24BIT167 - VANDITA NAWANI

Aim : To apply object oriented programming concepts using class and objects in python by writing programs that demostrate encapsulation, operator overloading and method definitons for various real-world scenario.

Hardware and Software requirements : hardware-16GB RAM, Intel Processor(i9), software: Python (Version 3.x.), Google Colab

System Configration : Operating System: Windows 11, IDE: Google Colab

Theory : Classes are user defined data types in object oriented programming that groups data and functions. An object is an instance of a class that holds values

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 __str__(self):
        sign = '+' if self.imag >= 0 else '-'
        return f"{self.real} {sign} {abs(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
        if denom == 0:
            raise ZeroDivisionError("Division by zero is undefined.")
        real = (self.real * other.real + self.imag * other.imag) / denom
        imag = (self.imag * other.real - self.real * other.imag) / denom
        return Complex(round(real, 2), round(imag, 2))


# Usage
c1 = Complex(4, 5)
c2 = Complex(2, -3)

print("First Complex Number: ", c1)
print("Second Complex Number:", c2)
print("Addition:      ", c1 + c2)
print("Subtraction:   ", c1 - c2)
print("Multiplication:", c1 * c2)
print("Division:      ", c1 / c2)
```

```
First Complex Number:  4 + 5i
Second Complex Number: 2 - 3i
Addition:       6 + 2i
Subtraction:    2 + 8i
Multiplication: 23 - 2i
Division:       -0.54 + 1.69i
```

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

```
class Matrix:
    def __init__(self, data):
        """
        Initializes a matrix with given data (a 3x3 list).
        """
        self.data = data

    def add(self, other):
        """
```

```
        """
        Adds two matrices and returns the result as a new matrix.
        """
        return Matrix([[self.data[i][j] + other.data[i][j] for j in range(3)] for i in range(3)])

    def multiply(self, other):
        """
        Multiplies two matrices and returns the result as a new matrix.
        """
        return Matrix([[sum(self.data[i][k] * other.data[k][j] for k in range(3)) for j in range(3)] for i in range(3)])

    def transpose(self):
        """
        Returns the transpose of the matrix.
        """
        return Matrix([[self.data[j][i] for j in range(3)] for i in range(3)])

    def show(self):
        """
        Displays the matrix in a readable format.
        """
        for row in self.data:
            print(row)

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

print("Matrix 1:")
m1.show()
print("Matrix 2:")
m2.show()

m3 = m1.add(m2)
print("\nAddition:")
m3.show()

m4 = m1.multiply(m2)
print("\nMultiplication:")
m4.show()

m5 = m1.transpose()
print("\nTranspose of Matrix 1:")
m5.show()
```

```
Matrix 1:
[1, 2, 3]
[4, 5, 6]
[7, 8, 9]
Matrix 2:
[9, 8, 7]
[6, 5, 4]
[3, 2, 1]

Addition:
[10, 10, 10]
[10, 10, 10]
[10, 10, 10]

Multiplication:
[30, 24, 18]
[84, 69, 54]
[138, 114, 90]

Transpose of Matrix 1:
[1, 4, 7]
[2, 5, 8]
[3, 6, 9]
```

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

```
import math

class Solid:
    def cube(self, side):
```

```
        area = 6 * side * side
        volume = side ** 3
        print("Cube - Surface Area:", area)
        print("Cube - Volume:", volume)

    def sphere(self, radius):
        area = 4 * math.pi * radius ** 2
        volume = (4/3) * math.pi * radius ** 3
        print("Sphere - Surface Area:", round(area, 2))
        print("Sphere - Volume:", round(volume, 2))


# Example usage
s = Solid()

print("1. Cube")
print("2. Sphere")

choice = int(input("Choose shape (1/2): "))

if choice == 1:
    side = float(input("Enter side: "))
    s.cube(side)
elif choice == 2:
    radius = float(input("Enter radius: "))
    s.sphere(radius)
else:
    print("Invalid choice.")
```

```
1. Cube
2. Sphere
Choose shape (1/2): 2
Enter radius: 4
Sphere - Surface Area: 201.06
Sphere - Volume: 268.08
```

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

```
import math

class Shape:
    def get_data(self):
        print("1. Square")
        print("2. Rectangle")
        print("3. Circle")
        self.choice = int(input("Choose shape (1/2/3): "))

        if self.choice == 1:
            self.side = float(input("Enter side of square: "))
        elif self.choice == 2:
            self.length = float(input("Enter length: "))
            self.breadth = float(input("Enter breadth: "))
        elif self.choice == 3:
            self.radius = float(input("Enter radius of circle: "))
        else:
            print("Invalid choice")
            self.choice = None

    def area(self):
        if self.choice == 1:
            return self.side ** 2
        elif self.choice == 2:
            return self.length * self.breadth
        elif self.choice == 3:
            return math.pi * self.radius ** 2

    def perimeter(self):
        if self.choice == 1:
            return 4 * self.side
        elif self.choice == 2:
            return 2 * (self.length + self.breadth)
        elif self.choice == 3:
            return 2 * math.pi * self.radius
```

```
# Example usage
s = Shape()
s.get_data()

if s.choice:
    print("Area:", round(s.area(), 2))
    print("Perimeter/Circumference:", round(s.perimeter(), 2))
```

```
1. Square
2. Rectangle
3. Circle
Choose shape (1/2/3): 3
Enter radius of circle: 3
Area: 28.27
Perimeter/Circumference: 18.85
```

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

```
class Time:
    def __init__(self, h=0, m=0, s=0):
        self.sec = h * 3600 + m * 60 + s

    def display(self):
        h = self.sec // 3600
        m = (self.sec % 3600) // 60
        s = self.sec % 60
        return f"{h:02d}:{m:02d}:{s:02d}"

    def add(self, other):
        return Time(0, 0, self.sec + other.sec)

    def subtract(self, other):
        return Time(0, 0, abs(self.sec - other.sec))


# Example usage
t1 = Time(2, 45, 30)
t2 = Time(1, 20, 50)

print("Time 1:", t1.display())
print("Time 2:", t2.display())
print("Add:", t1.add(t2).display())
print("Diff:", t1.subtract(t2).display())
```

```
Time 1: 02:45:30
Time 2: 01:20:50
Add: 04:06:20
Diff: 01:24:40
```

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

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

    def __eq__(self, other):
        return self.dmy == other.dmy

    def show(self):
        print(f"{self.dmy[0]:02d}/{self.dmy[1]:02d}/{self.dmy[2]}")

# Example usage
d1 = Date(24, 4, 2025)
d2 = Date(24, 4, 2025)
d3 = Date(1, 5, 2025)

d1.show()
d2.show()
d3.show()

print("d1 == d2:", d1 == d2)
```

```
print("d1 == d3:", d1 == d3)
```

```
24/04/2025
24/04/2025
01/05/2025
d1 == d2: True
d1 == d3: False
```

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

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

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

    def show(self):
        print("Weather Parameters:", self.data)

# Example usage
w = Weather(["temperature", "humidity", "pressure", "wind", "visibility"])
w.show()

# Check if 'humidity' is a weather parameter
print("humidity" in w)    # True
print("rainfall" in w)    # False
```

```
Weather Parameters: ['temperature', 'humidity', 'pressure', 'wind', 'visibility']
True
False
```

Implement a String class containing the following functions: a. Overloaded += operator function to perform string concatenation b. Method toLower() to convert upper case letters to lower case. c. Method toUpper() to convert lower case letters to upper case.

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

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

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

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

    def show(self):
        print(self.value)

# Example usage
s = String("Hello")
s += " World"
s.show()  # Output: Hello World
s.toLower()
s.show()  # Output: hello world
s.toUpper()
s.show()  # Output: HELLO WORLD
```

```
Hello World
hello world
HELLO WORLD
```