|  |
| --- |
| Experiment No. 4 |
| Creating functions, classes and objects using python |
| Date of Performance:19/01/24 |
| Date of Submission:02/02/24 |

**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, roll\_number, grade):

self.name = name

self.roll\_number = roll\_number

self.grade = grade

def display\_details(self):

print(f"Name: {self.name}")

print(f"Roll Number: {self.roll\_number}")

print(f"Grade: {self.grade}")

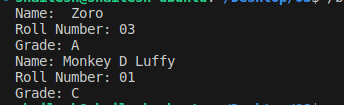
student1 = Student(" Zoro ", "03", "A")

student2 = Student("Monkey D Luffy", "01", "C")

student1.display\_details()

student2.display\_details()

**Output:**



**Conclusion:**

we delved into the concepts of functions, classes, and objects in Python, which are fundamental to object-oriented programming (OOP) and modular code design.

Functions are blocks of code that are defined to perform a specific task. They are reusable and can accept input parameters and return output values, making code more modular, readable, and maintainable.

Classes are blueprints or prototypes for creating objects. They encapsulate data and behavior, allowing for the creation of custom data types with attributes and methods. Classes provide a way to organize code logically and efficiently, promoting code reusability and scalability.

Objects are instances of classes, created using the class constructor. Each object has its own unique set of attributes and can execute methods defined within its class. Objects allow for the manipulation of data and behavior in a structured and organized manner.