St. Francis Institute of Technology, Mumbai-400 103

A.Y. 2023-24
Class: SE-ITA/ITB, Semester: IV
Subject: **Python Lab**.

Experiment – 8: Python program to implement Inheritance

1. **Aim: Write a menu driven program to demonstrate OOP in python.**
   - Create a class ‘Employee’ with following attribute as ‘EmpID’, ‘name’, ‘dept’, and ‘salary’
   - Print ‘name: ‘, %s and ‘salary:’, %10.2f when an object is printed.
   - Create a function to update the salary of a given employee. Print the total number of employees.
   - Create two derived classes “Manager” and “Staff” from base class “Employee” and display their details.

2. **Prerequisite:** Basic knowledge of Python.
3. **Objective:** Knowledge of classes and objects in python.
4. **Requirements:** Personal Computer (PC), Windows /Linux Operating System, IDLE 3.10 for Python3.

5. Pre-Experiment Exercise:
   Theory:

- **Inheritance:**

A class can inherit attributes and behaviour methods from another class, called the superclass. A class which inherits from a superclass is called a subclass, also called heir class or child class. Superclasses are sometimes called ancestors as well. There exists a hierarchy relationship between classes.

Syntax:

```
class BaseClass:
        Body of base class class
DerivedClass(BaseClass):
        Body of derived class
```

6. **Laboratory Exercise**

   A) Procedure
   - Open Idle for python
   - Open editor in Idle from menu file-new
   - Type python code with proper syntax
   - Save file with .py extension
   - Execute the code inside the saved file using shortcut key F5 or using menu: Run-Run module

   B) Program code with comments:

Write and execute your program code to achieve the given aim and attach it with your own comments with neat indentation.

7. Post-Experiments Exercise

A) Extended Theory:

1. Explain Interfaces in Python with one example.
2. Explain types of inheritance with syntax.

B) Questions/Programs:

1. Write a Python program to declare a base class College having two derived classes student and faculty and display their details.

C) Conclusion:

1. What was performed in the experiment/program.
2. What is the significance of experiment/program?

D) References

1. James Payne, "Beginning Python: Using Python 2.6 and Python 3.1", WroxPublication.
2. Dr. R. Nageshwara Rao, "Core Python Programming", Dreamtech Press.
3. https://www.python.org/
4. www.pythonforbeginners.com
5. https://www.geeksforgeeks.org/inheritance-in-python/

IN-LAB Program:

1) Write a menu driven program to demonstrate OOP in python.

● Create a class ‘Employee’ with following attribute as ‘EmpID’, ‘name’, ‘dept’, and ‘salary’

● Print ‘name: ‘, %s and ‘salary:’, %10.2f when an object is printed.

● Create a function to update the salary of a given employee. Print the total number of employees.

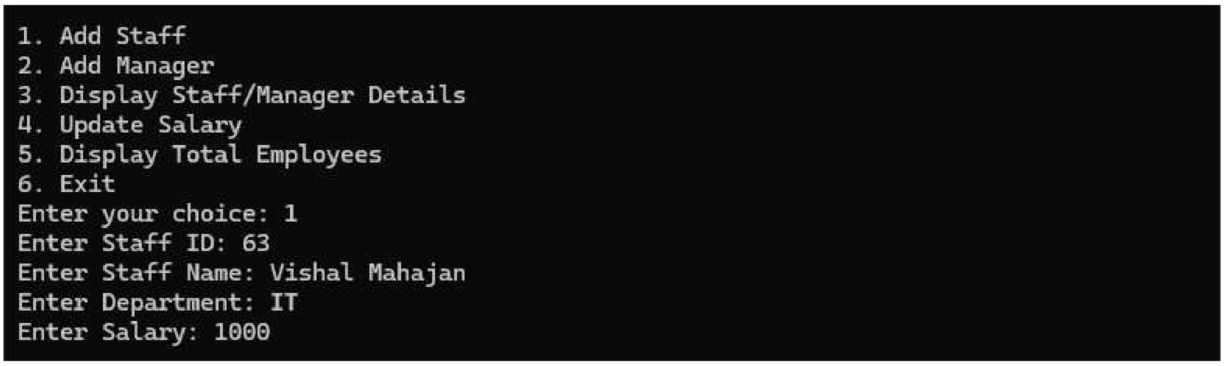
● Create two derived classes “Manager” and “Staff” from base class “Employee” and display their details.

Code:

```
class Employee:
    total_employees = 0
    def __init__(self, EmpID, name, dept, salary,role):
        self.EmpID = EmpID
        self.name = name
        self.dept = dept
        self.salary = salary
        self.role = role
        Employee.total_employees += 1

    def __str__(self):
        return "Name: %s, Salary: %10.2f" % (self.name, self.salary)

    def update_salary(self, salary):
        self.salary = salary

    def display(self):
        print("\nEmployee Details:")
        print("EmpID:", self.EmpID)
        print("Name:", self.name)
        print("Department:", self.dept)
        print("Salary:", self.salary)
        print("Role:", self.role)

    def total(self):
        print("\nTotal number of employees:", Employee.total_employees)



class Manager(Employee):
    def __init__(self, EmpID, name, dept, salary,role):
        super().__init__(EmpID, name, dept, salary,role)

    def display(self):
        super().display()

class Staff(Employee):
    def __init__(self, EmpID, name, dept, salary,role):
        super().__init__(EmpID, name, dept, salary,role)

    def display(self):
        super().display()

def main():
    employees = []

    while True:
        print("\n1. Add Staff")
        print("2. Add Manager")
        print("3. Display Staff/Manager Details")
        print("4. Update Salary")
        print("5. Display Total Employees")
        print("6. Exit")

        choice = int(input("Enter your choice: "))

        if choice == 1:
            EmpID = input("Enter Staff ID: ")
            name = input("Enter Staff Name: ")
            dept = input("Enter Department: ")
            salary = float(input("Enter Salary: "))
            employees.append(Staff(EmpID, name, dept, salary,role="Staff"))

        elif choice == 2:
            EmpID = input("Enter Manager ID: ")
            name = input("Enter Manager Name: ")
            dept = input("Enter Department: ")
            salary = float(input("Enter Salary: "))
            employees.append(Manager(EmpID, name, dept,
salary,role="Manager"))

        elif choice == 3:
            for employee in employees:
                employee.display()

        elif choice == 4:
            EmpID = input("Enter Employee ID to Update Salary: ")
            salary = float(input("Enter New Salary: "))
            for employee in employees:
                if employee.EmpID == EmpID:
                    employee.update_salary(salary)
                    print("Salary updated successfully.")
                    break
            else:
                print("Employee ID not found.")

        elif choice == 5:
            print("\nTotal number of employees:", Employee.total_employees)

        elif choice == 6:
            break

        else:
            print("Invalid choice. Please choose a valid option.")

if __name__ == "__main__":
    main()
```

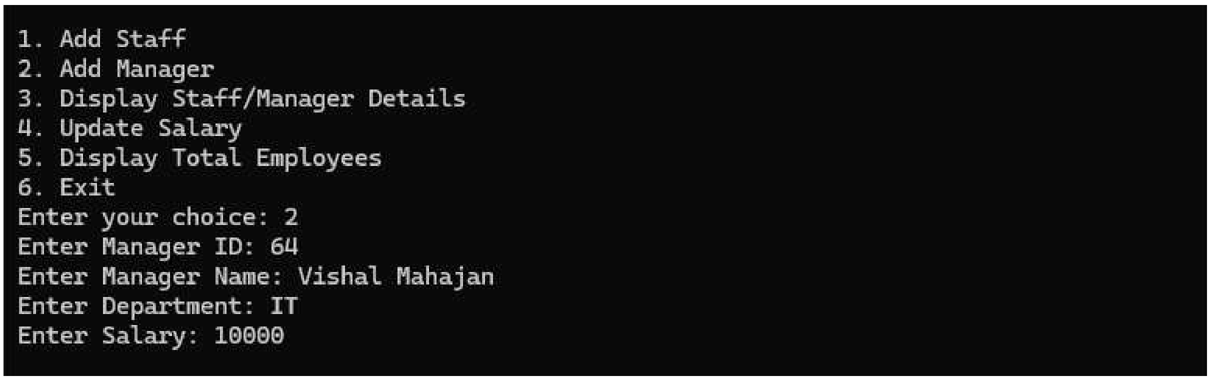
Output:

1) Adding Staff



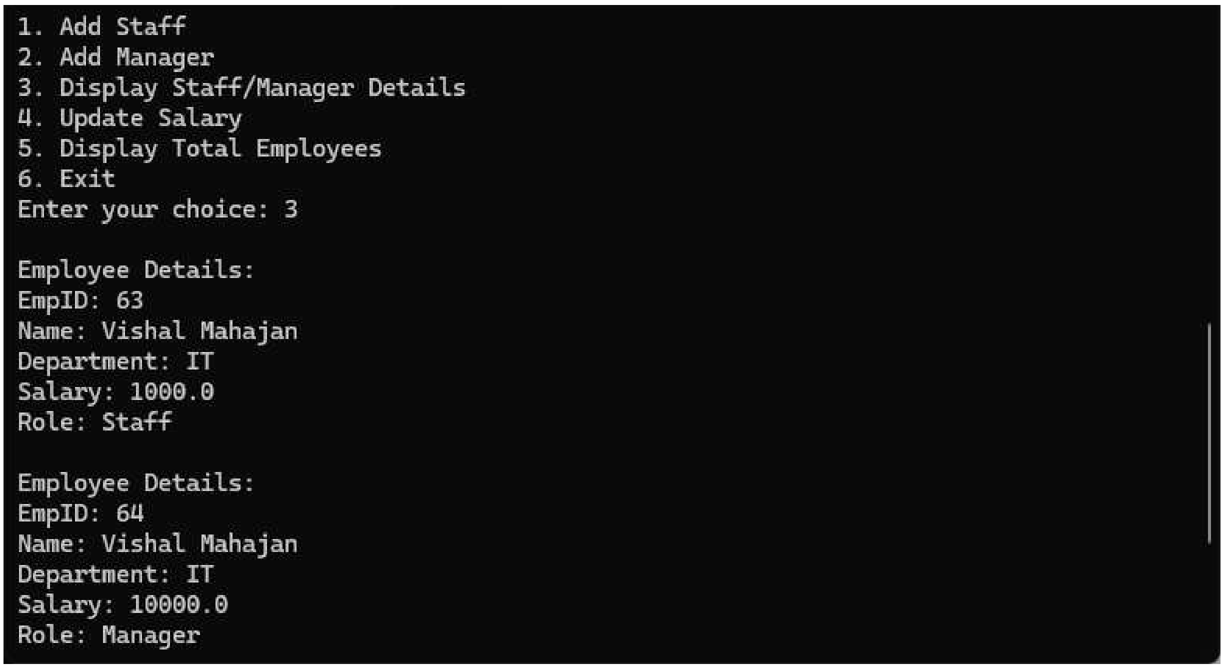
```
1. Add Staff
2. Add Manager
3. Display Staff/Manager Details
4. Update Salary
5. Display Total Employees
6. Exit
Enter your choice: 1
Enter Staff ID: 63
Enter Staff Name: Vishal Mahajan
Enter Department: IT
Enter Salary: 1000
```

2) Adding Manager



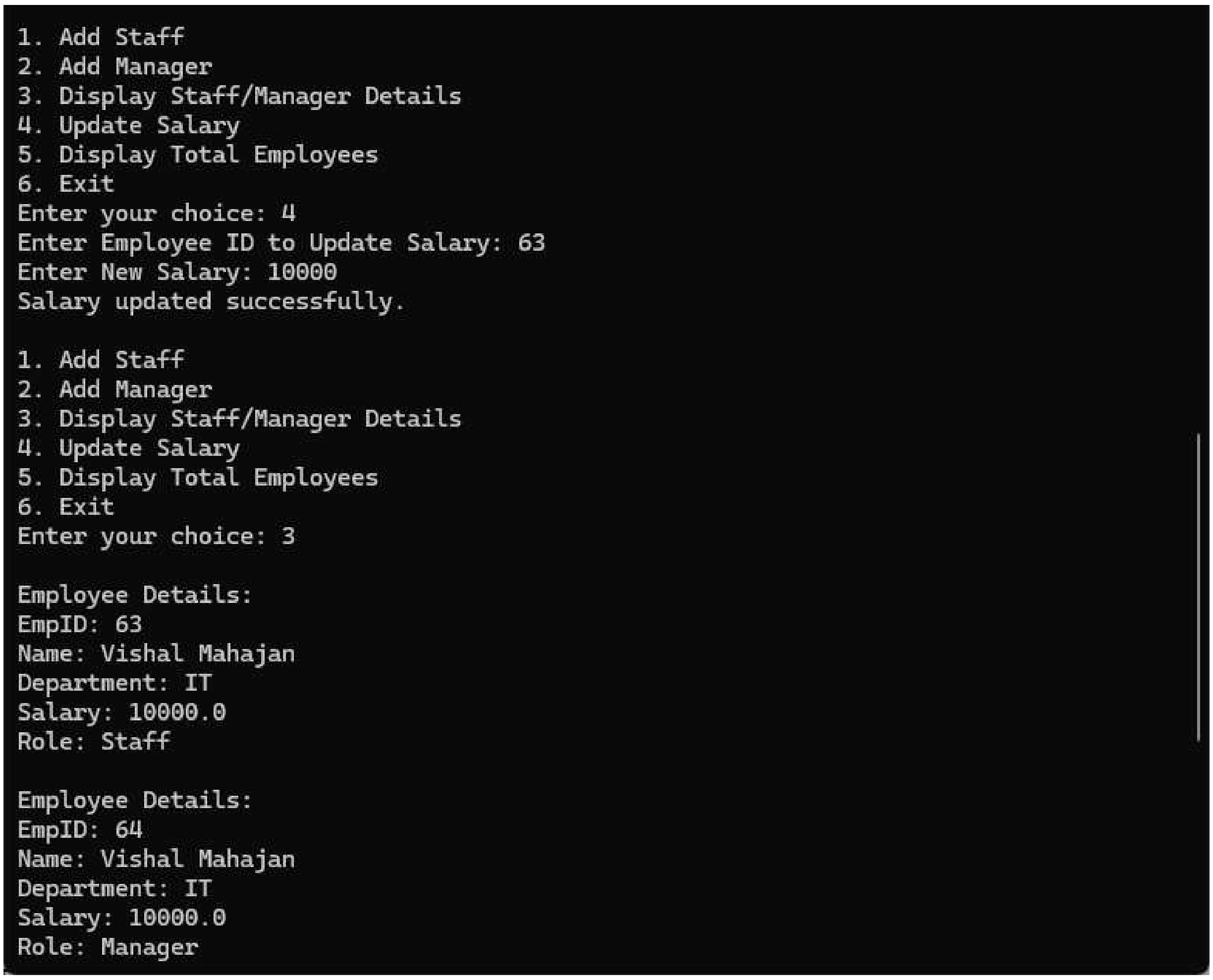
```
1. Add Staff
2. Add Manager
3. Display Staff/Manager Details
4. Update Salary
5. Display Total Employees
6. Exit
Enter your choice: 2
Enter Manager ID: 64
Enter Manager Name: Vishal Mahajan
Enter Department: IT
Enter Salary: 10000
```

3) Display Staff/Manager Details



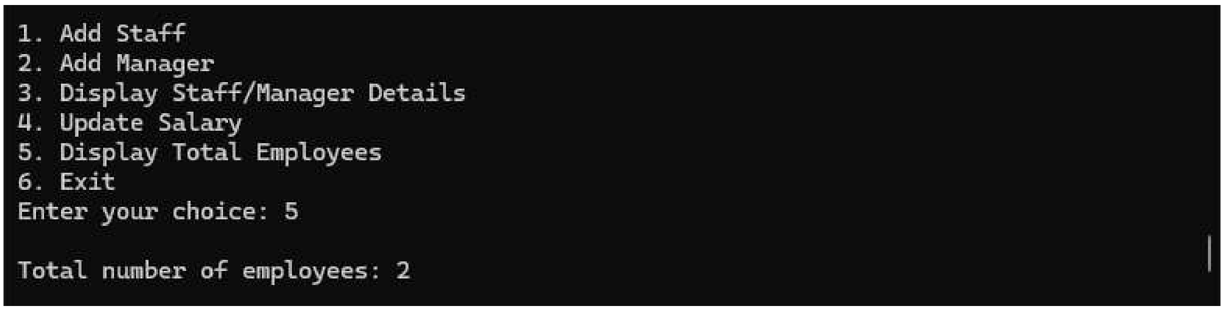
```
1. Add Staff
2. Add Manager
3. Display Staff/Manager Details
4. Update Salary
5. Display Total Employees
6. Exit
Enter your choice: 3

Employee Details:
EmpID: 63
Name: Vishal Mahajan
Department: IT
Salary: 1000.0
Role: Staff

Employee Details:
EmpID: 64
Name: Vishal Mahajan
Department: IT
Salary: 10000.0
Role: Manager
```

4) Update Salary



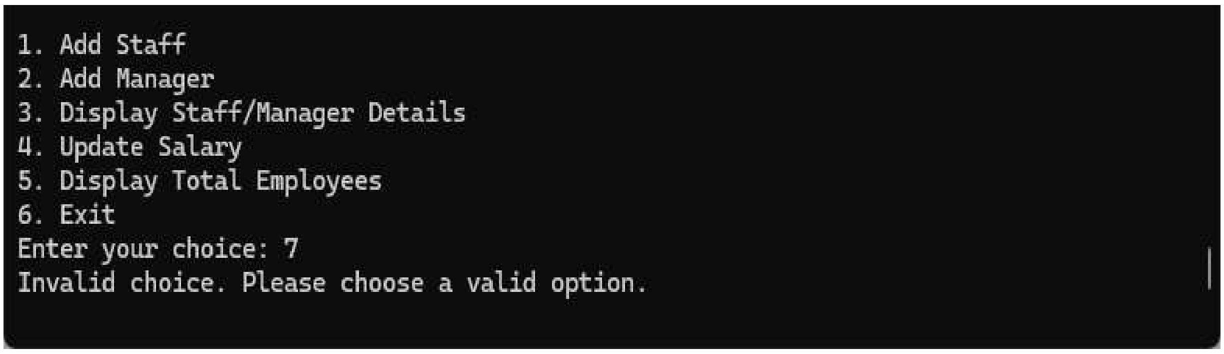
```
1. Add Staff
2. Add Manager
3. Display Staff/Manager Details
4. Update Salary
5. Display Total Employees
6. Exit
Enter your choice: 4
Enter Employee ID to Update Salary: 63
Enter New Salary: 10000
Salary updated successfully.

1. Add Staff
2. Add Manager
3. Display Staff/Manager Details
4. Update Salary
5. Display Total Employees
6. Exit
Enter your choice: 3

Employee Details:
EmpID: 63
Name: Vishal Mahajan
Department: IT
Salary: 10000.0
Role: Staff

Employee Details:
EmpID: 64
Name: Vishal Mahajan
Department: IT
Salary: 10000.0
Role: Manager
```

5) Display Total Employees



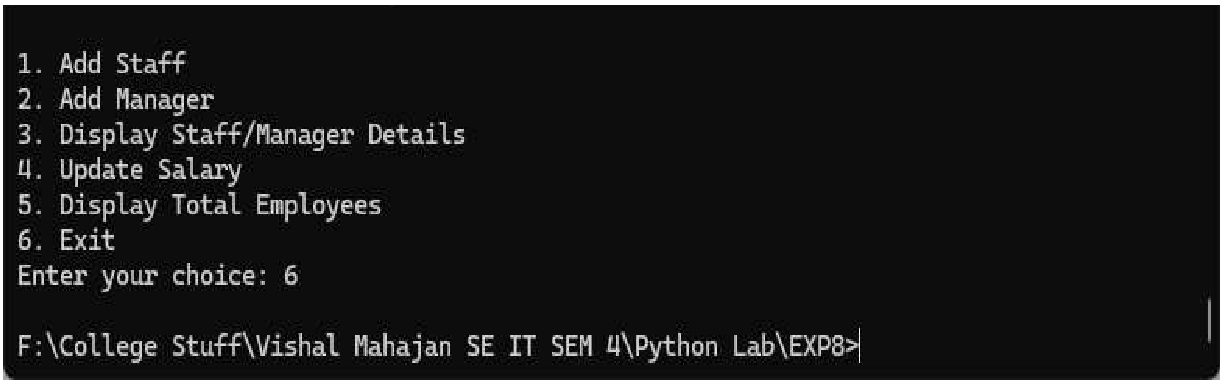
```
1. Add Staff
2. Add Manager
3. Display Staff/Manager Details
4. Update Salary
5. Display Total Employees
6. Exit
Enter your choice: 5

Total number of employees: 2
```

6) Invalid Choice



```
1. Add Staff
2. Add Manager
3. Display Staff/Manager Details
4. Update Salary
5. Display Total Employees
6. Exit
Enter your choice: 7
Invalid choice. Please choose a valid option.
```

7) Exit



```
1. Add Staff
2. Add Manager
3. Display Staff/Manager Details
4. Update Salary
5. Display Total Employees
6. Exit
Enter your choice: 6

F:\College Stuff\Vishal Mahajan SE IT SEM 4\Python Lab\EXP8>
```

Post- Lab Programs

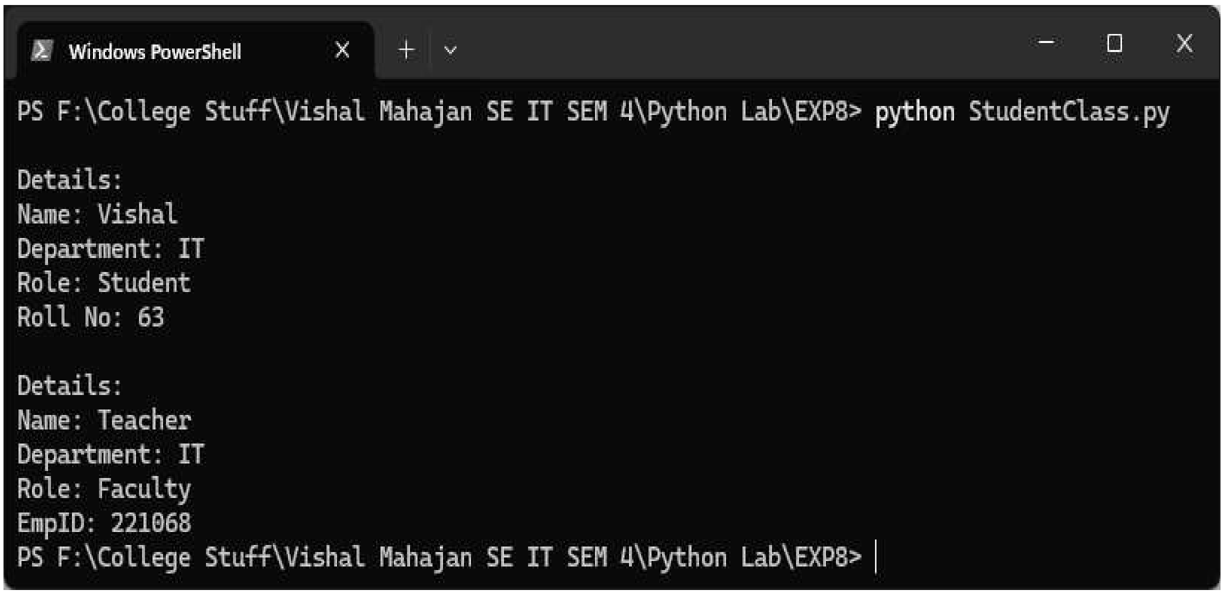
1) Write a Python program to declare a base class College having two derived classes student and faculty and display their details.

Code:

```
class College:
    def __init__(self, name, dept, role):
        self.name = name
        self.dept = dept
        self.role = role

    def display(self):
        print("\nDetails:")
        print("Name:", self.name)
        print("Department:", self.dept)
        print("Role:", self.role)




class Student(College):
    def __init__(self, name, dept, role, rollno):
        super().__init__(name, dept, role)
        self.rollno = rollno

    def display(self):
        super().display()
        print("Roll No:", self.rollno)
```

```python
class Faculty(College):
    def __init__(self, name, dept, role, empid):
        super().__init__(name, dept, role)
        self.empid = empid

    def display(self):
        super().display()
        print("EmpID:", self.empid)


Studentobj = Student("Vishal", "IT", "Student", 63)
Studentobj.display()

Facultyobj = Faculty("Teacher", "IT", "Faculty", 221068)
Facultyobj.display()
```

Output:



```
PS F:\College Stuff\Vishal Mahajan SE IT SEM 4\Python Lab\EXP8> python StudentClass.py

Details:
Name: Vishal
Department: IT
Role: Student
Roll No: 63

Details:
Name: Teacher
Department: IT
Role: Faculty
EmpID: 221068
PS F:\College Stuff\Vishal Mahajan SE IT SEM 4\Python Lab\EXP8>
```