Experiment No. 4
Creating functions, classes and objects using python
Name : Vaidehi D. Gadag
Branch/Div.: Comps-1 (C47)
Date of Performance: 07/02/2024
Date of Submission: 14/02/2024

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:

```
class Student:
    def __init__(self,name="",marks=0):
        self.name = name
        self.marks = marks
    def GradeCalculator(self):
        mark = self.marks
        if mark>60:
            return "Grade A"
        elif mark>50:
            return "Grade B"
        elif mark>40:
            return "Grade C"
        else:
            return "Fail"
s1=Student(input("Enter your name:"),int(input("Enter marks:")))
print("Student name: ",s1.name)
print("Student marks: ",s1.marks)
print("Student Grade:",s1.GradeCalculator())
print("*****************************************")
s2=Student(input("Enter your name:"),int(input("Enter marks:")))
print("Student name: ",s2.name)
print("Student marks: ",s2.marks)
print("Student Grade:",s2.GradeCalculator())
print("*****************************************")
s3=Student(input("Enter your name:"),int(input("Enter marks:")))
print("Student name: ",s3.name)
print("Student marks: ",s3.marks)
print("Student Grade:",s3.GradeCalculator())
print("*****************************************")
```

Output:

Enter your name: Vaidehi

Enter marks:90

Student name: Vaidehi

Student marks: 90

Student Grade: Grade A

****

Enter your name: Abc

Enter marks:80

Student name: Abc

Student marks: 80

Student Grade: Grade A

****

Enter your name: Vcet

Enter marks:100

Student name: Vcet

Student marks: 100

Student Grade: Grade A

****

Conclusion:

In conclusion, classes, objects, and functions are fundamental building blocks in Python programming, enabling the creation of robust and maintainable applications. Classes, serving as blueprints for objects, allow for the implementation of object-oriented programming principles, promoting code organization, reusability, and extensibility. Objects, as instances of classes, encapsulate data and behaviour, simplifying code complexity and enhancing modularity. Functions, as reusable blocks of code, perform specific tasks and can be defined both within classes as methods or independently, further promoting code modularity and reusability. Together, these components provide a powerful and flexible framework for developers to build scalable and efficient applications in Python. Mastering the implementation of classes, objects, and functions is essential for any Python developer, as it forms the foundation for more advanced programming concepts and techniques.