Lab 9

July 22, 2024

Dylan Liesenfelt

1 Exercise: Defining a Polygon Class

Define a Polygon class with attributes for the number of sides (num_sides) and the size of each side (side_length). Implement the ___init___ method to allow for creating instances of a Polygon with specific attributes.

[ ]:
```
class Polygon:
    def __init__(self, num_sides, side_lengths):
        self.sides = num_sides
        self.length = side_lengths

p1 = Polygon(3,1)
p2 = Polygon(6,3)

print(f'Polygon 1, sides:{p1.sides}, length:{p1.length}')
print(f'Polygon 2, sides:{p2.sides}, length:{p2.length}')
```

```
Polygon 1, sides:3, length:1
Polygon 2, sides:6, length:3
```

2 Exercise: Adding Methods to the Polygon Class

Implement an instance method perimeter that calculates the perimeter of the polygon. Also, implement an instance method description that returns a string describing the polygon, including the number of sides and its perimeter.

[ ]:
```
class Polygon:
    def __init__(self, num_sides, side_lengths):
        self.sides = num_sides
        self.length = side_lengths

    def perimeter(self):
        return self.sides * self.length

    def description(self):
```

```
        print(f'This Polygon has {self.sides} sides, each of a length of {self.
 ↪length}, and a perimeter of {self.perimeter()}')

p3 = Polygon(4,1)
p3.description()
```

```
This Polygon has 4 sides, each of a length of 1, and a perimeter of 4
```

3 Exercise: Introducing Class Variables

Introduce a class variable polygon_type that describes the type of polygon (e.g., "Polygon"). Modify the description method to include the polygon_type in the description.

[ ]:
```
class Polygon:
    polygon_type = 'Generic'

    def __init__(self, num_sides, side_lengths):
        self.sides = num_sides
        self.length = side_lengths

    def perimeter(self):
        return self.sides * self.length

    def description(self):
        print(f'A {self.polygon_type} with {self.sides} sides, each of a length␣
 ↪of {self.length}, and a perimeter of {self.perimeter()}')

p4 = Polygon(5,1)
p4.description()
```

```
A Generic with 5 sides, each of a length of 1, and a perimeter of 5
```

4 Exercise : Inheritance in Polygons and Polymorphism with

For these exercises, let's introduce a Circle class to use pi, and demonstrate polymorphism with an overridden description method in both Triangle and Square classes.

[ ]:
```
import math

class Polygon:
    polygon_type = "Generic Polygon"

    def __init__(self, num_sides, side_length):
        self.num_sides = num_sides
        self.side_length = side_length

    def perimeter(self):
```

```
        return self.num_sides * self.side_length

    def description(self):
        return f'A {self.polygon_type} with {self.num_sides} sides, each of␣
 ↪length {self.side_length}, and a perimeter of {self.perimeter()}.'

class Triangle(Polygon):
    polygon_type = "Triangle"

    def __init__(self, side_length):
        super().__init__(3, side_length)

class Square(Polygon):
    polygon_type = "Square"

    def __init__(self, side_length):
        super().__init__(4, side_length)

    def area(self):
        return self.side_length ** 2

    def description(self):
        return f'A {self.polygon_type} with {self.num_sides} sides, each of␣
 ↪length {self.side_length}, a perimeter of {self.perimeter()}, and an area of␣
 ↪{self.area()}.'

class Circle(Polygon):
    polygon_type = "Circle"

    def __init__(self, radius):
        self.radius = radius

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

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

    def description(self):
        return f'A {self.polygon_type} with a radius of {self.radius}, a␣
 ↪perimeter (circumference) of {self.perimeter()}, and an area of {self.
 ↪area()}.'

p5 = Triangle(3)
p6 = Square(3)
p7 = Circle(3)
print(p5.description())
```

```
print(p6.description())
print(p7.description())
```

```
A Triangle with 3 sides, each of length 3, and a perimeter of 9.
A Square with 4 sides, each of length 3, a perimeter of 12, and an area of 9.
A Circle with a radius of 3, a perimeter (circumference) of 18.84955592153876,
and an area of 28.274333882308138.
```

5 Exercise: Integrating Learned Concepts

Create a class called GeometryLibrary with methods to list, add and remove polygons.

[ ]:
```
class GeometryLibrary:
    def __init__(self):
        self.polygons = []

    def add_polygon(self, polygon):
        self.polygons.append(polygon)

    def remove_polygon(self, polygon):
        self.polygons.remove(polygon)

    def list_polygons(self):
        descriptions = [polygon.description() for polygon in self.polygons]
        return '\n'.join(descriptions)

library = GeometryLibrary()
triangle = Triangle(3)
square = Square(4)
circle = Circle(5)
library.add_polygon(triangle)
library.add_polygon(square)
library.add_polygon(circle)
print(library.list_polygons())
```

```
A Triangle with 3 sides, each of length 3, and a perimeter of 9.
A Square with 4 sides, each of length 4, a perimeter of 16, and an area of 16.
A Circle with a radius of 5, a perimeter (circumference) of 31.41592653589793,
and an area of 78.53981633974483.
```