative
 0 is : Neither Positive nor Negative
```

6) Scope of Variable

```
def my_func():
  x = 10
  print("Value inside function:",x)

x = 20
my_func()
print("Value outside function:",x)
```

```
Value inside function: 10
Value outside function: 20
```

7) Recursion

```
def fact(n):
  if (n>0):
    return (n*fact(n-1))
  elif (n == 0):
    return 1
  else:
    return 'Enter positive number!!'

print("Factorial 5 :", fact(5))
print("Factorial 0 :", fact(0))
print("Factorial -4 :", fact(-4))
```

```
Factorial 5 : 120
Factorial 0 : 1
Factorial -4 : Enter positive number!!
```

8) Lambda Functions

```
multiply = lambda x: x*2
multiply(5)
```

```
10
```

```
# Lambda function with filter

num = [10 ,4, 5,27, 67, 18 , 36]
print("All the numbers : " , num)

divByThree = list(filter(lambda x: (x%3 == 0) , num))
print("Divisible by 3: " , divByThree)
```

```
All the numbers :  [10, 4, 5, 27, 67, 18, 36]
Divisible by 3:  [27, 18, 36]
```

Name: Meith Navlakha
SAP : 60004190068
Python Lab

9) Inner/Nested Functions

```
def outerfunc(s):

  # define inner function
  def innerfunc():
    print('Hello,', s)

  innerfunc() # call inner function

#  call outer function
outerfunc('Meith')
```

```
Hello, Meith
```

Scope of Variables in Nested Function

```
# Scope of Variables in Nested Functions
def outerfunc():
  s = 'In Outer Function'
  # define inner function
  def innerfunc():
    s = 'In Inner Function'
    print(s)

  print(s)
  innerfunc() # call inner function

#  call outer function
outerfunc()
```

```
In Outer Function
In Inner Function
```

Name: Meith Navlakha
SAP : 60004190068
Python Lab

10) 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
```

10) 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
```

Name: Meith Navlakha
SAP : 60004190068
Python Lab

11) 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
```

CONCLUSION:

In this experiment we learnt about function, class and object in python. A function is a block of related statements that performs a specific task. In python we have inbuilt and user defined functions. In Built-In functions I explored abs, bin, len, all, chr functions. Also, functions can have parameters which take in arguments when the functions are called. When the number of arguments is not known, multiple arguments can also be passed using *args which stores the arguments into a tuple and multiple key value pair as **kwargs which stores the arguments into a dictionary. Lastly, in functions I also learnt about Inner/Nested functions, Recursion and Lambda functions 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.