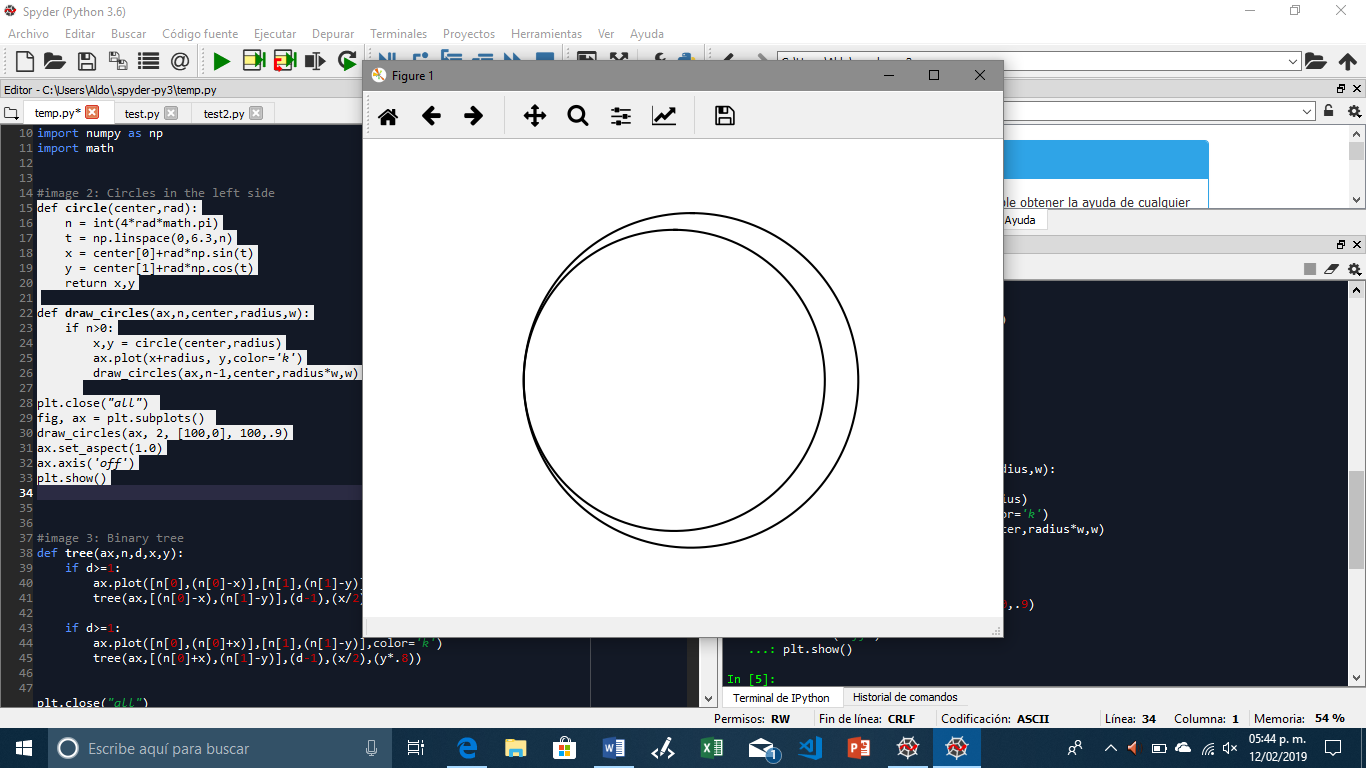
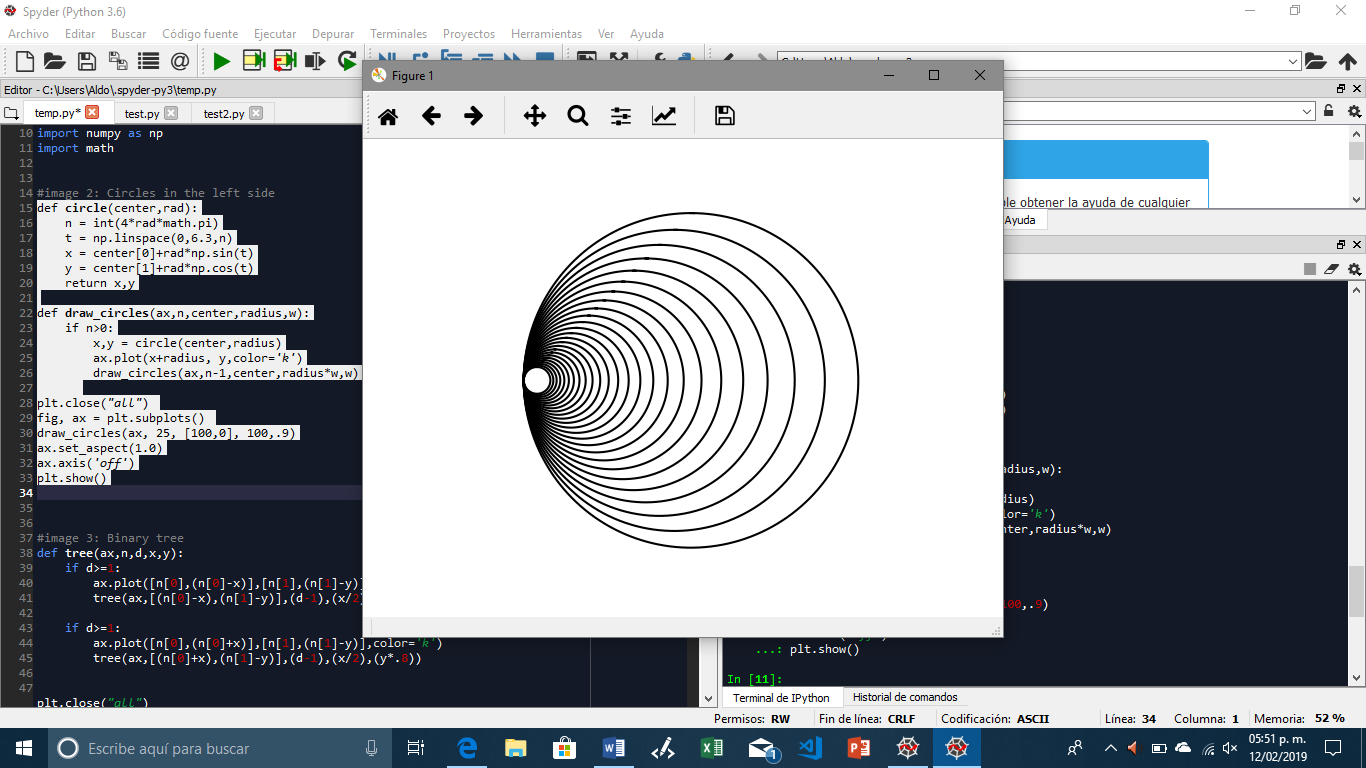
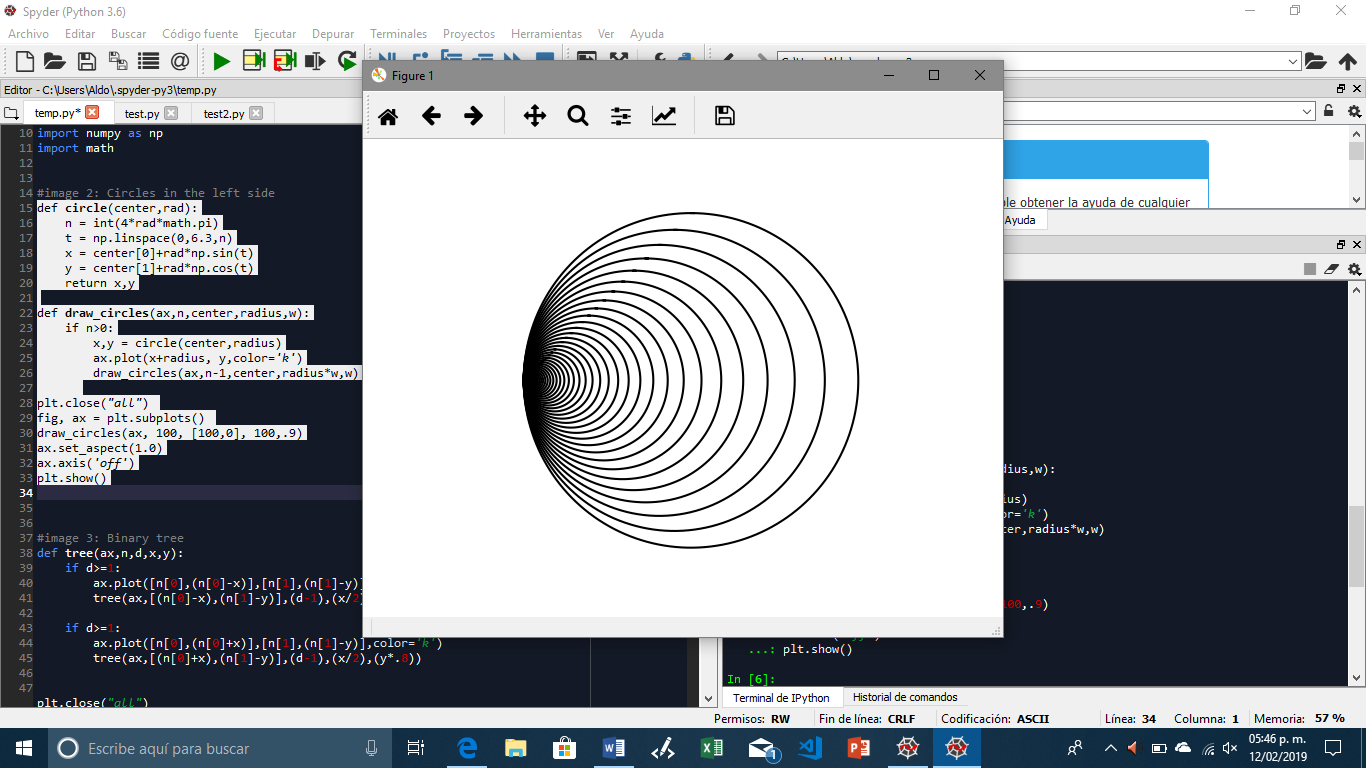
**CS2302 Lab 1**

For this lab, our assignment is to plot different figures with a program in Python, using recursive methods based on the original code provided by professor Fuentes to draw circles and squares.

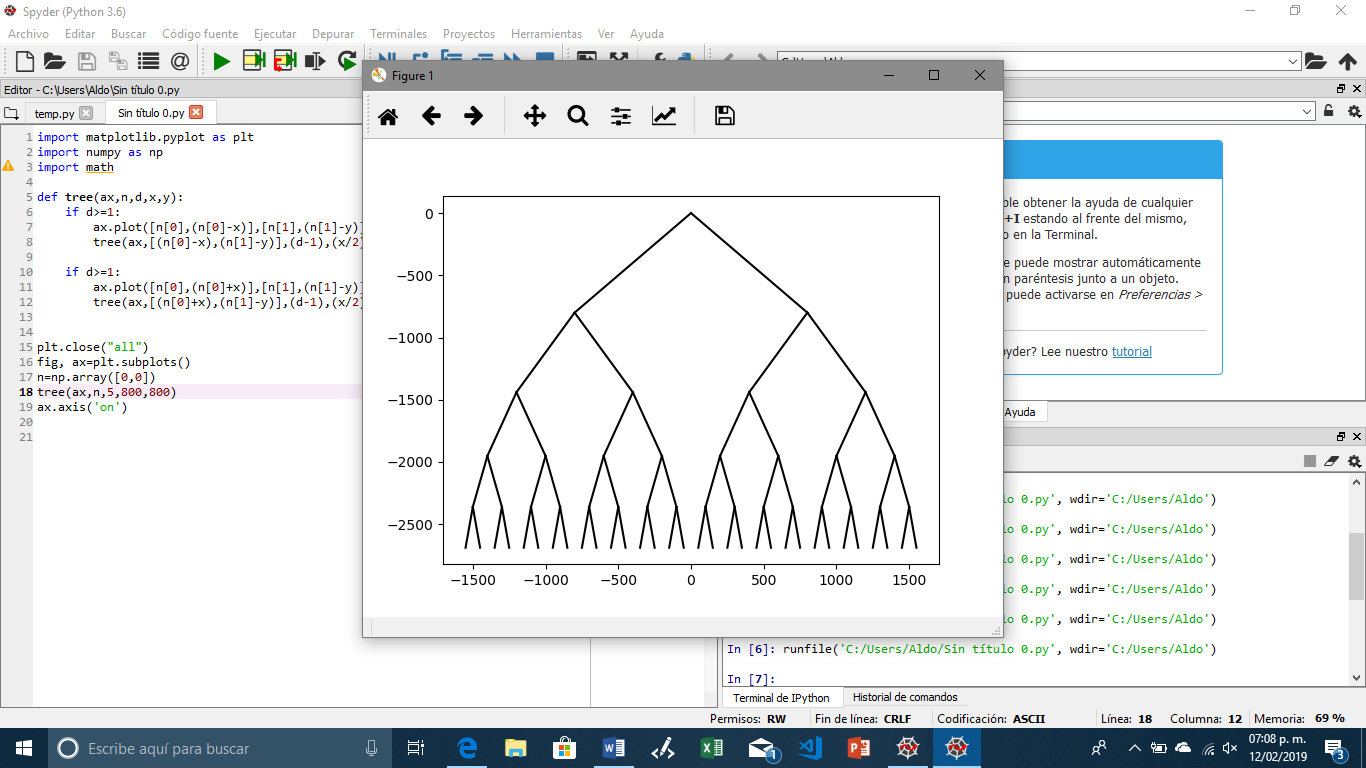
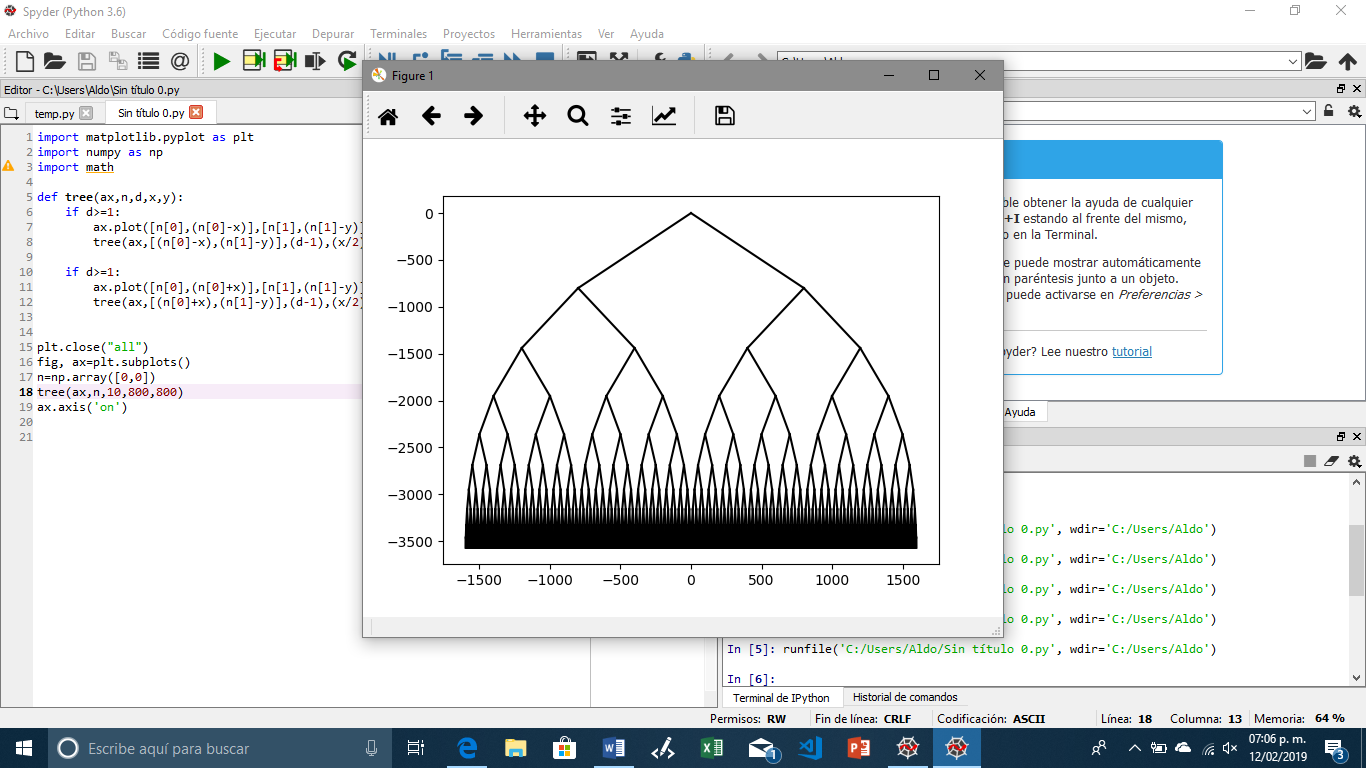
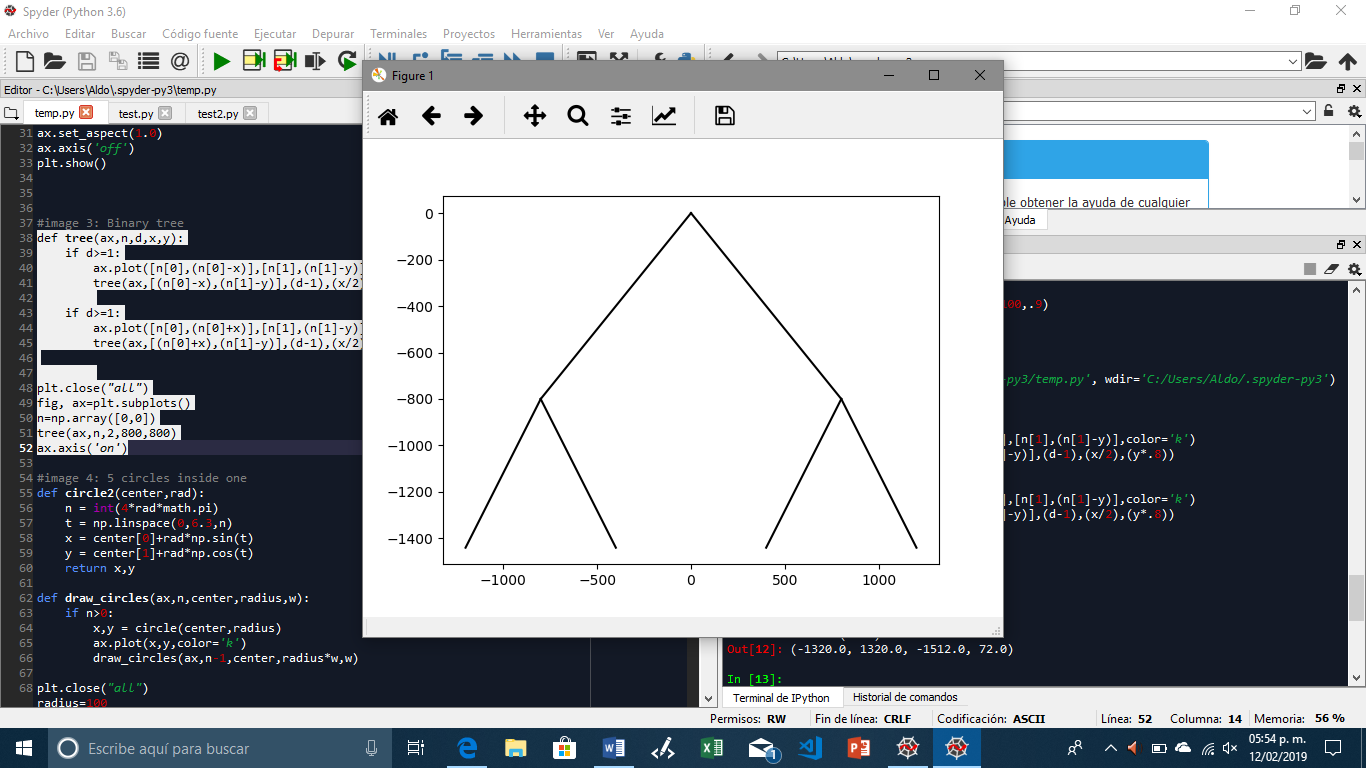
In my first attempt of trying to solve the problem, I decided to stick to original code provided, and modify some of the parameters to see how the figures where affected. This option did not help a lot, however, it helped to draw one of the figures. After seeing how this method was not being successful, I decided to apply the advises provided in class, and analyzed each figure carefully, trying to understand how it was composed, and trying to find the best way to create it with code. When we understand how each figure is constructed it’s easier to take that logic to code. It is important to mention that I was not able to complete the first figure, I was trying to make 4 recursive calls, one for each square, but my logic was not correct.

For the test cases, after completing a method, I decided to first test it with a low input, just trying to see if it was able to complete it, if it was, then I tested it with a big input to see if there were errors, or if the program breaks at certain point, after seeing that the two previous inputs work, I decide to input a middle value, trying to get a figure similar to those in the lab assignment. If they all work, then I go to the next figure. Here I have some examples of the test cases that I used for the second and third figures:

Input=2 Input=100 Input=25

Input=2 Input= 10 Input=5



In conclusion, I learned that one of the most important things when doing recursion, is to understand how our problem is working, hos is changing after every iteration, so we can apply the right parameters, and the right number of recursive calls to solve a problem.

**Code:**

import matplotlib.pyplot as plt

import numpy as np

import math

#image 2: Circles in the left side

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

if n>0:

x,y = circle(center,radius)

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

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

plt.close("all")

fig, ax = plt.subplots()

draw\_circles(ax, 25, [100,0], 100,.9)

ax.set\_aspect(1.0)

ax.axis('off')

plt.show()

#image 3: Binary tree

def tree(ax,n,d,x,y):

if d>=1:

ax.plot([n[0],(n[0]-x)],[n[1],(n[1]-y)],color='k')

tree(ax,[(n[0]-x),(n[1]-y)],(d-1),(x/2),(y\*.8))

if d>=1:

ax.plot([n[0],(n[0]+x)],[n[1],(n[1]-y)],color='k')

tree(ax,[(n[0]+x),(n[1]-y)],(d-1),(x/2),(y\*.8))

plt.close("all")

fig, ax=plt.subplots()

n=np.array([0,0])

tree(ax,n,20,800,800)

ax.axis('on')

#image 4: 5 circles inside one

def circle2(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\_circles(ax,n,center,radius,w):

if n>0:

x,y = circle(center,radius)

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

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

plt.close("all")

radius=100

fig, ax = plt.subplots()

draw\_circles(ax, 1, [100,0], 100,.9)

draw\_circles(ax, 1,[radius/3,0],radius/3,.3)

draw\_circles(ax,1,[100,0],radius/3,.3)

draw\_circles(ax,1,[2\*radius-(radius/3),0],radius/3,.3)

draw\_circles(ax,1,[100,2\*radius/3],radius/3,.3)

draw\_circles(ax,1,[100,-(2\*radius/3)],radius/3,.3)

ax.set\_aspect(1.0)

ax.axis('off')

plt.show()