TRIBHUVAN UNIVERSITY

INSTITUTE OF ENGINEERING

PULCHOWK CAMPUS

A REPORT ON

Programming in Python Language

SUBMITTED BY:

SUSHANT THAKUR (081BEL092)

SUBMITTED TO:

PRAKASH CHANDRA PRASAD SIR,

DEPARTMENT OF ELECTRONICS & COMPUTER ENGINEERING

QUESTION 1:

Define a class Student with attributes name, roll_number, and marks. Implement a method display_info() that prints the details of the student. Create an instance of Student and call the display_info() method to display the student's details..

CODE 1:

```
class Student:
    def __init__(self, name, roll_number, marks):
        self.name = name
        self.roll_number = roll_number
        self.marks = marks

    def display_info(self):
        print(f"Name: {self.name}")
        print(f"Roll Number: {self.roll_number}")
        print(f"Marks: {self.marks}")

s1 = Student("Rahul", 101, 88)
s1.display_info()
```

OUTPUT 1:

```
Name: Rahul
Roll Number: 101
Marks: 88
```

OUESTION 2:

Create a base class Animal with a method speak() that prints "Animal makes a sound". Derive a class Dog from Animal and override the speak() method to print "Dog barks". Instantiate the Dog class and call its speak() method.

CODE 2:

```
class Animal:
```

```
    def speak(self):
        print("Animal makes a sound")


class Dog(Animal):
    def speak(self):
        print("Dog barks")


# Example
pet = Dog()
pet.speak()
```

OUTPUT 2:

```
Dog barks
```

QUESTION 3:

Define a class BankAccount with private attributes account_number and balance. Implement methods to deposit and withdraw money, ensuring that the balance cannot go below zero. Provide a method to get the account details. Test the class by performing deposit and withdrawal operations.

CODE 3:

```
class BankAccount:
    def __init__(self, account_number, balance=0):
        self.__account_number = account_number
        self.__balance = balance


    def deposit(self, amount):
        self.__balance += amount
        print(f"Deposited ₹{amount}. New Balance = ₹{self.__balance}")
```

```
    def withdraw(self, amount):
        if amount <= self.__balance:
            self.__balance -= amount
            print(f"Withdrew ₹{amount}. New Balance = ₹{self.__balance}")
        else:
            print("Insufficient Balance!")

    def get_details(self):
        print(f"Account Number: {self.__account_number}, Balance: ₹{self.__balance}")



# Example
acc1 = BankAccount(123456789, 5000)
acc1.deposit(2000)
acc1.withdraw(3000)
acc1.withdraw(6000)
acc1.get_details()
```

OUTPUT 3:

```
Deposited ₹2000. New Balance = ₹7000
Withdrew ₹3000. New Balance = ₹4000
Insufficient Balance!
Account Number: 123456789, Balance: ₹4000
```

QUESTION 4:

Create a base class Shape with a method area(). Derive two classes Rectangle and Circle from Shape. Implement the area() method in both derived classes. Instantiate Rectangle and Circle, and demonstrate polymorphism by calling their area() methods.

CODE 4:

```
import math
```

```
class Shape:
    def area(self):
        pass

class Rectangle(Shape):
    def __init__(self, length, breadth):
        self.length = length
        self.breadth = breadth

    def area(self):
        return self.length * self.breadth

class Circle(Shape):
    def __init__(self, radius):
        self.radius = radius

    def area(self):
        return math.pi * self.radius * self.radius



# Example
shapes = [Rectangle(10, 5), Circle(7)]
for s in shapes:
    print("Area:", s.area())
```

OUTPUT 4:

```
Area: 50
Area: 153.93804002589985
```

QUESTION 5:

Define a class Person with attributes name and age. Derive a class Employee from Person with additional attributes employee_id and salary. Implement a method display_employee() in Employee that prints all the details. Create an instance of Employee and display the information.

CODE 5:

```
class Person:
    def __init__(self, name, age):
        self.name = name
        self.age = age


class Employee(Person):
    def __init__(self, name, age, employee_id, salary):
        super().__init__(name, age)
        self.employee_id = employee_id
        self.salary = salary


    def display_employee(self):
        print(f"Name: {self.name}, Age: {self.age}, ID: {self.employee_id}, Salary: ₹{self.salary}")


# Example
emp1 = Employee("abc", 28, "E101", 50000)
emp1.display_employee()
```

OUTPUT 5:

```
Name: abc, Age: 28, ID: E101, Salary: ₹50000
```

QUESTION 6:

Define a class Vector with attributes x and y. Overload the + operator to add two Vector objects. Implement the __add__() method and test it by adding two Vector instances.

CODE 6:

```
class Vector:
    def __init__(self, x, y):
        self.x = x
        self.y = y

    def __add__(self, other):
        return Vector(self.x + other.x, self.y + other.y)

    def display(self):
        print(f"({self.x}, {self.y})")


# Example
v1 = Vector(2, 5)
v2 = Vector(4, 7)
v3 = v1 + v2
v3.display()
```

OUTPUT 6:

```
(6, 12)
```

QUESTION 7:

Create a class Book with attributes title and author. Overload the __str__() method to return a string representation of the Book object in the format "Title by Author". Test this method by printing a Book instance.

CODE 7:

```
class Book:
    def __init__(self, title, author):
        self.title = title
        self.author = author

    def __str__(self):
        return f"'{self.title}' by {self.author}"



# Example
b1 = Book("The White Tiger", "Aravind Adiga")
print(b1)
```

OUTPUT 7:

```
'The White Tiger' by Aravind Adiga
```

QUESTION 8:

Define a class Time with attributes hours, minutes, and seconds. Overload the == operator to compare two Time objects for equality. Implement the __eq__() method and test it by comparing two Time instances.

CODE 8:

```
class Time:
    def __init__(self, hours, minutes, seconds):
        self.hours = hours
        self.minutes = minutes
        self.seconds = seconds

    def __eq__(self, other):
        return (self.hours == other.hours and
                self.minutes == other.minutes and
                self.seconds == other.seconds)



# Example
t1 = Time(10, 30, 15)
t2 = Time(10, 30, 15)
t3 = Time(9, 45, 20)

print(t1 == t2)  # True
print(t1 == t3)  # False
```

OUTPUT 8:

```
True
False
```

QUESTION 9:

Define a class Person with attributes name and age. Define another class Address with attributes street, city, and zipcode. Create a Contact class that contains an instance of Person and Address. Implement methods to display the contact details. Create a Contact object and display its information.

CODE 9:

```
class Person:
    def __init__(self, name, age):
        self.name = name
        self.age = age

class Address:
    def __init__(self, street, city, zipcode):
        self.street = street
        self.city = city
        self.zipcode = zipcode

class Contact:
    def __init__(self, person, address):
        self.person = person
        self.address = address

    def display_contact(self):
        print(f"Name: {self.person.name}, Age: {self.person.age}")
        print(f"Address: {self.address.street}, {self.address.city} - {self.address.zipcode}")



# Example
```

```
p1 = Person("Amit", 22)
a1 = Address("MG", "USA", "560001")
c1 = Contact(p1, a1)
c1.display_contact()
```

OUTPUT 9:

```
Name: Amit, Age: 22
Address: MG, USA - 560001
```

==========

GitHub: https://github.com/SushanThakur/2nd-sem-assignment/tree/master/lab-6/