Introduction

Today in studio you will complete several tasks that will help you get closer to completing your assignment. You should be prepared to engage with your classmates for the entire studio time. The main goals for today are:

1. Practice getting input and working with variables
2. Practice calling and using functions
3. Practice using the command line
4. More practice with If Statements

Deliverables: You will submit your completed assignment as a .pdf at the end of the studio session. Include the following:

1. Did you prefer specifying the shapes in the code (part 1), or using the interactive method (part 2) for plotting? Why?
   a. What are some of the benefits of each method?
   b. Some of the cons?
2. In part 2, we used the command line to
3. Include your PNGs from parts 1,2 and 3
   a. Which PNG is your favorite? Why?

Part 1: Plotting shapes (~45 min)

First, download eng103_studio3.py from Canvas if you haven't already. Open this file and take a look at the functions for drawing shapes (lines 24-47). Take a look at the inputs of the function and make sure you understand what the functions need to run.

Now scroll down and find the function studio3_part1(). We will be editing this function first to plot some shapes.

As an example to help you see how to call functions for plotting shapes, we have plotted a lavender circle of radius 1 centered at (0,0). Look through lines 127-137 and read the comments to get a better understanding of the steps to plot a shape. Now, you will plot shapes yourself.

1. First choose your shape and call one of the following functions:
   a. Be sure to fill in the x,y center position and the dimensions
   b. RECTANGLE
      ```
      self.draw_static_rectangle(center x, center y, height, width)
      ```
   c. CIRCLE
      ```
      self.draw_static_circle(center x, center y, radius)
      ```
   d. ELLIPSE
      ```
      self.draw_static_ellipse(center x, center y, height, width, tilt)
      ```
2. Next, decide how see-through you want your shape
   a. 0 is fully see-through and 1 is fully solid

   b. Set alpha equal to that value in the following function:
      ```
      p = PatchCollection(self.patches, alpha=0.5)
      ```

3. Then, chose the color you want your shape to be
   a. Replace 'lavender' in the example with your color in the following function:
      ```
      p.set_facecolor('lavender')
      ```

4. Finally, copy the following two lines, but don't change anything about them
   a. The first function is adding the shape you just specified to the list of shapes to plot
      ```
      self.ax.add_collection(p)
      ```
   b. The second line is clearing your patches so that you don't overwrite your color and opacity choices
      ```
      self.patches = []
      ```

Your final block of code to plot a shape will look something like the following:

```
self.draw_static_SHAPE(center x, center y, DIMENSIONS)
p = PatchCollection(self.patches, alpha=FILL THIS IN)
p.set_facecolor(FILL THIS IN)
self.ax.add_collection(p)
self.patches = []
```

After you edit this code, change the plot title to have your name and save your PNG as something unique. Remember to change it each time you run the code, or it will only save the most recent PNG. Or, if you would rather save it manually on the plot you can comment out this line. When you run your code, make sure the main function is running studio3.studio3_part1().

Plot at least one of each shape and try different placements, colors, and opacities. The order you plot the shapes in will also matter. The shapes plotted first will show up *behind* the following shapes if there is overlap. Think of it like layering up shapes as you plot. This allows you to do some cool color things if your shapes are not 100% opaque.

Save your favorite PNG and put it in your PDF.

Part 2: Plotting shapes interactively (~45 min)

Now that you understand the inputs for the functions to plot shapes works in this code, we are going to plot them interactively using mouse clicks and input from the terminal.

Navigate to the interactive_plot() function. Go through the code and read the comments and try to understand what the code is doing. In summary, the code waits for a mouse click, and saves the x,y coordinates of the click as x and y. We will use the x and y variables later when calling our shape plotting functions.

Around line 68 you will see that an If/Elif/Else statement begins that looks at what the user typed in the terminal, asks the user for relevant terminal input, and plots the correct shape. You will be adding an elif statement for the circle shape.

Let's look at the rectangle if statement as an example:

1. First, we set our condition:

```
if shape == 'rectangle':
```

2. Next, we use a print statement to prompt the user to enter the relevant dimensions in the terminal. For rectangle, we need width and height.

```
print("Width?")
```

3. Then, we save what the user types in the terminal as a float variable called width.

```
width = float(input())
```

4. Now we do the same for height. First, prompt the user with a print statement.

```
print("Height?")
```

5. And save the input as a float variable called height.

```
height = float(input())
```

6. Finally, we call the function to plot a rectangle and enter the needed variables. For rectangle, a little math has to be done to get the click point to be the center of the rectangle, but this is not necessary for the ellipse and the circle.

```
self.draw_static_rectangle(x-(width/2), y-(height/2), height, width)
```

Now scroll down to where the code says ##### EDIT HERE ##### and write the elif statement for the circle. Remember that the circle function inputs are center x, center y, and the radius. We already have center x and center y from the mouse click, but we need the user to input the radius into the terminal.

After writing your elif statement, scroll down to the bottom of the interactive_plot() function and change the name of your PNG to save it. Or, if you would rather save it manually on the plot you can comment out this line.

Now, we need to change the plot title. Scroll down to the studio3_part2() function and find the plt.title() function and edit it so it has your name.

Finally, scroll down to the main function and change studio3.studio3_part1() to studio3.studio3_part2() and run the code!

Plot at least one of each shape and try different placements, colors, and opacities. The order you plot the shapes in will also matter. The shapes plotted first will show up *behind* the following shapes if there is overlap. Think of it like layering up shapes as you plot. This allows you to do some cool color things if your shapes are not 100% opaque.

Save your favorite PNG and put it in your PDF.

Part 3: Make fun pictures (~30 min)

Pick your favorite method of plotting shapes and mess around with the types of shapes, placement, colors, opacity, and plotting order of the shapes to make some fun pictures! Save some of your favorite combinations and put them in your PDF to submit.

Include in your PDF:

1. Did you prefer specifying the shapes in the code (part 1), or using the interactive method (part 2) for plotting? Why?
    a. What are some of the benefits of each method?
    b. Some of the cons?
2. In part 2, we used the command line to take input from the user and turn it into something visual. How could you utilize command line input in a larger art installation, especially an interactive one?
3. Include your PNGs from parts 1,2 and 3
    a. Which PNG is your favorite? Why?