Experiment No. 4
Creating functions, classes and objects using python
Date of Performance:
Date of Submission:

Experiment No. 4

Title: Creating functions, classes and objects using python

Aim: To study and create functions, classes and objects using python

Objective: To introduce functions, classes and objects in python

Theory:

A function is a block of code which only runs when it is called.

You can pass data, known as parameters, into a function.

A function can return data as a result.

A class is a user-defined blueprint or prototype from which objects are created. Classes provide a means of bundling data and functionality together. Creating a new class creates a new type of object, allowing new instances of that type to be made. 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.

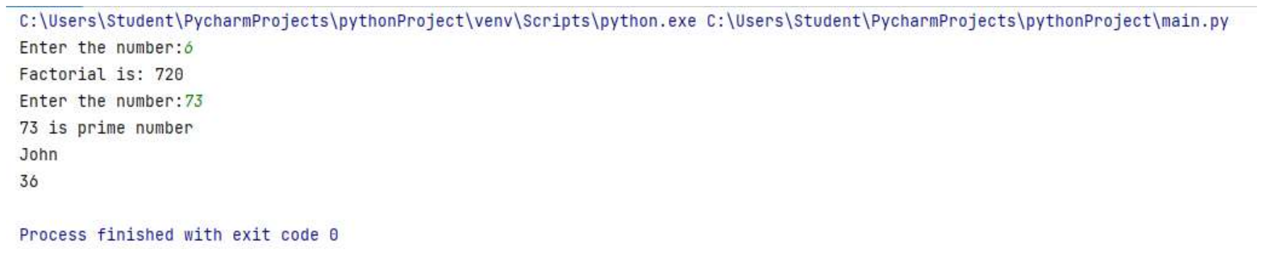
To understand the need for creating a class let's consider an example, let's say you wanted to track the number of dogs that may have different attributes like breed, age. If a list is used, the first element could be the dog's breed while the second element could represent its age. Let's suppose there are 100 different dogs, then how would you know which element is supposed to be which? What if you wanted to add other properties to these dogs? This lacks organization and it's the exact need for classes.

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. A class is like a blueprint for an object.

Code:

```
# Factorial of a number

x = int(input("Enter the number:"))

def fact(x):

    if(x==1):

        return 1;

    else:

        return x*fact(x-1);

print("Factorial is:",fact(x))


# Prime number

x = int(input("Enter the number:"))

def prime(x):

    num=2

    while(x>num):

        if(x%num==0):

            print(x,"is not a prime number")
            break
        num+=1
    else:
        print(x,"is prime number")
prime(x)


# Employee
class employee:
  def __init__(self, name, age):
    self.name = name
    self.age = age
p1 = employee("John", 36)
print(p1.name)
print(p1.age)
```

Output:



```
C:\Users\Student\PycharmProjects\pythonProject\venv\Scripts\python.exe C:\Users\Student\PycharmProjects\pythonProject\main.py
Enter the number:6
Factorial is: 720
Enter the number:73
73 is prime number
John
36

Process finished with exit code 0
```

Conclusion:

- **Factorial Calculation**: This portion of the code prompts the user to input a number, calculates its factorial using a recursive function, and then prints the result. The factorial function (fact) recursively computes the factorial of the given number x. It demonstrates the concept of recursion and showcases a common mathematical operation.
- **Prime Number Check**: Here, the code prompts the user to input a number and checks whether it is a prime number or not. It iteratively divides the number by integers starting from 2 up to the number itself to determine its primality. This part of the code illustrates a simple algorithm to check for prime numbers.
- **Employee Class**: This section defines a simple class named employee, which has attributes name and age. An instance of this class is created with specific values for name and age, and then its attributes are accessed and printed. This demonstrates the basics of defining and utilizing classes in Python.