COP 4045 – Dr. Marques **Summer 2021**

Assignment #8

Assignment Overview

In this assignment you will create a simple program for computing and displaying properties of 2D shapes.

Project Specification

This is a group assignment.
Students are encouraged (but not required) to work in groups of max 3 students.

Ideally, the group should be organized around three main tasks / duties:

- Design of the solution ("architect" role)
- Coding of the solution ("developer" role)
- Documentation of the solution ("reporter" role)

You are <u>required</u> to indicate in your submission "who did what" and document the entire process, from sketching the original plans and dividing up the tasks all the way to polishing the interface, testing the solution, and preparing the report.

In this assignment you will design an OOP solution to 2D shape representation and computation of basic properties (area, perimeter) in Python.

Requirements

You will start by designing a class **`Shape`** with very few attributes, essentially an initializer and a string representation.

You will then design and implement derived (inherited) classes for at least five shapes: circle, triangle, rectangle, pentagon, and hexagon.

The class **`Circle`** should be derived directly from the class **`Shape`**; other shapes should be implemented as derived from a class **`Polygon`**, which is a child of class **`Shape.`**

You can choose to handle squares as a special case of rectangles or create a class **`Square.`**

See this[1] for ideas.

The initializer for each class should handle the initialization process differently:

- Triangles must have their sides checked to see if they make a triangle indeed.
- Rectangles can have any (positive-valued floating-point) value for length and width.
- Pentagons and hexagons can be assumed to be regular polygons, i.e., specifying the length of one side should be enough.

[1] https://johnfoster.pge.utexas.edu/numerical-methods-book/PythonIntro_ObjectOrientedProgramming.html

Your program should demonstrate that it can produce correct results for the driver program below[2] (available on Canvas):

```python
# Driver program for A8 – COP 4045

## Create variables

my_circle = Circle(2)
# create a circle of radius = 2

my_triangle = Triangle(3, 1.7, 4.9)
# attempt to create a triangle with "incorrect" values
# should produce error message

my_triangle = Triangle(3, 7, 4.6)
# create a triangle passing the length of each side

my_rectangle = Rectangle(3, 4.5)
# create a rectangle of sides 3 and 4.5

my_pentagon = Pentagon(3)
# create a pentagon with sides of equal length

my_hexagon = Hexagon(3)
# create a hexagon with sides of equal length

###########################################

## Print area and perimeter for each variable

print(my_circle.area)
print(my_circle.perimeter)

print(my_triangle.area)
print(my_triangle.perimeter)

print(my_rectangle.area)
print(my_rectangle.perimeter)

print(my_pentagon.area)
print(my_pentagon.perimeter)

print(my_hexagon.area)
print(my_hexagon.perimeter)
```

[2] You are allowed (and encouraged!) to group the print statements and format them nicely.

Notes and Hints:

- Follow the "cardinal rules" of programming in Python (as per the textbook).
- The `Circle` class (textbook, chapter 16) might be useful.
- If you choose to assign (x,y) coordinates to the shapes (*not required*), the `Point` class (textbook, chapter 11) might be useful.
- Start by breaking the program down into parts and solve smaller problems before producing the final solution.
- Try to handle special cases and prevent runtime errors to the best of your knowledge.
- Don't overdo it!

Deliverables

You must submit (via Canvas):

- The **link**[3] to a Jupyter notebook on Google Colab containing your entire solution. It must include:
    - Header:
        - Team members' names, date, course name + code, assignment number
    - Your source code
    - Results (of multiple runs) + meaningful comments
    - Plots (*optional*)
    - Figures (*optional*)
    - References (including your "sources of inspiration" for the code)
    - Comments (README-like): installation instructions, dependencies, etc.
    - Project notes (describing what my TA and I cannot see by looking at your source code and/or running your program).
        - Examples: design decisions, documented limitations, future improvements, etc.
    - Your **Conclusion** with a summary of your insights and lessons learned.

Bonus opportunities:

There are <u>no</u> bonus opportunities (unless, of course, you guess my zoom background in related lectures). ☺

[3] When sharing the link to your Google Colab notebook, choose the 'anyone with the link can open it' option, i.e., **don't make it specific to a domain** (such as fau.edu) **or individual** (instructor or TA).