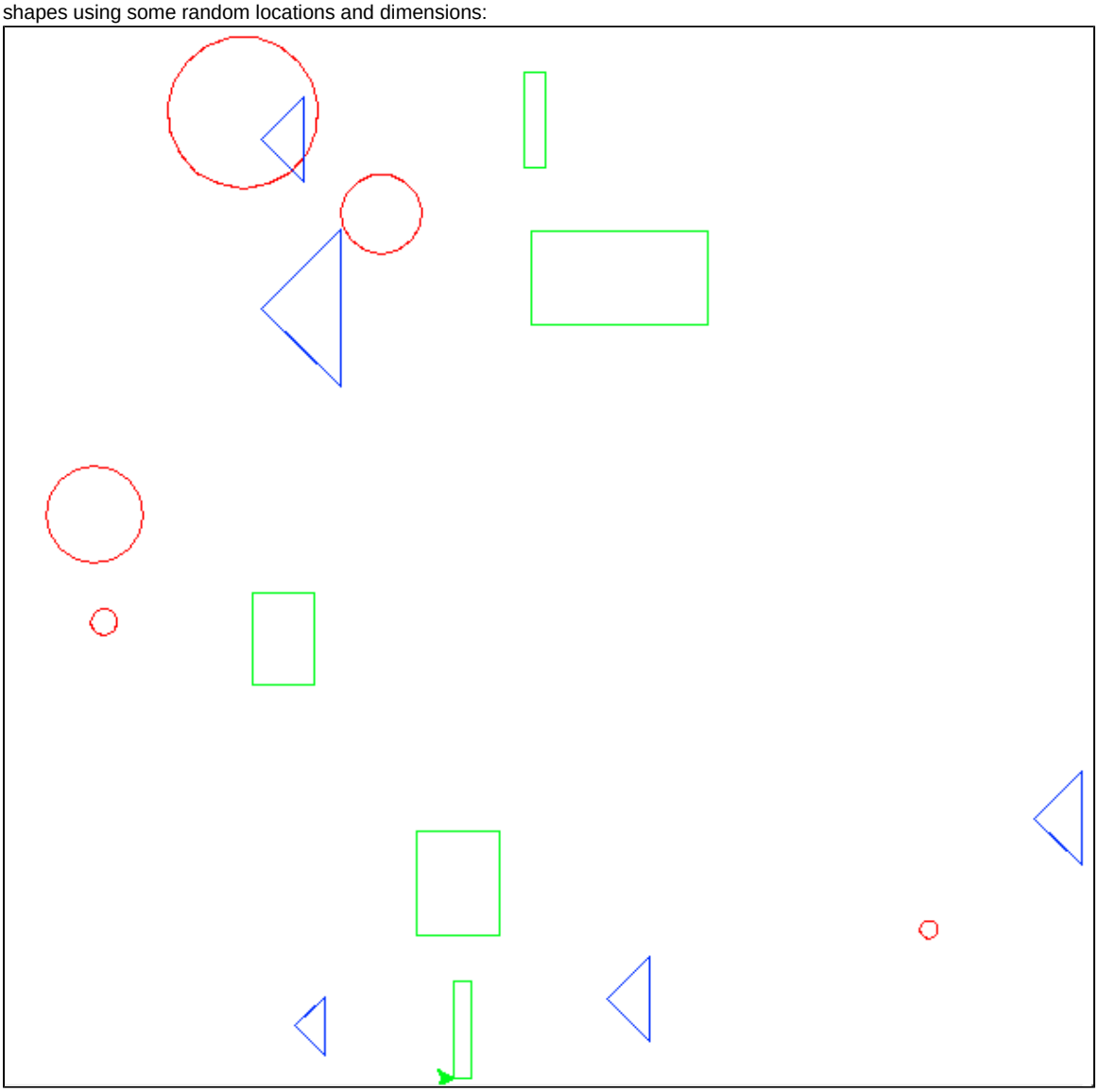
Harry's 2nd cs proj

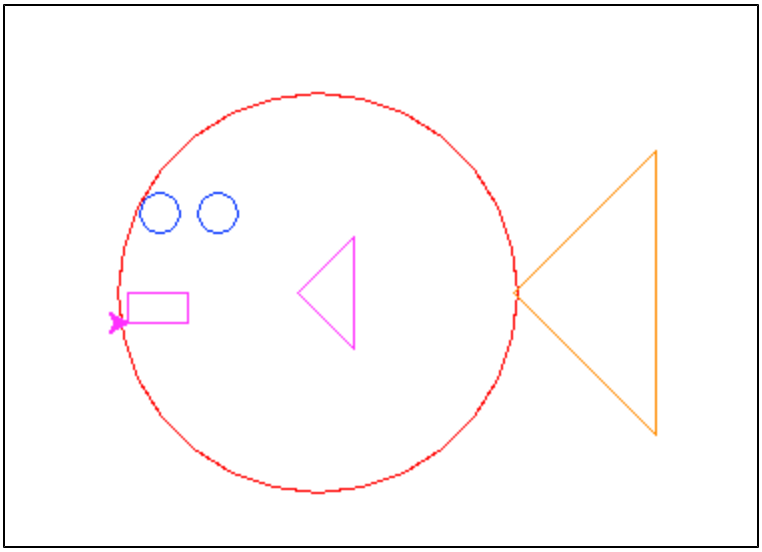
The purpose of this project was to practice defining and using functions as we did in last weeks lab. But this time we complicated matters by grouping a set of similar functions into one python file. Then we imported those functions into a new file, just as we do when we import the turtle package, thus avoiding the mess of looking at the functions' code when we used them.

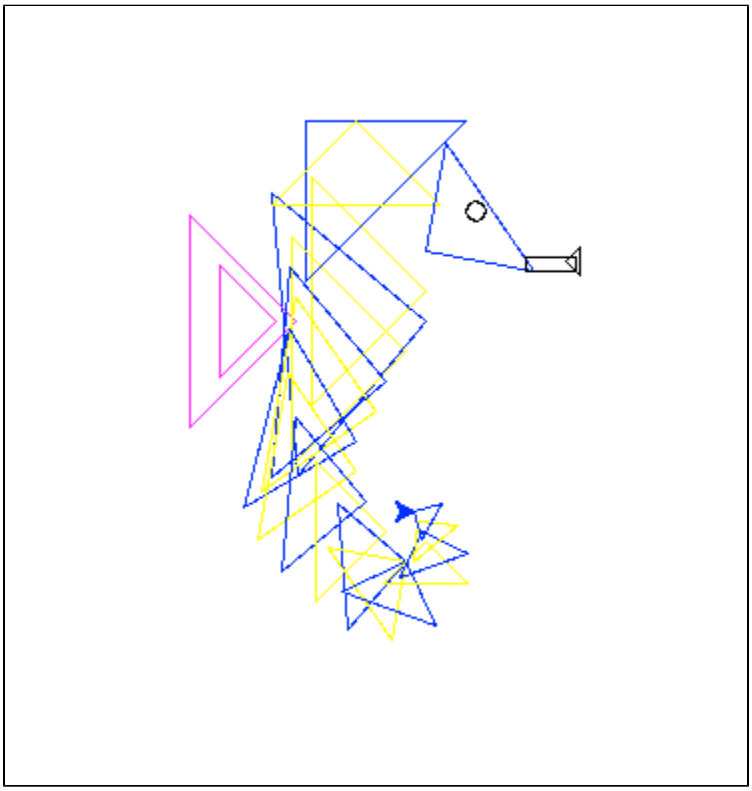
Our task was to make an undersea scene using turtle. To make the scene I would need some basic shapes. I decided to define functions for a rectangle, circle, and triangle. These functions would take location, dimensions, and color as parameters. Here's a picture of a test of my three shapes using some random locations and dimensions:



As you can see, all of the triangles are drawn in the same orientation. That's because I made sure to rotate the pen back to 0° after if finished drawing every shape. You can see it at 0° in the bottom middle. If I didn't take it back to the same angle every time, the triangles and squares would come out tilted by the angle the pen had before starting the shape.

Next I combined sets of shapes into two different drawing functions, one for a seahorse and one for a fish.



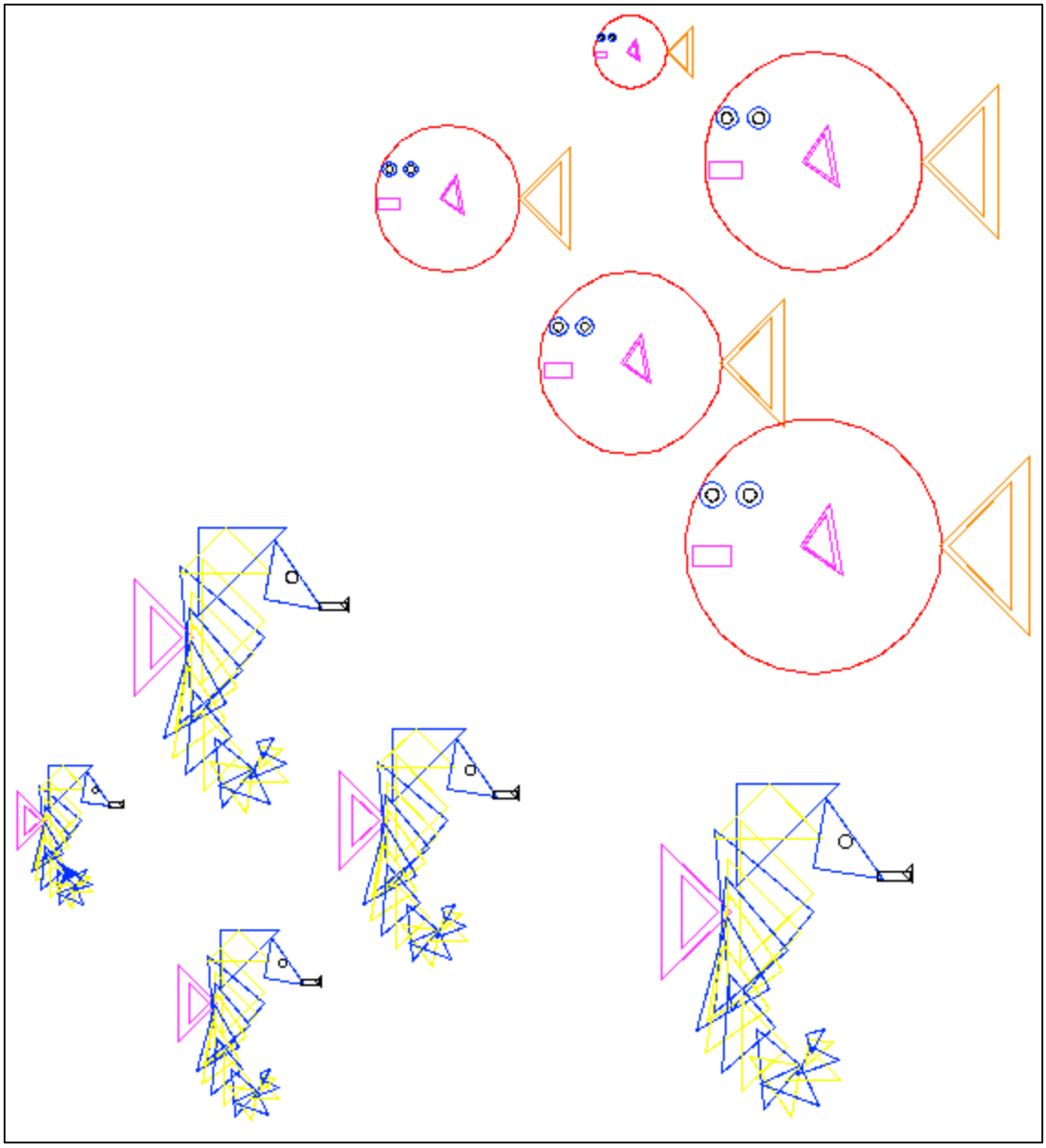


Use your imagination. In defining their functions I added parameters for position and size. No matter where I want to draw them or what size, the proportions of their constituent shapes stay the same. I achieved this by making sure not only that their constituent shapes would resize appropriately, but also that the distances between shapes would scale proportionally. Here's a snippet of code from the fish() function defining the fish's lifelike tail:

triangle(x+98*scale, y+100*scale, 0, 100*scale, "darkorange")

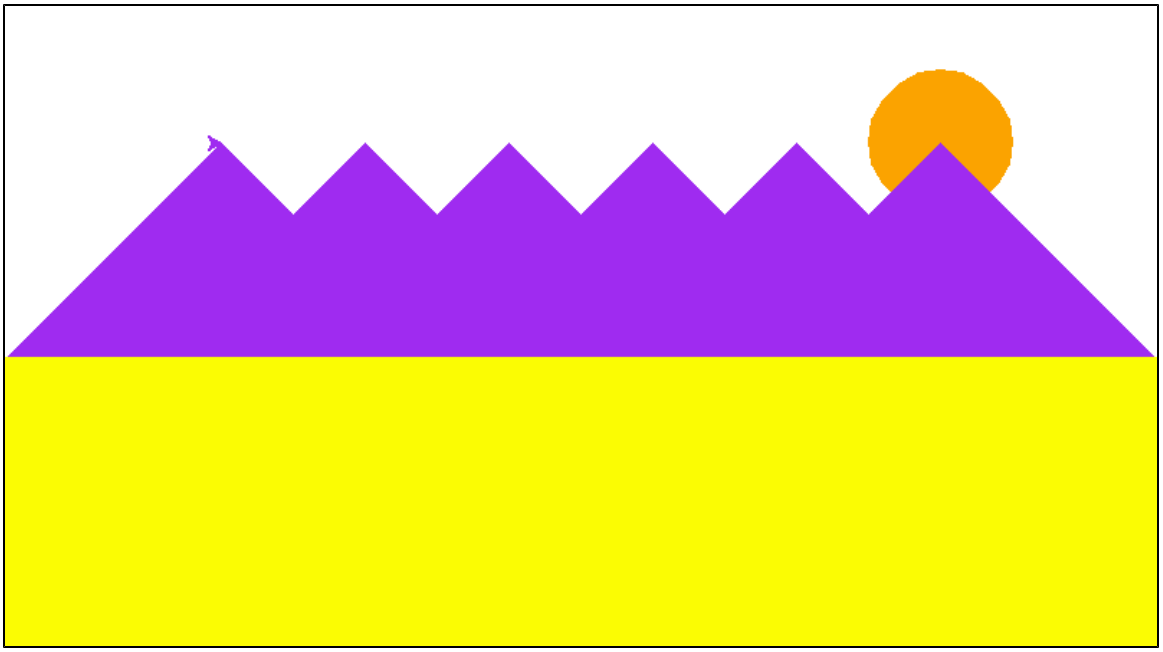
I defined the triangle function earlier in the file. It takes 5 parameters: x position, y position, rotation (here 0), size, and color. The size, x position and y position all depend on the scale factor, a parameter of the fish() function. To make the tail appear in the right place it had to be 98 pixels to the right and 100 pixels up from the bottom of the circle, the point x,y. But if the scale went up then the 98 and 100 would have to go up proportionally. That's the derivation of the math there.

Next I put some fish and seahorses together to make my underwater scene. Take a look:



I made this picture using the fish and seahorse functions, which in turn were made up of the triangle, rectangle and circle functions. The interesting part here was that I did not call on the fish and seahorse functions in the same file I defined them in. Instead I made a different file and used the import command to import the functions from my original file defining all these shapes. The import command makes all of the functions of that file available to me, but I don't have to see them. Thus I could program my underwater scene without clutter. The file where I scripted out this scene contained only 12 lines of code.

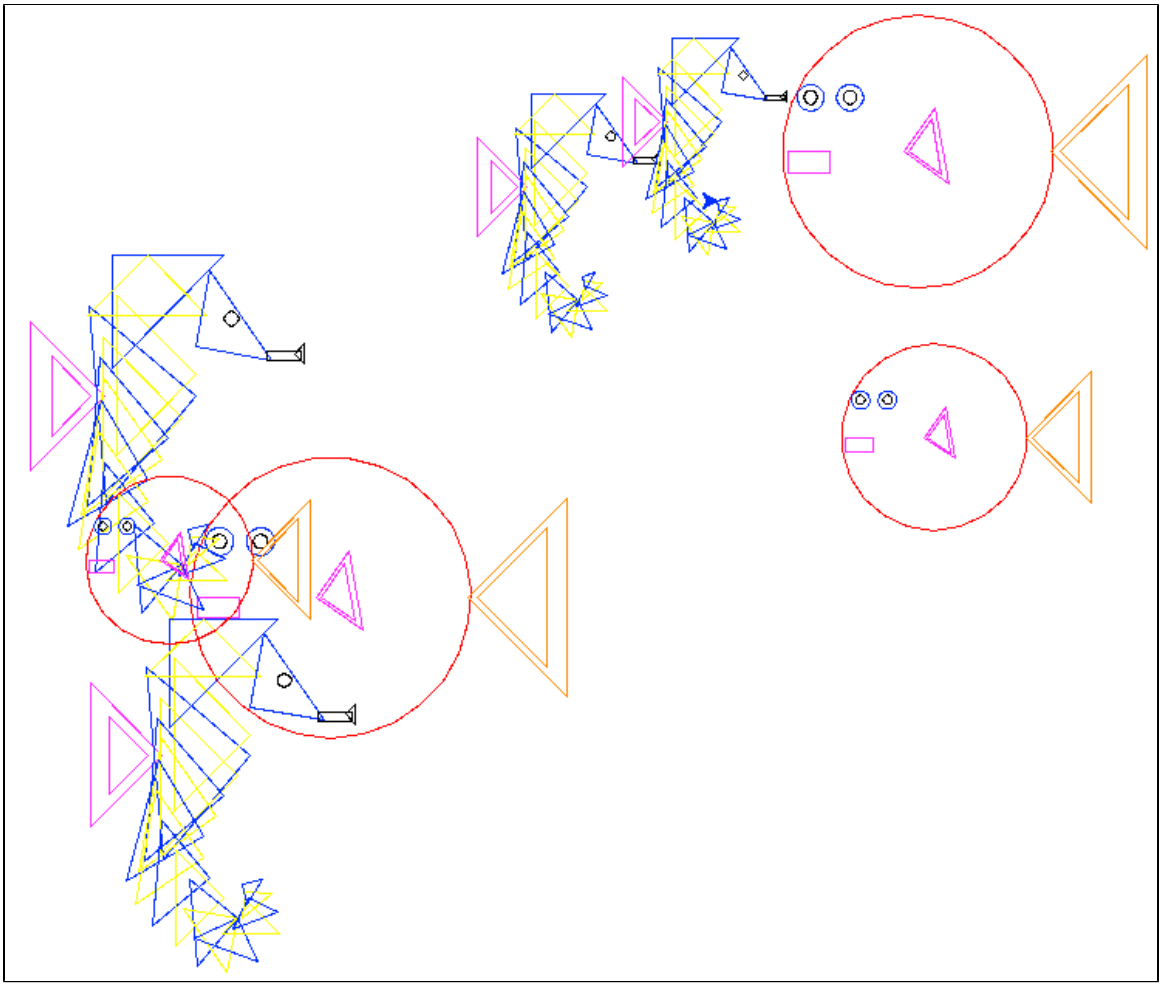
Next I drew a totally different picture , this one a scene from last summer: the mojave desert of california. I used the same process of defining shapes in my shapes.py file and using the shape functions in my main.py file. Here's the pic



I also extended my code with a function to draw a random underwater scene. It didn't turn out very well because the fish and seahorses draw over one another in a jumbled mess. I would have used the for loop just like in the instructions, but I found that the random.random() function would spit out really small numbers which would shrink my fish and seahorses to mere pixels. So instead of using a for loop I used a while loop which allowed me more control over the state of the indexing number. Here's a snippet:

```
index = 1
while index < 5:     # while the index is less than 5, the code below executes
    x = random.random()     # x is a random real number between 0 and 1
    if x > 0.5:
        index += 1     # adds 1 to the index
        # this part then draws a fish of scale factor x in a random location
```

The index only increases if x is greater than 0.5, and fish only stop being drawn once the index reaches 5. Here's the result:



In this project I learned that it is organizationally attractive to put a bunch of similar functions in one .py file as a repository for other .py files to draw from. I also learned how to use a while loop to ensure certain conditions are fulfilled before the loop terminates.