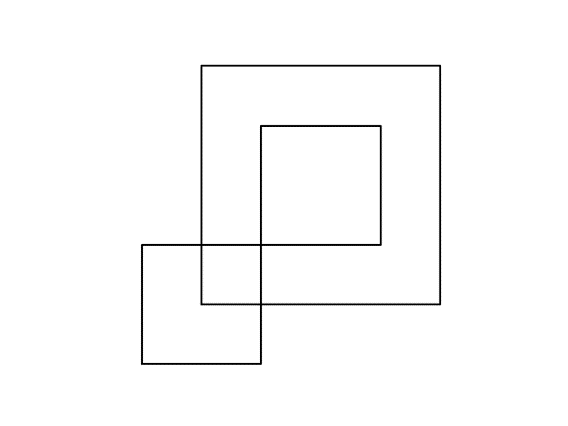
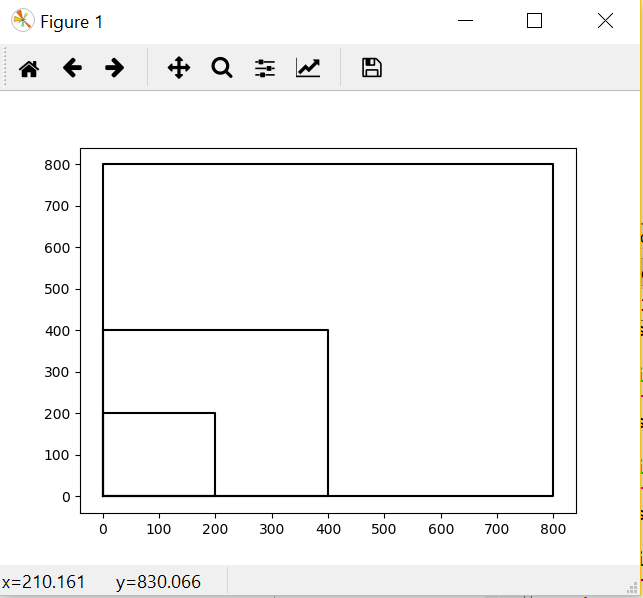
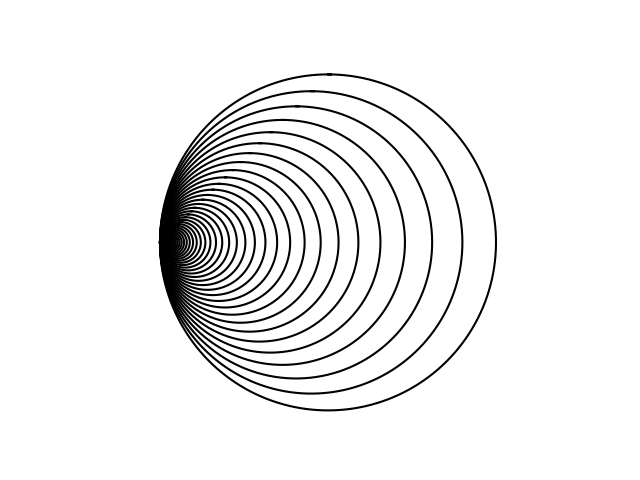
Lab 1 Report

Introduction

For this lab we were asked to use recursion. To move shapes around in different ways with recursive calls.

Proposed solution Design

I was only able to draw one figure. Which were the shifting circles to the left. What I did for it was I made a new center. Which was the original center at index 0 multiplied by the weight which is w. For the Circle diagram I know I needed 5 recursive calls which would make one circle each inside a big circle. I thought just multiplying the radius by 2/3 and then adding it and subtracting it in my calls would change something but nothing moved. The squares was also rough because you need four calls that will change the size and position of the circle. I thought by having the circle and square on its own would make the task simpler. I was wrong the math behind it is a little bit tedious. The only figure which I could not figure out how to work was the Tree because of. How you need to get three points and then just have two recursive calls for the left and the right.



Conclusion

I learned how an element in the array can be modified all together or one by one. It is a neat feature in python

Appendix

import numpy as np

import matplotlib.pyplot as plt

def draw\_squares(ax,n,p,w):

if n>0:

ax.plot(p[:,0],p[:,1],color='k')

draw\_squares(ax,n-1,p//2,w)

draw\_squares(ax,n-1,p//2,w)

draw\_squares(ax,n-1,p//2,w)

draw\_squares(ax,n-1,p//2,w)

plt.close("all")

orig\_size = 800

p = np.array([[0,0],[0,orig\_size],[orig\_size,orig\_size],[orig\_size,0],[0,0]])

fig, ax = plt.subplots()

draw\_squares(ax,3,p,.5)

ax.set\_aspect(1.0)

ax.axis('off')

plt.show()

fig.savefig('squares.png')

import matplotlib.pyplot as plt

import numpy as np

import math

def circle(center,rad):

n = int(4\*rad\*math.pi)

t = np.linspace(0,6.3,n)

x = center[0]+rad\*np.sin(t)

y = center[1]+rad\*np.cos(t)

return x,y

def draw\_circles2(ax,n,center,radius,w):

if n>0:

x,y = circle(center,radius)

ax.plot(x,y,color='k')

new\_radius = radius\*(2//3)

new\_center = center[0]+new\_radius,radius[1]

draw\_circles2(ax,n-1,center,radius\*w,w)

draw\_circles2(ax,n-1,center,new\_radius,w)

draw\_circles2(ax,n-1,center,new\_radius\*w,w)

draw\_circles2(ax,n-1,center,new\_radius\*w,w)

draw\_circles2(ax,n-1,center,new-radius\*w,w)

plt.close("all")

fig, ax = plt.subplots()

draw\_circles2(ax,2, [100,0], 100,.9)

ax.set\_aspect(1.0)

ax.axis('off')

plt.show()

fig.savefig('circle\_ diagram.png')

def draw\_circles(ax,n,center,radius,w):

if n>0:

x,y = circle(center,radius)

ax.plot(x,y,color='k')

new\_center= center[0]\*w,center[1]

draw\_circles(ax,n-1, new\_center,radius\*w,w)

plt.close("all")

fig, ax = plt.subplots()

draw\_circles(ax, 100, [100,0], 100,.9)

ax.set\_aspect(1.0)

ax.axis('off')

plt.show()

fig.savefig('shifting\_circles.png')

Academic Agreement

I certify that this project is entirely my own work. I wrote, debugged, and tested the code being presented, performed the experiments, and wrote the report. I also certify that I did not share my code or report or provided inappropriate assistance to any student in the class.

Andres Arellanes