Yahir Rivas

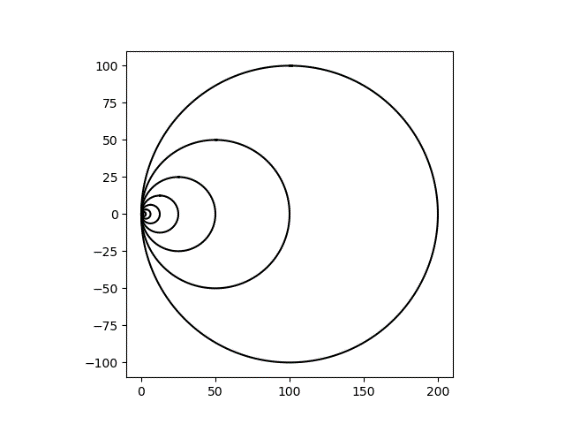
Instructor: Dr. Olac Fuentes

CS 2302

**Lab 1 Report**

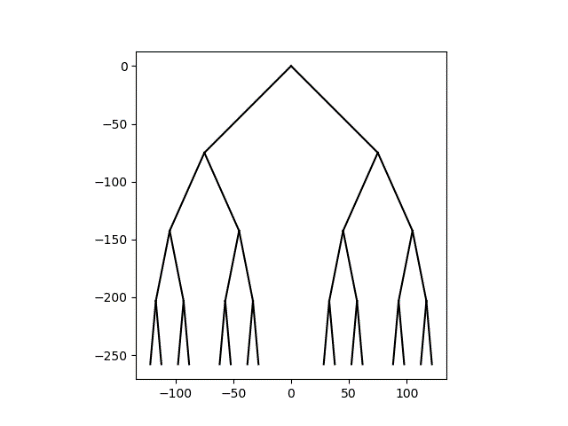
For this lab, I was asked to submit a program consisting of plotting different figures using the mat plot library and recursion. I was given two codes, one for plotting a square and another one for plotting a circle. The goal was to identify what these codes did and implement them to solve the exercises that were asked for.

With the codes given and changing some variables, I realized what some of the code did, which helped me for the following assignments. I started with the second exercise since I knew that one needed very few modifications of the code already provided. That exercise asked to modify the circle to move left and make it smaller every recursive call. The code provided already made the circle smaller by multiplying the radius by a fraction of the original one using another variable. Using that same variable to change the x coordinate I was able to modify the center to move to the left which made the circle look like this:

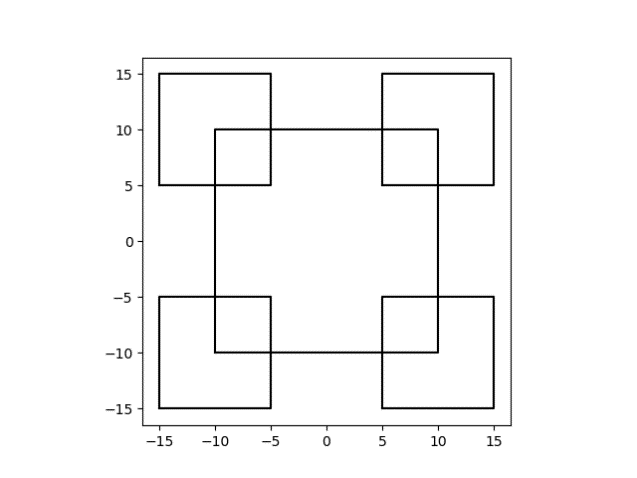


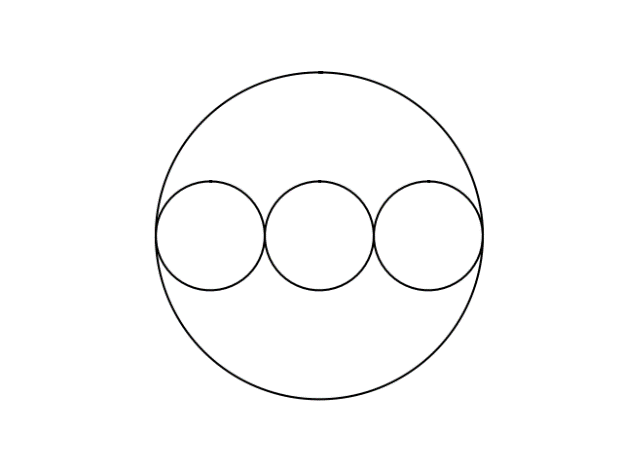
The only thing remaining to do was modify the number of circles and fraction they would be changed by for the remaining circles.

The next code I did was the third exercise which binary tree since I had some understanding of how to plot the lines. So just like the circle I plotted the line for direction x and y and reduced their length by the same amount in each recursive call so that they would never intersect.

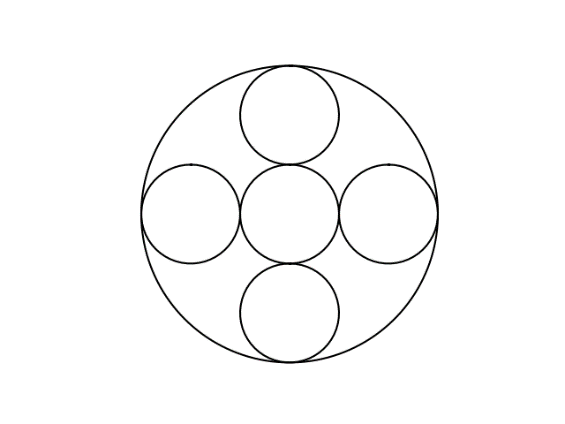


Next, I did the first exercise, which consisted of plotting squares like the one already provided, in the corner of last recursive call squares. For doing this I knew that I needed to change the center of the square to be that of the corner which I wanted to plot the square at. So, using the distance from the center of the square to any of its vertices, I created a variable to modify the center of the square and the rate at which it changed in size. Using the axis of x and y axis I knew that adding or subtracting that variable I created would change where the square’s center would be located at and I made a recursive call for each square that would be drawn.



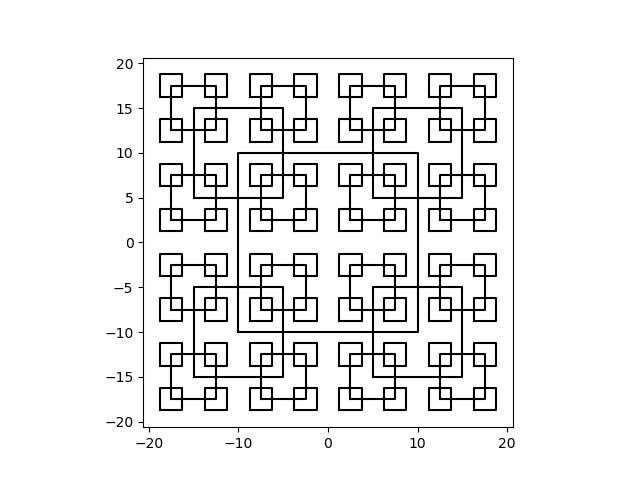
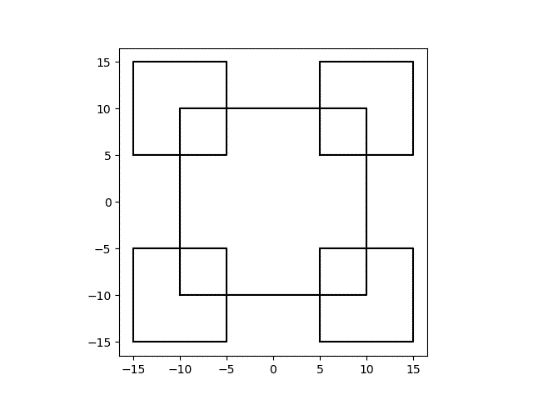
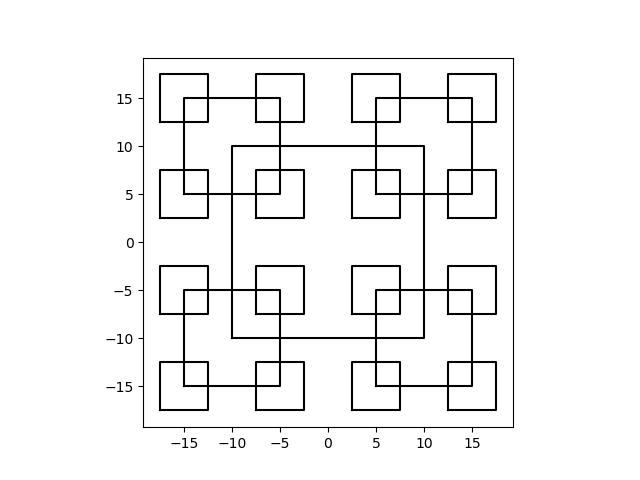
The last problem I solved was the 4th one. This figure consisted of drawing 3 circles inside another one along the diameter. I used a similar concept to the square problem. I got the diameter to the circle, divided it by 3 and then created 3 circles inside it. Making a figure like this:

I applied the same strategy for the y axis and created two other circles since the middle one was the same for x and y axis. Then, I moved each circle to the desired direction by using the second radius value. Ending with the following figure:



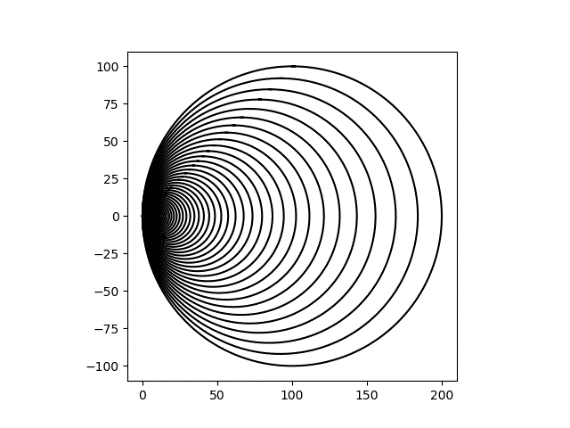
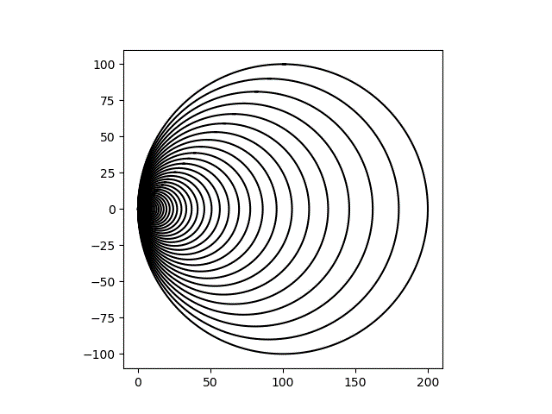
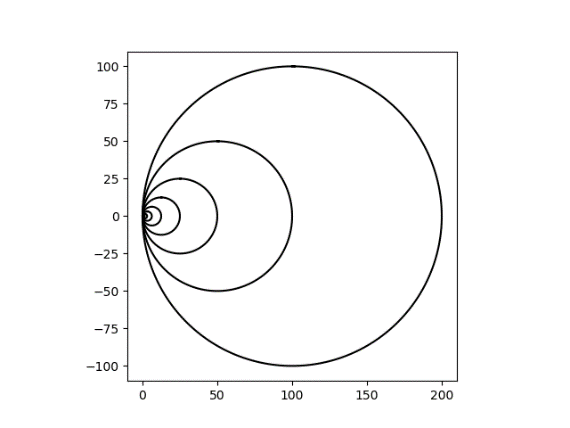
**Results**

For these experiments I tested the output and running times of the figures under different circumstances. I changed inputs such as number of figures, size of radius, and distance between each figure. For this I used the time library in python to calculate the running time for each of the different inputs and compare the results to analyze what they mean.

**Figure 1**

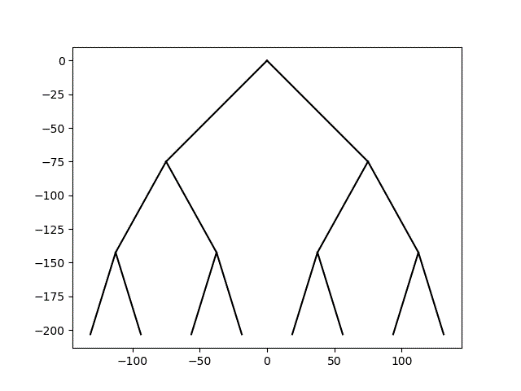
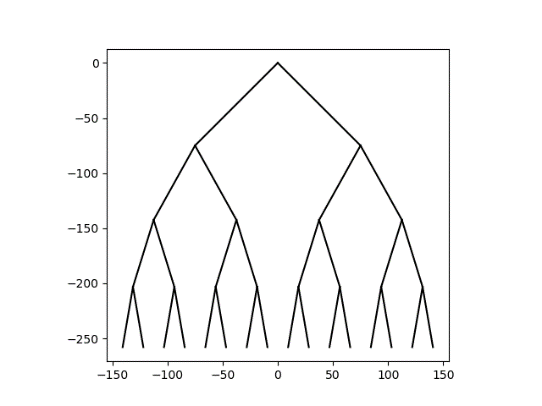
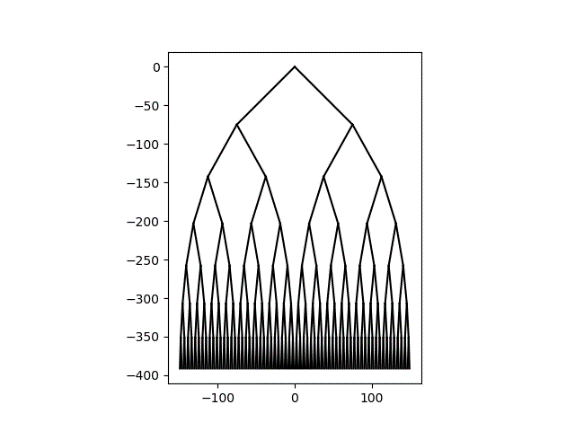
**Running Time: 0.02s Running Time: 0.04s Running Time: 0.19s**

There are 3 different outputs for figure 1. The first one uses two recursive calls, one for the center square and another one for the corner squares, displaying a running time of 0.02s. For the next one I changed the number of recursive calls to three increasing the running time exponentially to 0.04s. The last input was made with four recursive calls which made the figure a lot more saturated than with previous calls and drastically increasing the running time to 0.19s.

******Figure 2**

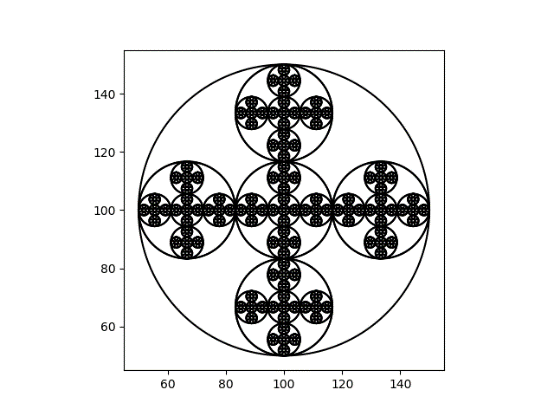
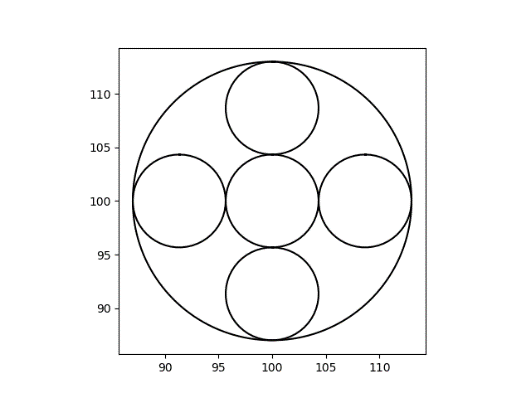
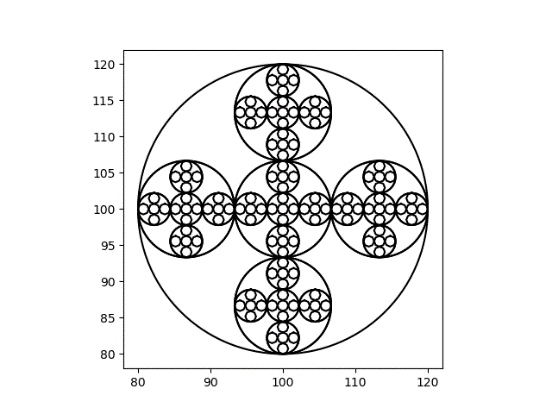
**Running Time: .08s Running Time: 0.11s Running Time: 0.20s**

Unlike the previous figure, I changed two different outputs this time. The inputs I changed were the number of circles drawn and the rate at which the radius changes. The rightmost picture has the greatest number of circles and the lowest amount of change in radius. The running time was much closer in this figure than in the previous one although there was still a significant increase from run two to run three.

**Figure 3**

**Running Time: 0.03s Running Time: 0.08s Running Time: 0.60s**

For figure 3 there was only one variable changed, the number of levels of the binary tree. But as we can see there was a very big difference between test 2 and test 3. As we have noticed, the more there is to plot, the longer it takes for the program to run. This program Big-O notation is also linear which means that it should take more, the bigger the number of calls.

**Figure 4**

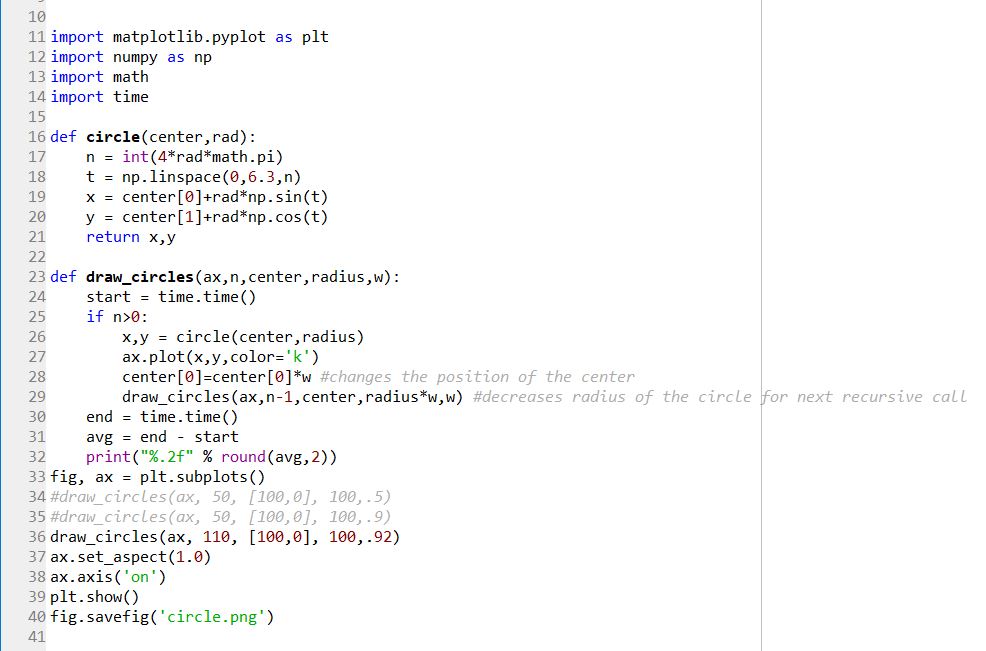
**Running Time: 0.02s Running Time: 0.39 Running Time: 2.90s**

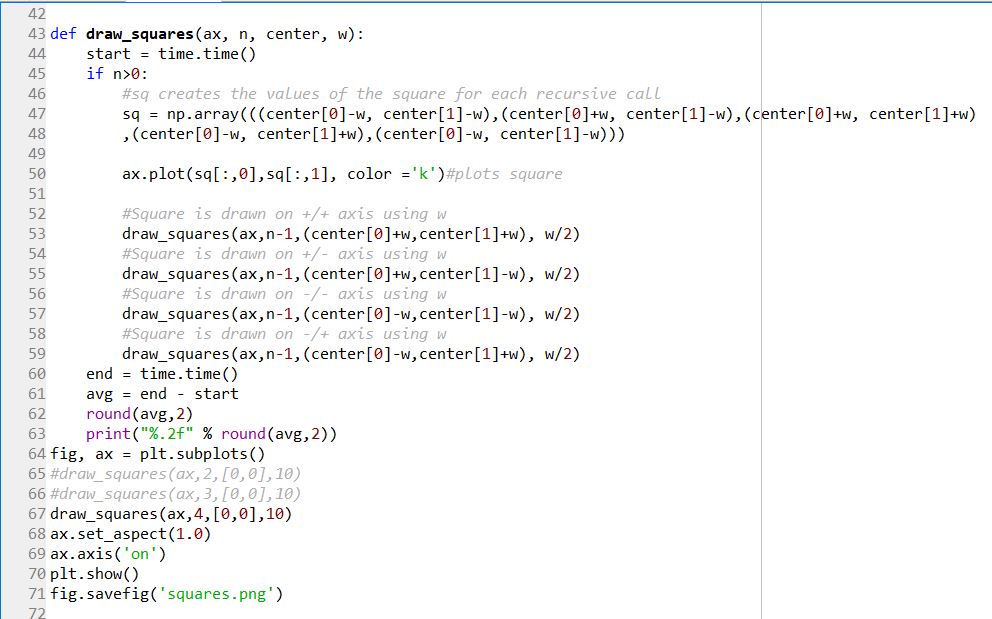
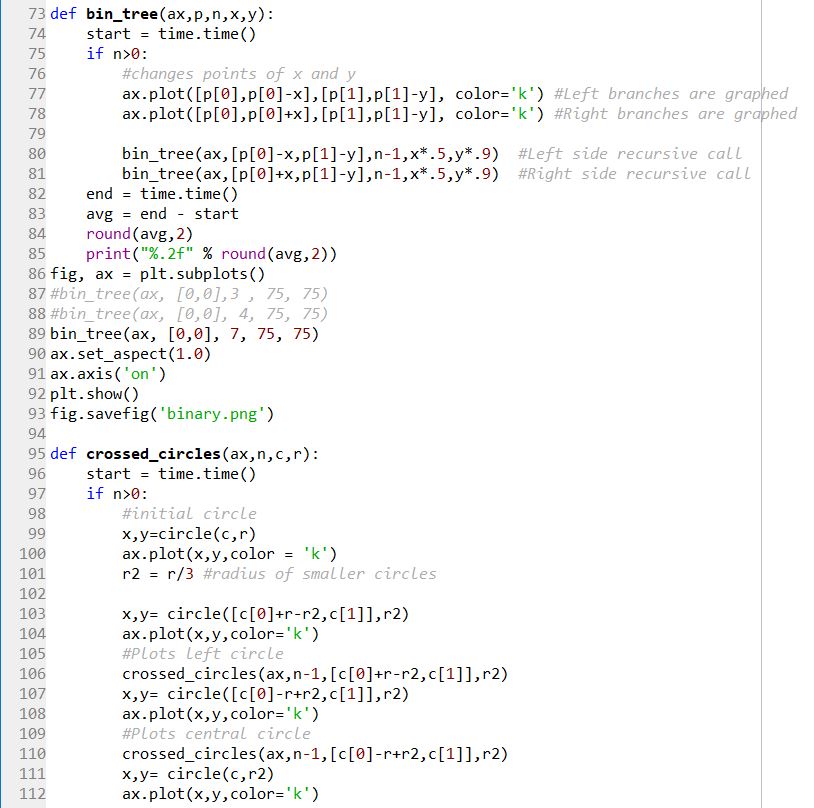
In this last figure there were two variables that changed, the number of circles and their radius. While the running time of the first test was one of the fastest, it was evident that this figure required more running time than the previous ones. The third test had the slowest test running time by 483%. Considering this method has five different recursive calls, it was expected to be slower than the other figures.

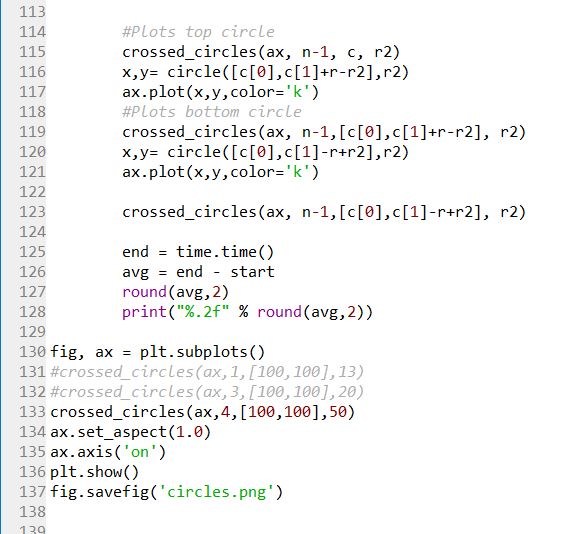
**Conclusion**

What I learned from this lab was recognizing code and modifying it to create whatever you need. For the project we needed to identify what the code provided by the professor did, and then modify that same code to create different figures. The first steps to this was changing some variables and recognizing what they do, then slowly understand what everything is supposed to do until you get a tougher objective and start creating your own code. This project also helped me understand a lot better recursion in Python. While doing the report I realized that this had a lot to do with running times as well and their importance in coding. While some programs may run slow depending on the number of variables and recursive calls they have, other are also affected by the language and interface in which they’re working.

This lab helped me understand that being efficient with your code is extremely important, specially when dealing with higher level programs that need more variables. I believe this lab was a good way to practice coding comprehension while also introducing recursion and time complexity analysis. At the beginning it seemed very though to understand exactly what everything did, but as you try changing variables, experimenting and practicing, you find out how to modify it and then create your own program.

**Appendix**

****

****