24BIT196-Shreya Nair

AIM: To understand and implement object-oriented programming concepts using classes and objects in Python by writing programs that demonstrate encapsulation, operator overloading, and method definitions for various real-world scenarios.

HARDWARE & SOFTWARE REQUIREMENTS: Hardware:16GB RAM, Intel Processor(i9), Software: Python (Version 3.x), Google Colab (Cloud-based)

SYSTEM CONFIGURATION: Operating System: Windows 11, IDE: Google Colab

THEORY: Classes are user-defined data types in object-oriented programming that group data (attributes) and functions (methods) together.An object is an instance of a class thatholds actual values.

REFERENCES: Geeks for Geeks, Python Documentation: https://docs.python.org/3/

1) Write a program to create a class that represents Complex numbers containing real and imaginary parts and then use it to perform complex number addition, subtraction, multiplication, and division.

```
class Complex:
    def __init__(self, real, imag):
        self.real = real
        self.imag = imag

    def display(self):
        print(f"{self.real} + {self.imag}i")

    def __add__(self, other):
        return Complex(self.real + other.real, self.imag + other.imag)

    def __sub__(self, other):
        return Complex(self.real - other.real, self.imag - other.imag)

    def __mul__(self, other):
        real = self.real * other.real - self.imag * other.imag
        imag = self.real * other.imag + self.imag * other.real
        return Complex(real, imag)

    def __truediv__(self, other):
        denom = other.real**2 + other.imag**2
        real = (self.real * other.real + self.imag * other.imag) / denom
        imag = (self.imag * other.real - self.real * other.imag) / denom
        return Complex(real, imag)
c1 = Complex(7, 8)
c2 = Complex(10, 9)

print("Addition:")
(c1 + c2).display()

print("Subtraction:")
(c1 - c2).display()

print("Multiplication:")
(c1 * c2).display()

print("Division:")
(c1 / c2).display()
```

```
Addition:
17 + 17i
Subtraction:
-3 + -1i
Multiplication:
-2 + 143i
Division:
0.7845303867403315 + 0.09392265193370165i
```

2)Write a program that implements a Matrix class and performs addition, multiplication, and transpose operations on 3x3 matrices.

```
class Matrix:
    def __init__(self, data):
        self.data = data
```

```
        def display(self):
            for row in self.data:
                print(row)

        def add(self, other):
            result = [[self.data[i][j] + other.data[i][j] for j in range(3)] for i in range(3)]
            return Matrix(result)

        def multiply(self, other):
            result = []
            for i in range(3):
                row = []
                for j in range(3):
                    val = sum(self.data[i][k] * other.data[k][j] for k in range(3))
                    row.append(val)
                result.append(row)
            return Matrix(result)

        def transpose(self):
            result = [[self.data[j][i] for j in range(3)] for i in range(3)]
            return Matrix(result)

    m1 = Matrix([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
    m2 = Matrix([[1, 2, 3], [6, 5, 4], [7, 8, 9]])

    print("Addition:")
    m1.add(m2).display()

    print("Multiplication:")
    m1.multiply(m2).display()

    print("Transpose of Matrix 1:")
    m1.transpose().display()
```

```
Addition:
[2, 4, 6]
[10, 10, 10]
[14, 16, 18]
Multiplication:
[34, 36, 38]
[76, 81, 86]
[118, 126, 134]
Transpose of Matrix 1:
[1, 4, 7]
[2, 5, 8]
[3, 6, 9]
```

3) Write a program to create a class that can calculate the surface area and volume of a solid. The class should also accept relevant data

```
import math

class Cylinder:
    def __init__(self, radius, height):
        self.radius = radius
        self.height = height

    def surface_area(self):
        return 2 * math.pi * self.radius * (self.radius + self.height)

    def volume(self):
        return math.pi * self.radius**2 * self.height
c = Cylinder(4, 7)
print("Surface Area:", c.surface_area())
print("Volume:", c.volume())
```

```
Surface Area: 276.46015351590177
Volume: 351.85837720205683
```

4)Write a program to create a class that can calculate the perimeter/circumference and area of a regular shape. The class should accept relevant data.

```
import math

class Circle:
    def __init__(self, radius):
```

```
        self.radius = radius

    def area(self):
        return math.pi * self.radius**2

    def circumference(self):
        return 2 * math.pi * self.radius

c = Circle(9)
print("Area of Circle:", c.area())
print("Circumference of Circle:", c.circumference())
```

```
Area of Circle: 254.46900494077323
Circumference of Circle: 56.548667764616276
```

5) Write a program that creates and uses a Time class to perform various time arithmetic operations.

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

    def display(self):
        print(f"{self.hours:02d}:{self.minutes:02d}")

    def add(self, other):
        total_minutes = self.minutes + other.minutes
        extra_hours = total_minutes // 60
        total_minutes %= 60
        total_hours = self.hours + other.hours + extra_hours
        return Time(total_hours, total_minutes)

t1 = Time(5, 45)
t2 = Time(1, 45)

print("Time 1:")
t1.display()
print("Time 2:")
t2.display()

print("Sum of Time:")
t1.add(t2).display()
```

```
Time 1:
05:45
Time 2:
01:45
Sum of Time:
07:30
```

6) Write a program to create a Date class that has day, month, and year attributes. Define an overloaded == operator to compare two Date objects.

```
class Date:
    def __init__(self, day, month, year):
        self.day = day
        self.month = month
        self.year = year

    def __eq__(self, other):
        return (self.day == other.day and
                self.month == other.month and
                self.year == other.year)

d1 = Date(22, 4, 2025)
d2 = Date(22, 4, 2025)
d3 = Date(27, 5, 2025)

print("Dates are equal?" , d1 == d2)  # True
print("Dates are equal?" , d1 == d3)  # False
```

```
Dates are equal? True
Dates are equal? False
```

7) Create a class Weather that has a list containing weather parameters. Define an overloaded in operator to check whether an item is present in the list.

```
class Weather:
    def __init__(self, parameters):
        self.parameters = parameters

    def __contains__(self, item):
        return item in self.parameters

w = Weather(["temperature", "humidity", "wind", "pressure"])
print("humidity" in w)  # True
print("rainfall" in w)  # False
```

```
True
False
```

8) Implement a String class containing the following:

Overloaded += operator to perform string concatenation

Method toLower() to convert uppercase to lowercase

Method toUpper() to convert lowercase to uppercase

```
class String:
    def __init__(self, text):
        self.text = text

    def __iadd__(self, other):
        self.text += other.text
        return self

    def toLower(self):
        self.text = self.text.lower()

    def toUpper(self):
        self.text = self.text.upper()

    def display(self):
        print(self.text)

s1 = String("Hello")
s2 = String("Python")

s1 += s2
print("After concatenation:")
s1.display()

s1.toLower()
print("Lowercase:")
s1.display()

s1.toUpper()
print("Uppercase:")
s1.display()
```

```
After concatenation:
HelloPython
Lowercase:
hellopython
Uppercase:
HELLOPYTHON
```