

NAME: SUPRIYA SUBHASH VIRKAR CLASS: SE-3 ROLL NO: 58

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: def factorial():

```
a=int(input("enter a number"))
```

```
f=1
```

```
i=1
```

```
while(i<=a):
```

```
f=f*i
```

```
i=i+1
```

```
print(f)
```

```
factorial()
```

def primeNumber():

```
num = int(input("Enter a number: "))
```

```
flag = False
```

```
if num <2:
```

```
    print(num, "is not a prime number")
```

```
elif num > 2:
```

```
    for i in range(2, num):
```

```
        if num % i == 0:
```

```
            flag = True
```

```
            break
```

```
    if flag:
```

```
        print(num, "is not a prime number")
```

```
    else:
```

```
        print(num, "is a prime number")
```

```
primeNumber()
```

```
class Student:
```

```
    def __init__(self):
```

```
        self.name="supriya"
```

```
        self.age=20
```

```
self.marks=100
```

```
weight=46
```

```
pointer=7.7
```

```
print("hello")
```

```
print(self.marks+self.age)
```

```
print(weight+pointer)
```

```
def display(self):
```

```
year=2
```

```
print(self.name,self.age,self.marks,year)
```

```
def dance(self):
```

```
dancep=True
```

```
print(dancep)
```

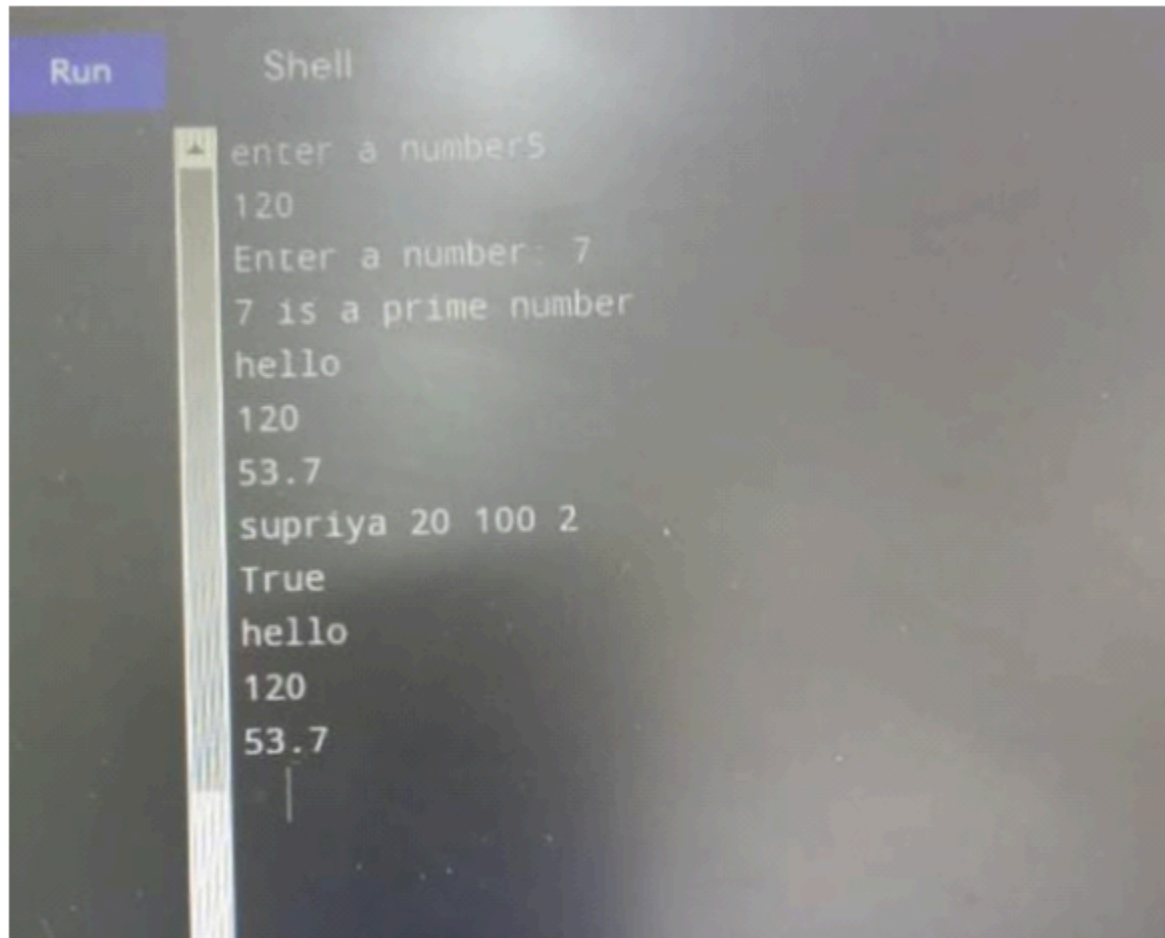
```
s1=Student()
```

```
s1.display()
```

```
s1.dance()
```

```
s1.__init__()
```

Output:

A screenshot of a Python Shell window. The window has a title bar with 'Run' and 'Shell' buttons. The shell displays the following text: 'enter a numbers', '120', 'Enter a number: 7', '7 is a prime number', 'hello', '120', '53.7', 'supriya 20 100 2', 'True', 'hello', '120', '53.7'. A vertical scrollbar is visible on the left side of the shell area.

```
Run Shell
enter a numbers
120
Enter a number: 7
7 is a prime number
hello
120
53.7
supriya 20 100 2
True
hello
120
53.7
```

Conclusion:

- Functions, classes, and objects are essential building blocks of Python programming, enabling modular, organized, and reusable code development.
- Functions facilitate code abstraction, decomposition, and reuse by encapsulating logic into callable units.
- Classes and objects promote code organization, data abstraction, and code reuse through encapsulation, inheritance, and polymorphism.
- Understanding how to create and use functions, classes, and objects is crucial for writing scalable, maintainable, and efficient Python code.

- Embracing object-oriented programming principles empowers developers to design flexible, extensible, and robust software solutions in Python.