|  |
| --- |
| Experiment No. 4 |
| Creating functions, classes and objects using python |
| Date of Performance: 05/02/2024 |
| Date of Submission: 12/02/2024 |

**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, age, grade):

self.name = name

self.age = age

self.grade = grade

def study(self):

print(f"{self.name} is studying diligently.")

def get\_grade(self):

return self.grade

student1 = Student("Alice", 17, "A+")

student2 = Student("Bob", 16, "B+")

print(f"{student1.name} is {student1.age} years old and achieved a grade of {student1.get\_grade()}.")

print(f"{student2.name} is {student2.age} years old and achieved a grade of {student2.get\_grade()}.")

student1.study()

student2.study()

**Output:**

Alice is 17 years old and achieved a grade of A+.

Bob is 16 years old and achieved a grade of B+.

Alice is studying diligently.

Bob is studying diligently.

**Conclusion:**

This Python program introduces the Student class, featuring attributes like name, age, and grade, coupled with methods for studying and grade retrieval. It instantiates two Student objects, representing Alice and Bob, allowing access to their attributes and invocation of methods. Through this demonstration, the script illustrates key concepts of object-oriented programming, encapsulation, and method invocation in Python. It underscores the adaptability of classes and objects in structuring and managing data, fostering code readability and modularity.