Lab 5

CS161

For each of the problems below be sure to design your solution before you write any Python code. Follow the software development stages:

1. Analyze the problem
2. Determine the specifications for your solution
3. Create a design for your solution in pseudocode
4. Implement your design in Python
5. Test/debug your solution

Each Python module you turn in must begin with a block, called a *docstring*, that looks like the example below (see PEP 257 for more details). The module docstring must follow the shebang and coding statements in your module; i.e. it must be the first actual line of code in the module.

In your module docstring show the work you did following the software development stages above. Make notes about the problem as you analyze it; write specifications for what your program must do to find a solution; write your program out in pseudocode *before* you start to write Python; and design a set of test data that you can use to prove your program is producing correct results. Only after you've done all of this should you try to translate your pseudocode to Python.

```
#! /usr/bin/env python3
# coding=utf-8

"""
Brief, prescriptive, one sentence description of what the module does.

A more detailed explanation of what the module does. Use complete
sentences and proper grammar. Treat this as if it is documentation someone
else will be reading (because it is). Include any notes from your problem
analysis as well as your program specifications here.

Pseudo code, to be written before any actual Python code.

Assignment or Lab #

Firstname Lastname
Firstname Lastname of partner (if applicable)
"""
<two blank lines>
<your code>
```

Every function or method you write must also include its own docstring. That header will also use triple quotes and begin with a prescriptive, one sentence description of what the function or method does. If the function or method takes arguments, they should be described as well as any non-obvious return value(s). See examples below or PEP 257 for more details).

```
def print_greeting( ):
    """Print a greeting to the user."""
    print(“Hello out there in userland!”)



def hypotenuse(side_a, side_b):
    """
    Calculates the length of the hypotenuse of a right triangle using
    the formula c^2 = a^2 + b^2.

    side_a - the length of side A of the triangle
    side_b - the length of side B of the triangle
    """
    c = a**2 + b**2
    return math.sqrt(c)
```

Pair Programming

In this lab we are going to use **pair programming**.

PICK A NEW LAB PARTNER – SOMEONE YOU HAVEN’T DONE A LAB WITH BEFORE

Pair programming is an agile software development technique in which two programmers work together at one workstation. One, the ***driver****, writes code while the other, the* ***observer*** *or* ***navigator****, reviews each line of code as it is typed in. The two programmers switch roles frequently.*

While reviewing, the observer also considers the "strategic" direction of the work, coming up with ideas for improvements and likely future problems to address. This is intended to free the driver to focus all of their attention on the "tactical" aspects of completing the current task, using the observer as a safety net and guide.

Choose one partner in the lab to work with. You can pair program in one of two ways:

1. Two people can work at one computer, occasionally switching the driver and observer roles. It is *critical* that both members be engaged in the work. You should take turns being in the driver and observer roles with only the driver typing and the observer making comments or driving the direction of the code.
2. Two people can work at separate machines using Visual Studio Live Share. Live Share, available as a free plug-in for VS Code and already installed on the school’s Surface devices, enables you and your partner to collaborate on the same codebase without the need any difficult configuration. When you share a collaborative session, your partner sees the context of the workspace in their editor. This means they can read the code you shared without having to clone a repo or install any dependencies your code relies on. They can use rich language

features to navigate within the code; not only just opening other files as text but using semantic analysis-based navigation like Go to Definition or Peek. Again, you should take turns being in the driver and observer roles with only the driver typing and the observer making comments or driving the direction of the code.

Exercise 1

An archery target consists of a central circle of yellow surrounded by concentric rings of red, blue, black and white. Each ring has the same width, which is the same as the radius of the yellow circle. Write a program that draws such a target. *Hint*: Objects drawn later will appear on top of objects drawn earlier.

Write and save your code in a file named `archery.py` for full credit.

Exercise 2

Write a program that displays information about a rectangle drawn by the user.

Input: Two mouse clicks for the opposite corners of a rectangle.

Output: Draw the rectangle.

Print the perimeter and area of the rectangle.

Formulas:

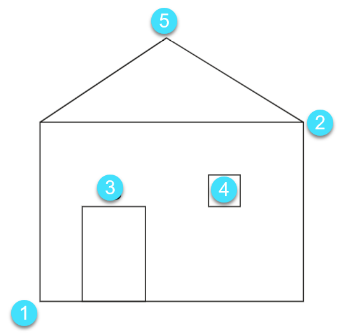
$$area = length \times width$$

$$perimeter = 2(length + width)$$

Write and save your code in a file named `rectangle.py` for full credit.

Exercise 3

Write a program that allows the user to draw a simple house using five mouse clicks. The first two clicks will be the opposite corners of the rectangular frame of the house. The third click will indicate the center of the top edge of a rectangular door. The door should have a total width that is $^1/_5$ of the width of the house frame. The sides of the door should extend from the corners of the top down to the bottom of the frame. The fourth click will indicate the *center* of a square window. The window is half as wide as the door. The last click will indicate the peak of the roof. The edges of the roof will extend from the point at the peak to the corners of the top edge of the house frame.



Submitting Your Lab

Put all the Python files you wrote or downloaded for this lab in a folder named `lab-4.py`, then copy it and all its contents into the folder you've shared with your instructor.