PAT TASK - 4
Day 5 OOPS with Class and object

Using the python object oriented programming scheme (oops) kindly complete the following tasks which is gives as below:

1) **Create a python class called circle with constructor which will take a list as an argument for the task. The list is [10,501,22,37,100,999,87,351].**

Python class **`Circle`** that takes a list as an argument in its constructor. The class is designed in an object-oriented programming style (OOPS).

```
class Circle:
    def __init__(self, radii):
        """Constructor that initializes the Circle object with a list of radii."""
        if isinstance(radii, list):
            self.radii = radii
        else:
            raise ValueError("Input must be a list")

    def __repr__(self):
        """Representation method for Circle object."""
        return f"Circle(radii={self.radii})"

# Usage
circle = Circle([10, 501, 22, 37, 100, 999, 87, 351])
print(circle)
```

`__init__` Method: This is the constructor method, which initializes the object when it's created. It takes one argument `radii`, which is expected to be a list.
`__repr__` Method: This method is used to provide a string representation of the object, making it easier to understand when printed.

2) create proper member variable inside the task if requied and use them when necessary for example for this task create a class private variable named pi = 1.141

`Circle` class with a proper private class variable `_pi` set to `1.141`. The class now uses this variable to calculate the area and circumference for each circle.

```
class Circle:
    # Private class variable
    _pi = 1.141

    def __init__(self, radii):
        """Constructor that initializes the Circle object with a list of radii."""
        if isinstance(radii, list):
            self._radii = radii
        else:
            raise ValueError("Input must be a list")

    def calculate_areas(self):
        """Method to calculate the area of each circle using the custom value of pi."""
        return [self._pi * (r ** 2) for r in self._radii]

    def calculate_circumferences(self):
        """Method to calculate the circumference of each circle using the custom value of pi."""
        return [2 * self._pi * r for r in self._radii]

    def __repr__(self):
        """Representation method for Circle object."""
        return f"Circle(radii={self._radii})"

# Usage
circle = Circle([10, 501, 22, 37, 100, 999, 87, 351])

# Calculating areas
areas = circle.calculate_areas()
print(f"Areas: {areas}")

# Calculating circumferences
circumferences = circle.calculate_circumferences()
print(f"Circumferences: {circumferences}")
```

Private Class Variable `_pi`: This is a class-level private variable intended to store a custom value of π\piπ. It is used in calculations within the class.

Private Member Variable `_radii`: This is a private instance variable that stores the list of radii passed to the constructor.

`calculate_areas` Method: This method calculates the area for each radius using the formula area=pi×r2\text{area} = \text{pi} \times r^2area=pi×r2.

`calculate_circumferences` Method: This method calculates the circumference for each radius using the formula circumference=2×pi×r\text{circumference} = 2 \times \text{pi} \times circumference=2×pi×r.

`__repr__` Method: Provides a string representation of the `Circle` object.

Areas: [114.1, 286501.641, 552.9820000000001, 1555.3690000000001, 11410.0, 1137280.3590000001, 8644.569000000001, 141590.541]

Circumferences: [22.82, 1143.282, 50.204, 84.414, 228.20000000000002, 2281.858, 198.26799999999997, 800.0919999999999]

3) From the given list create two class methods Area and perimeter which will be going to calculate the area and perimeter.

```
class Circle:
    # Private class variable
    _pi = 1.141

    def __init__(self, radii):
        """Constructor that initializes the Circle object with a list of radii."""
        if isinstance(radii, list):
            self._radii = radii
        else:
            raise ValueError("Input must be a list")

    @classmethod
    def calculate_area(cls, radius):
        """Class method to calculate the area of a circle for a given radius."""
        return cls._pi * (radius ** 2)

    @classmethod
    def calculate_perimeter(cls, radius):
        """Class method to calculate the perimeter (circumference) of a circle for a given radius."""
        return 2 * cls._pi * radius
    def areas(self):
        """Method to calculate the area of each circle in the list."""
        return [self.calculate_area(r) for r in self._radii]

    def perimeters(self):
        """Method to calculate the perimeter of each circle in the list."""
        return [self.calculate_perimeter(r) for r in self._radii]

    def __repr__(self):
        """Representation method for Circle object."""
        return f"Circle(radii={self._radii})"

# Usage
circle = Circle([10, 501, 22, 37, 100, 999, 87, 351])

# Calculating areas using the class method
areas = circle.areas()
print(f"Areas: {areas}")

# Calculating perimeters using the class method
perimeters = circle.perimeters()
print(f"Perimeters: {perimeters}")
```

Output

```
Areas: [114.1, 286501.641, 552.9820000000001, 1555.3690000000001, 11410.0,
1137280.3590000001, 8644.569000000001, 141590.541]
Perimeters: [22.82, 1143.282, 50.204, 84.414, 228.20000000000002, 2281.858,
198.26799999999997, 800.0919999999999]
```

the areas and perimeters of circles with the given radii `[10, 501, 22, 37, 100, 999, 87, 351]` calculated using the custom value of π\piπ set to `1.141`.

Class Variable `_pi`: This private variable holds the custom value of π\piπ used throughout the calculations.
Class Method `calculate_area`: This method is a `@classmethod` that calculates the area for a single radius using the formula area=pi×r2\text{area} = \text{pi} \times r ^2area=pi×r2.

Class Method `calculate_perimeter`: This method is a **`@classmethod`** that calculates the perimeter (circumference) for a single radius using the formula perimeter=2×pi×r\text{perimeter} = 2 \times \text{pi} \times r perimeter=2×pi×r.
Instance Method `areas`: Iterates over the list of radii and uses the `calculate_area` class method to compute the area for each radius.