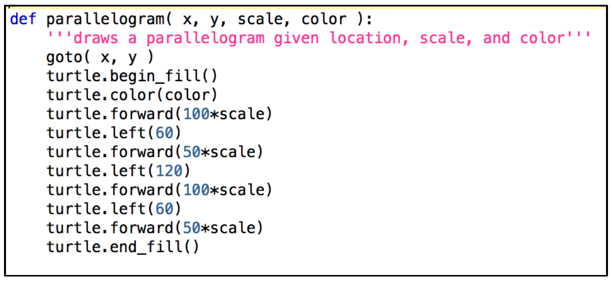
Project2

In this project, I coded and used more functions and used parameters to make my code more efficient. Using parameters, or variables defined in functions, allows my code to become more complex without getting out of hand. This project focused on taking function coding in Python one step further by making code more organized, more clear, and more efficient with parameters and annotations(coding notes).

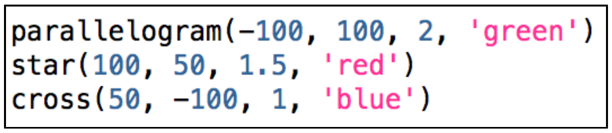
First off, in **task 1** copied my code and coding notes from the Python file I created during Lab 2 on January, 12th that introduced function parameters to the CS151 class. I renamed the copied Python file shapelib.py and used that file throughout my project as my "Shape Library" or library of functions. Next, I was asked to create a file titled main.py that imports the functions from my shapelib.py by simply writing "import shapelib" at the top of my code. Then, I created two functions in my main.py file titled outdoors1 and outdoors2 to be used later in my project.

Task 2 asked me to create two simple shape functions with at least three parameters. The three parameters that **task 2** asked the functions to have were x location, y location, and scale. The goal of this task was to create two simple functions that would successfully run and draw no matter what parameters the function was given. I choose to create three functions, one that drew a parallelogram, one that drew a star, and one that drew a cross. All three of these functions took four parameters: (x, y, scale, color). Here is what the code for my parallelogram() function looked like:



```
def parallelogram( x, y, scale, color ):
    '''draws a parallelogram given location, scale, and color'''
    goto( x, y )
    turtle.begin_fill()
    turtle.color(color)
    turtle.forward(100*scale)
    turtle.left(60)
    turtle.forward(50*scale)
    turtle.left(120)
    turtle.forward(100*scale)
    turtle.left(60)
    turtle.forward(50*scale)
    turtle.end_fill()
```

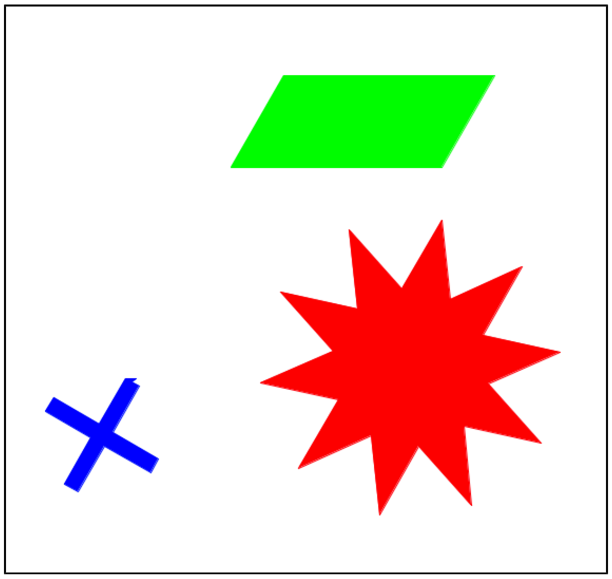
The functions for the other two simply shapes in **task 2**, star() and cross(), had similar code. Notice the first line of code in the function reads "goto(x, y)." In lab period 2, our class created a function called goto() that told python to pick up and go to the inserted coordinates and put down. Here I am calling that function to use in the beginning of my parallelogram function. Then, I was asked to run my simple functions with different parameter inputs and save the resulting image. Here is the top-level drawing code I wrote to call all three of the functions:



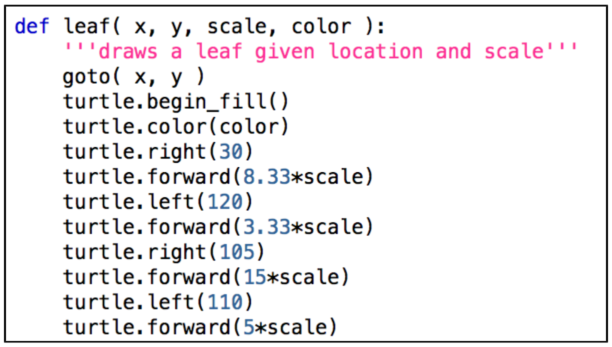
```
parallelogram(-100, 100, 2, 'green')
star(100, 50, 1.5, 'red')
cross(50, -100, 1, 'blue')
```

Notice how in each call to function I included the four parameters of x location, x location, scale, and color. This will instruct Python to use those parameters and draw the shape at those coordinates, at that scale, in that color. Here what the resulting Python image looked like:

Required Image1:(simple shapes)

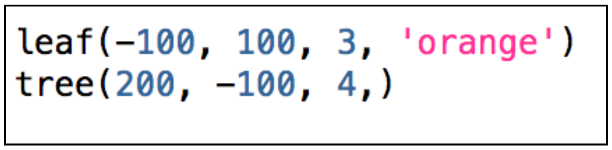


Task 3 asked to make more functions that draw objects that one might find in the Maine Outdoors. **Task 3 *was very similar to task 2.** *The goal was to define some functions with a few parameters that drew objects from the Maine Outdoors that would draw properly at any location, scale or orientation. I decided to code two functions, one that created a leaf and one that created a tree. Just like in my task 2 section, these functions took three parameters: (x, y, scale). The leaf() function took a fourth parameter for color, but the tree() function did not. Here is a short snippet of the code for the function leaf():



```python
def leaf( x, y, scale, color ):
    '''draws a leaf given location and scale'''
    goto( x, y )
    turtle.begin_fill()
    turtle.color(color)
    turtle.right(30)
    turtle.forward(8.33*scale)
    turtle.left(120)
    turtle.forward(3.33*scale)
    turtle.right(105)
    turtle.forward(15*scale)
    turtle.left(110)
    turtle.forward(5*scale)
```

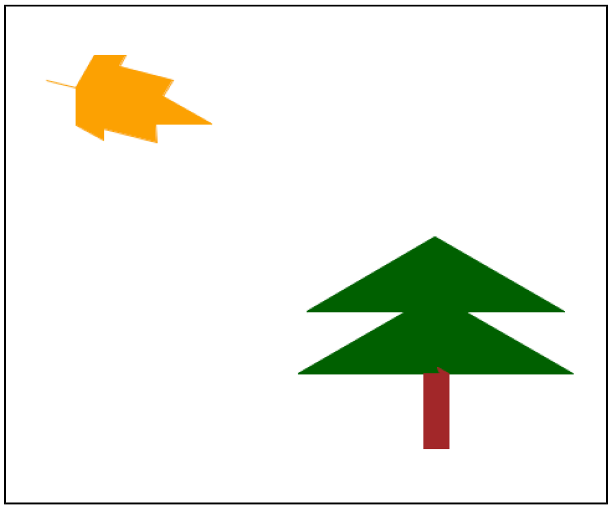
Notice again my first line of code inside the leaf() function calls the goto() function defined in CS151 Lab to move Python turtle to a given coordinates. The tree() function looked similar to the leaf function but obviously with specific aspects of its own. The tree() included Python turtle code for color inside the function. I used new turtle command begin_fill(), color(), and end_fill() to code this. Then, I was asked to run my simply Maine objects functions with different parameter inputs and save the resulting image. Here is the top-level drawing code I wrote to call the two functions:



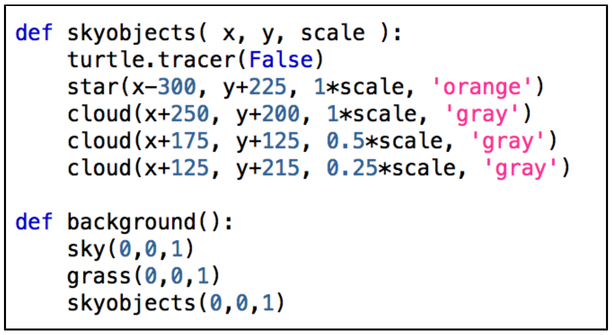
```python
leaf(-100, 100, 3, 'orange')
tree(200, -100, 4,)
```

When I ran this code, Python drew this image:

Require Image2:(simple Maine objects)

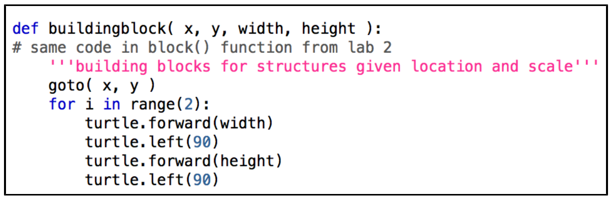


Task 4 finally made use of our second file, main.py. The task asked to use the functions I defined in task 2 and task 3 to make define the function outdoors1() in main.py. But, in my case the Maine outdoor location I chose needed more functions than just the simply image functions from the previous tasks. For my first Maine outdoor location I decided to create Mayflower Hill at Colby College in Waterville, ME. So, next I worked on coding a bunch of new functions that pertained to Mayflower Hill. First, using the same techniques from the previous tasks I created functions for clouds (cloud()), a sky (sky()), grass (grass()). Then, I made a function called skyobjects() that called the star from task 2 to be the sun and called the cloud function. Also, I made another function called background() that contained the functions for sky() and grass(). These code for the two functions background() and skyobjects() looked like this:



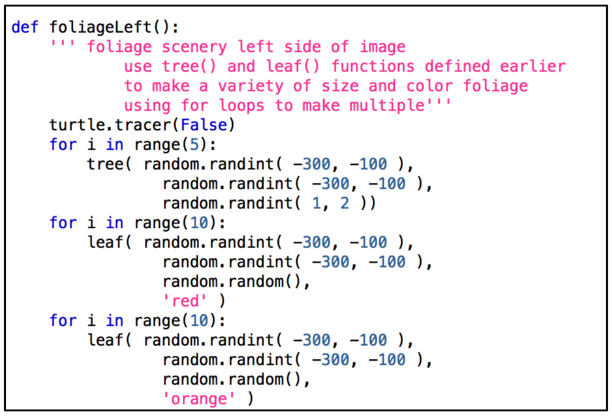
```
def skyobjects( x, y, scale ):
    turtle.tracer(False)
    star(x-300, y+225, 1*scale, 'orange')
    cloud(x+250, y+200, 1*scale, 'gray')
    cloud(x+175, y+125, 0.5*scale, 'gray')
    cloud(x+125, y+215, 0.25*scale, 'gray')

def background():
    sky(0,0,1)
    grass(0,0,1)
    skyobjects(0,0,1)
```

Next for **task 4** I had to create functions to draw Miller Library. To do this, I used created a function called buildingblocks() that draws a four sided polygon using a for loop and taking the parameters x location, y location, width, and height. This function buildingblocks() was able to create all of Miller Library including the pillars. I then created a function called MillerLibrary() that used a bunch of buildingblocks() functions along with turtle.color() functions to draw and color the library. MillerLibrary() took parameters (x, y, scale). Also, I created a small function called C() that just drew a C on the top of the library. The buildingblocks() function is a very important function because I used it throughout the entirety of my project. Here is what the function code looked like:



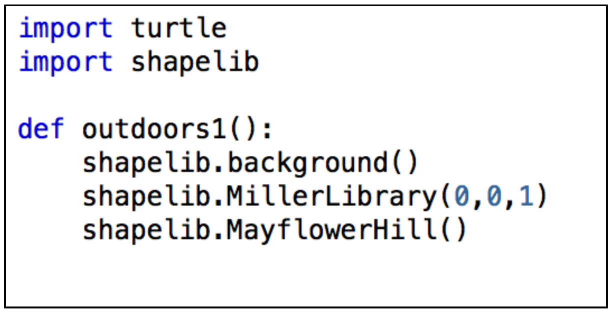
```
def buildingblock( x, y, width, height ):
# same code in block() function from lab 2
    '''building blocks for structures given location and scale'''
    goto( x, y )
    for i in range(2):
        turtle.forward(width)
        turtle.left(90)
        turtle.forward(height)
        turtle.left(90)
```

The last piece of my **task 4** work was the create the Mayflower Hill scenery. To do this, I used the two functions I defined earlier leaf() and tree(). I created two new functions called foliageLeft() and foliageRight(). These two functions are by far the two most complex functions I have written. Using knowledge from Lab2, I used for loops, random.random(), and random.randint() to spread foliage across my image. For foliageLeft() I did a for loop for tree(x, y, scale) and assigned its parameters using the turtle random package. I imported this random package at the top of my code by typing "import random." I also use a for loop for leaf(x, y, scale, color) and assigned its parameters(besides color) with turtles random package. I used random.randint() to input a range of values I want Python to randomly choose from to send my function to start. Also, I used random.random() to pick a number between 0 and 1 to assign to the scale of the two functions leaf() and tree(). Here is what the code for foliageLeft() looked like:



```
def foliageLeft():
    ''' foliage scenery left side of image
            use tree() and leaf() functions defined earlier
            to make a variety of size and color foliage
            using for loops to make multiple'''
    turtle.tracer(False)
    for i in range(5):
        tree( random.randint( -300, -100 ),
                random.randint( -300, -100 ),
                random.randint( 1, 2 ))
    for i in range(10):
        leaf( random.randint( -300, -100 ),
                random.randint( -300, -100 ),
                random.random(),
                'red' )
    for i in range(10):
        leaf( random.randint( -300, -100 ),
                random.randint( -300, -100 ),
                random.random(),
                'orange' )
```

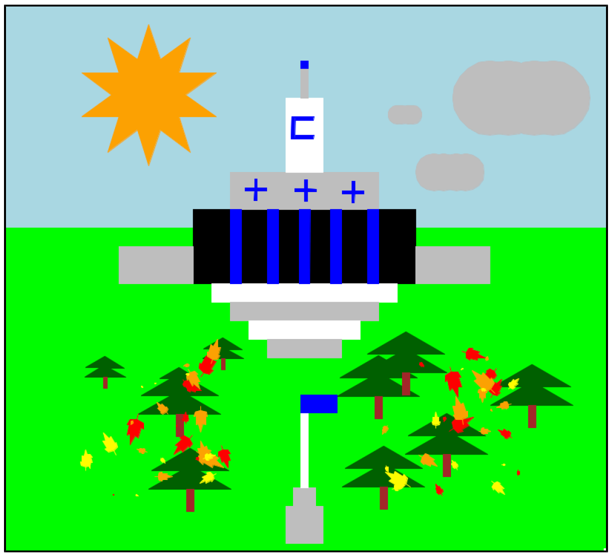
Notice that I called the leaf() function more than once. Actually, I called it three times and this was to use three separate colors, yellow, red, and orange. The code for foliageRight() is identical but with different coordinate/location parameters. Next, I defined one final function for **task 4** called flagpole() which used my function buildingblocks() and turtle color commands to create a flagpole. I consolidated all these functions in a function called MayflowerHill(). So, in the end, for **task 4** I created a bunch of new functions but consolidated them into three functions titled background(), MillerLibrary() and MayflowerHill(). Lastly, in my main.py file I defined the function outdoors1() with these three main functions from shapelib.py for **task 4.** Here is what my code for the function outdoor1() looked like:



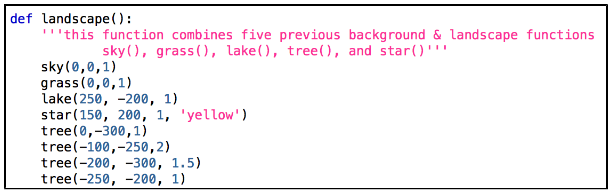
```python
import turtle
import shapelib

def outdoors1():
    shapelib.background()
    shapelib.MillerLibrary(0,0,1)
    shapelib.MayflowerHill()
```

Then, I ran main.py and the outdoors1() function and this is the image I produced:

Required Image3:(Outdoor1 Mayflower Hill)

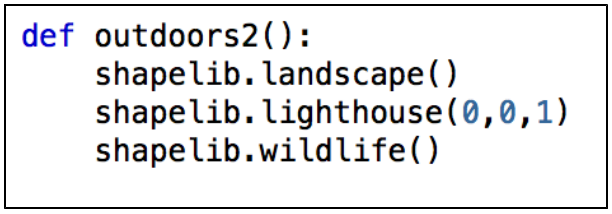


Task 5 was basically the same as task 4, but I was asked to make an entirely new Maine Outdoors scene. The second scene I decided to create was a classic Maine lighthouse near a lake with a moose and lobster in the picture. Again, to do this I had to create and define a bunch more functions. First, I created a function called lake() that took three parameters, x location, y location, and scale. In this function, I used my goto() function and the turtle.circle() function. I coded a number of side by side circles colored blue to make the lake. Next, I created a new function called landscape() that combined the lake function with some of my previous functions including star(), cloud(), sky(), grass() and tree(). The first few lines of code for landscape() were as follows:



```python
def landscape():
    '''this function combines five previous background & landscape functions
            sky(), grass(), lake(), tree(), and star()'''
    sky(0,0,1)
    grass(0,0,1)
    lake(250, -200, 1)
    star(150, 200, 1, 'yellow')
    tree(0,-300,1)
    tree(-100,-250,2)
    tree(-200, -300, 1.5)
    tree(-250, -200, 1)
```

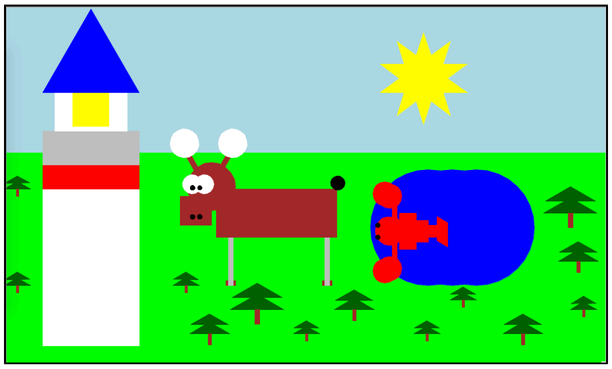
Next, I had to create functions for the animals. I created moose() and lobster() and used both turtle.circle commands as well as the buildingblock() function I created earlier. Also, for the lobster I used my parallelogram() function to make the tail. Again, I used turtle.color(), turtle.begin_fill(), and turtle.end_fill() to color this function and other functions throughout my project. Both of these new functions took the parameters (x, y, scale). I'm not showing an image of this code because it is very similar to my previous code. Like before, I then consolidated these two new functions into a new function called wildlife() that basically called lobster() and moose() to function. The last part I had to create for this second Maine outdoor image was the lighthouse. To do this, I first created a function called roof() that took the parameters (x, y, scale) and used basic turtle commands to draw a blue triangle on top of my lighthouse. Then, I created a function called lighthouse that again used my function buildingblocks(), color commands, and called the new roof() to draw the lighthouse. Finally, just like in task 4 I took the three consolidated functions required to draw my second image, landscape(), wildlife(), and lighthouse(), and defined the second outdoor function in my main.py file. The code for the function outdoors2() was as follows:



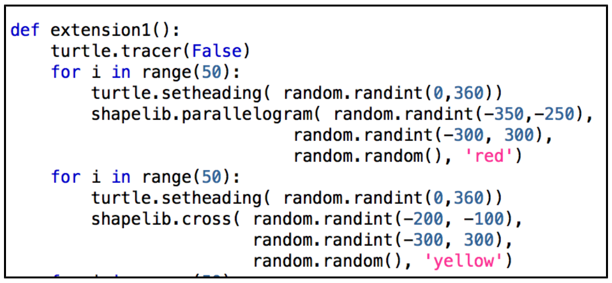
```python
def outdoors2():
    shapelib.landscape()
    shapelib.lighthouse(0,0,1)
    shapelib.wildlife()
```

Then, I ran main.py and the outdoors1() function and this is the image I produced:

Required Image4:(Outdoor2 Lighthouse Lake)

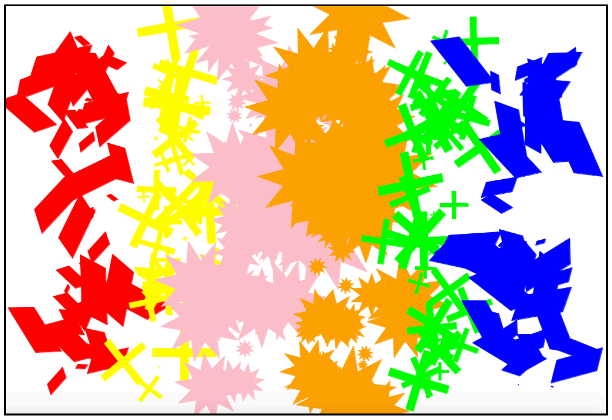


After I completed the required tasks for the project, I was encouraged to do further extensions. I decided to do two extensions. My first extension, **extension 1**, puts to use a number of the new things I learned through this lab. With this first extension, I wasn't going for beauty or creativity, rather I wanted to simply depict a number of the new Python commands, functions, and code I learned. For this extension, I used the three simple shape functions I defined in task 2. These three shape functions(parallelogram(), star(), and cross()) all take four parameters of x location, y location, scale, and color. In a new file titled RandomRepeatExtension.py I created a new function extension1() that called my simple shapes from shapelib.py. Again, I used "import shapelib" to do this. I used for loops to repeat the drawing of these functions multiple times. I also used another new skill I learned, "import random", to define three of the parameters in the three simple shape functions. I wanted to randomly generate starting locations between certain integers using random.randint() for the x and y parameters and I wanted to randomly generate a number between 0 and 1 for the scale parameter. In addition, I put in the color parameter myself. Here is an example of some code I did for **extension 1**:



```python
def extension1():
    turtle.tracer(False)
    for i in range(50):
        turtle.setheading( random.randint(0,360))
        shapelib.parallelogram( random.randint(-350,-250),
                                random.randint(-300, 300),
                                random.random(), 'red')
    for i in range(50):
        turtle.setheading( random.randint(0,360))
        shapelib.cross( random.randint(-200, -100),
                        random.randint(-300, 300),
                        random.random(), 'yellow')
```

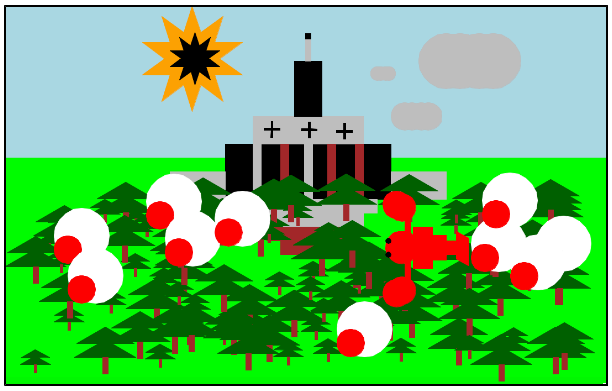
Through this extension, I really wanted to demostrate my skill with using randomly located and repeating images. When I ran RandomRepeatExtension.py and the function extension1() this is the image I produced:



My second, **extension 2**, was a spin off of my first Maine Outdoor function in task 4. I decided to mess with some of the colors, add so for loops, and some random location parameters to create the Colby Apocalypse. In my new file titled ColbyApocalypseExtension2.py I again imported random, turtle, and shapelib to gain access to all three of these collection of commands and functions. I then copied most of my code from shape lib for MillerLibrary(), but I then went through and changed a bunch of its colors to make the building looks old and run down. Then, I added a black star with my star() function to make the sun look like it is turning black. Then, I created a for loop with the tree() function from my shapelib.py and created random location parameters with random.randint(). This was to give the campus the look of being overgrown. The last piece of this extension is where things get odd. I then wrote a function called ghost() that creates circular alien-cyclopes-ghosts with one big red eye. Again, with another for loop and random commands, I spread these alien-cyclopes-ghost invaders all over the library lawn. Finally, I decided to make the King of the Alien Invaders a lobster. This lobster has taken over Colby and the rest of Maine because he is fed up with human eating his family members. He recruited an army of alien-cyclopes-ghosts to help him. Anyways... In my code, I used a lot of for loops and random package commands; also, like the entire project, I made good use of parameters again. Here is some of the code from **extension 2**:

```
    for i in range(100):
        shapelib.tree( random.randint( -500, 500 ),
                random.randint( -300, 0 ),
                random.randint( 1, 2 ))
    shapelib.star( -250, 240, 0.5, 'black')
def ghosts():
    for i in range(10):
        ghost(random.randint( -500, 500 ),
                random.randint( -300, 0 ))
```

Then, I ran ColbyApocalypseExtension2.py and the three ColbyApocalypse(), ghosts(), and lobster() functions and this is the image I produced:



In conclusion, through this project, I sharpened my function skills and gained a better understanding of parameters. I feel like I became a more efficient coder because I now know how to make code simpler and less lengthy. Also, I learned a lot of new Python skills including the random package, for loops, and adding color. All of these things came in handy throughout my project and its extensions. I can make much more complex images with much less code. Being able to add location and scale parameters to function definitions was a vital skill I learned. Overall, this project made me a more organized and efficient coder.

I received help from TA Mike with the command turtle.setheading() and worked alongside fellow CS151 coder Stefan Kohli during write up.