**Lab #1**

CS 2302

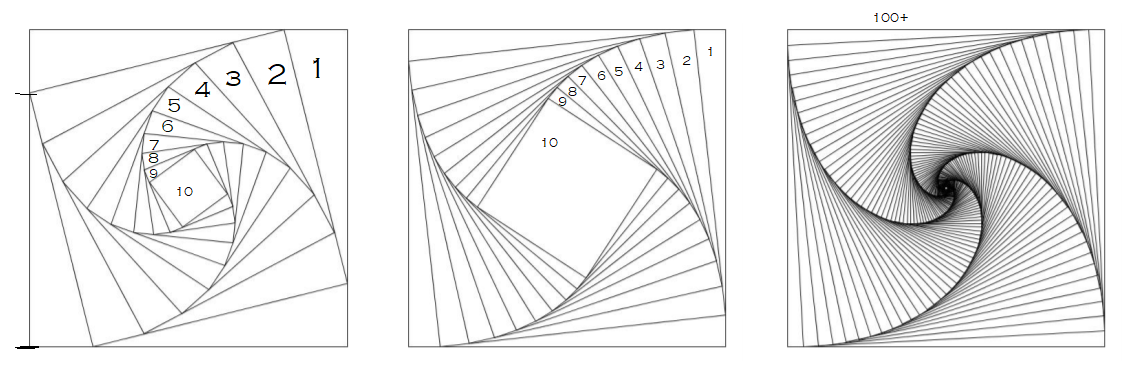
Ana Luisa Mata Sanchez

# Introduction

This assignment is focused on implementing recursion to draw repetitive figures using mathplotlib. The first problem is drawing rotating circles, the second is drawing circles with the same center that decrease in size, the third is drawing a square with 4 squares successively, the fourth is drawing circles that intersect the same point, the fifth is drawing a binary tree, and the sixth is drawing 5 equally sized circles inside one circle.

# Proposed solution design and implementation

**Module 1**



Each of these figures are drawn with the following module:

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

if n>0:

i1 = [1,2,3,0,1] 🡪 Simplifies the code

q = p\*w + p[i1]\*(1-w) 🡪 Obtains the new weighted points for the next square

ax.plot(p[:,0],p[:,1],linewidth=.5,color='k') 🡪 Plots the new square

draw\_squares(ax,n-1,q,w) 🡪 Keeps plotting new squares n times

(Made by Dr. Fuentes)

1. Figure 1 is obtained by setting the weight to .2 (effectively making the new point closer to the first of the two points in the equation) and drawing 10 squares, as illustrated in the figure

Ex. draw\_squares(ax,**10**,p,**.2**)

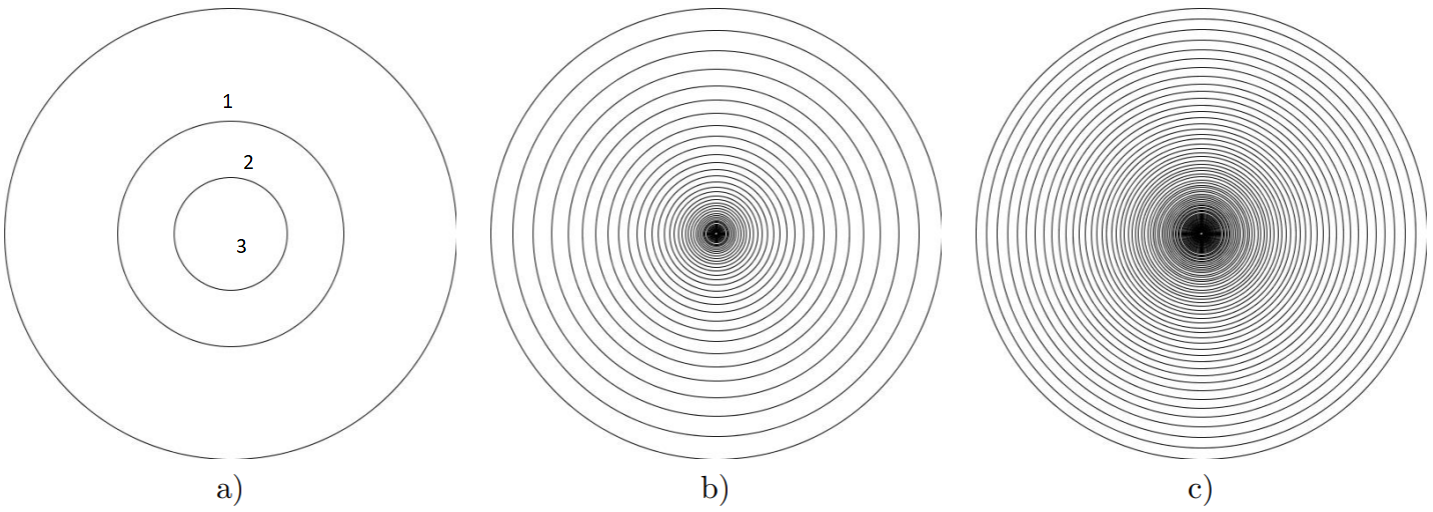
1. Figure 2 is obtained by setting the weight to .1 and drawing 10 squares, as illustrated in the figure

Ex. draw\_squares(ax,**10**,p,**.1**)

1. Figure 3 is obtained by setting the weight to .1 and drawing 70 or more squares.

Ex. draw\_squares(ax,**70**,p,**.1**)

**Module 2**



Each of these figures are drawn with the following module:

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

if n>0:

x,y = circle(center,radius) 🡪 Creates the circle itself using the circle method

ax.plot(x,y,linewidth=.5,color='k') 🡪 Plots the circle

draw\_circles(ax,n-1,center,radius\*w,w) 🡪 Makes the radius smaller by a w proportion but keeping the center the same

(Made by Dr. Fuentes)

1. Figure a is obtained by setting the weight to .5 (effectively making the new radius half of the original) and drawing 3 circles, as illustrated in the figure

Ex. draw\_circles(ax, **3**, [1000,0], 1000,**.5**)

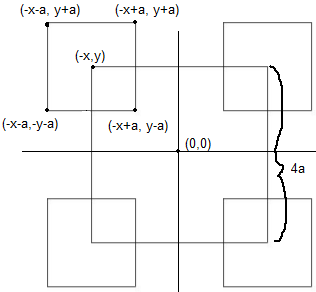
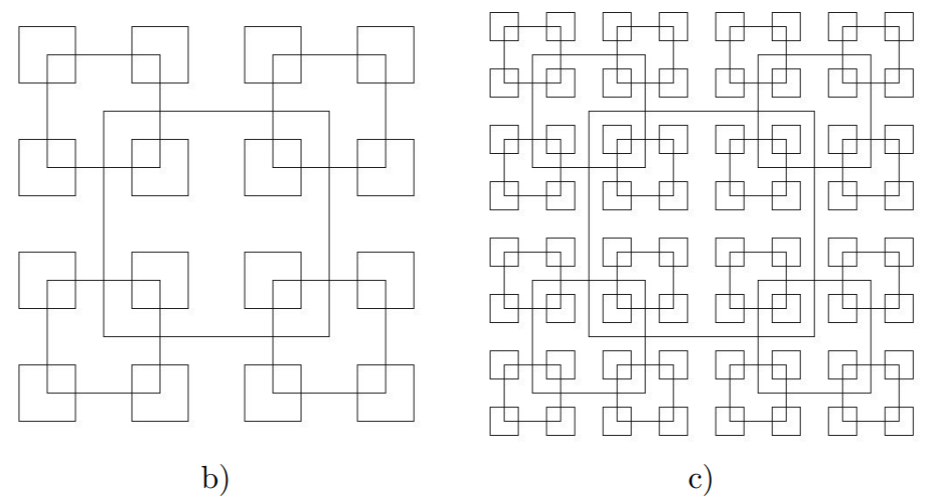
1. Figure b is obtained by setting the weight to .8 and drawing 50 circles approximately.

Ex. draw\_circles(ax, **50**, [1000,0], 1000,**.8**)

1. Figure c is obtained by setting the weight to .9 and drawing 80 circles approximately.

Ex. draw\_circles(ax, **80**, [1000,0], 1000,**.9**)

**Module 3**

Each of these figures are drawn with the following module:

def draw\_square\_children(ax,n,p):

if n>0:

ax.plot(p[:,0],p[:,1],linewidth=.5,color='k') 🡪 Draws the square

distance = math.hypot((p[1][0] - p[2][0]),(p[1][1] - p[2][1]))//4 🡨 Calculates the distance that the new points will be shifted. Represented as a in the figure

Each of the following recursive calls shift the original point 4 times in order to create a new square:

draw\_square\_children(ax,n-1,np.array([[p[0][0]-distance,p[0][1]-distance],[p[0][0]-distance,p[0][1]+distance],[p[0][0]+distance,p[0][1]+distance],[p[0][0]+distance,p[0][1]-distance],[p[0][0]-distance,p[0][1]-distance]])) 🡨 Creates the bottom left square

draw\_square\_children(ax,n-1,np.array([[p[1][0]-distance,p[1][1]-distance],[p[1][0]-distance,p[1][1]+distance],[p[1][0]+distance,p[1][1]+distance],[p[1][0]+distance,p[1][1]-distance],[p[1][0]-distance,p[1][1]-distance]])) 🡨 Creates the top left square (This is the example shown in the figure)

draw\_square\_children(ax,n-1,np.array([[p[2][0]-distance,p[2][1]-distance],[p[2][0]-distance,p[2][1]+distance],[p[2][0]+distance,p[2][1]+distance],[p[2][0]+distance,p[2][1]-distance],[p[2][0]-distance,p[2][1]-distance]])) 🡨 Creates the top right square

draw\_square\_children(ax,n-1,np.array([[p[3][0]-distance,p[3][1]-distance],[p[3][0]-distance,p[3][1]+distance],[p[3][0]+distance,p[3][1]+distance],[p[3][0]+distance,p[3][1]-distance],[p[3][0]-distance,p[3][1]-distance]])) 🡨 Creates the bottom right square\

1. Figure a is obtained by setting the center of the square at (0,0) and running the method twice.

Ex.

P=np.array([[orig\_size,orig\_size],[orig\_size,orig\_size],[orig\_size,orig\_size],[orig\_size,-orig\_size],[-orig\_size,-orig\_size]])

draw\_square\_children(ax,**2**,p)

1. Figure b is obtained by setting the center of the square at (0,0) and running the method three times.

Ex.

P=np.array([[orig\_size,orig\_size],[orig\_size,orig\_size],[orig\_size,orig\_size],[orig\_size,-orig\_size],[-orig\_size,-orig\_size]])

draw\_square\_children(ax,**3**,p)

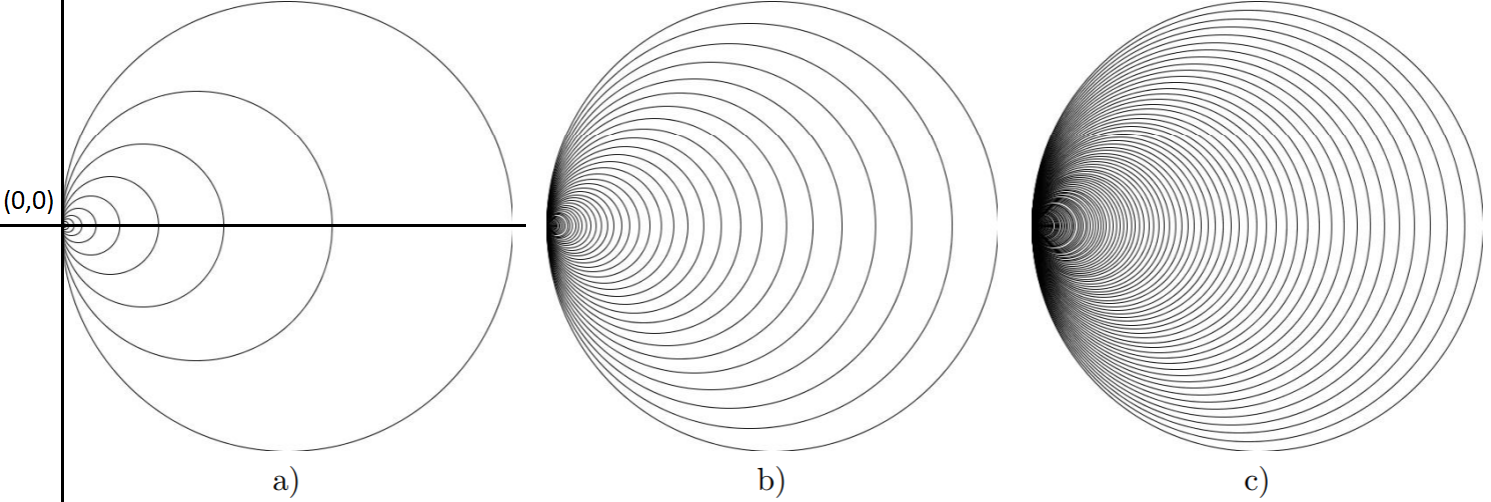
1. Figure c is obtained by setting the center of the square at (0,0) and running the method four times.

Ex.

P=np.array([[orig\_size,orig\_size],[orig\_size,orig\_size],[orig\_size,orig\_size],[orig\_size,-orig\_size],[-orig\_size,-orig\_size]])

draw\_square\_children(ax,**4**,p)

**Module 4**



Each of these figures are drawn with the following module:

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

if n>0:

x,y = circle(center,radius) 🡪 Creates the circle itself using the circle method

ax.plot(x,y,linewidth=.5,color='k') 🡪 Plots the circle

center = np.array([center[0]\*w,0]) 🡪Shifts the center towards the right by a w proportion

radius = center[0] 🡪 Sets the new radius

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

1. Figure a is obtained by making the point of intersection at (0,0), drawing 10 circles and setting weight as .6.

Ex. draw\_touching\_circles(ax, **10**, [2000,0], 2000,**.6**)

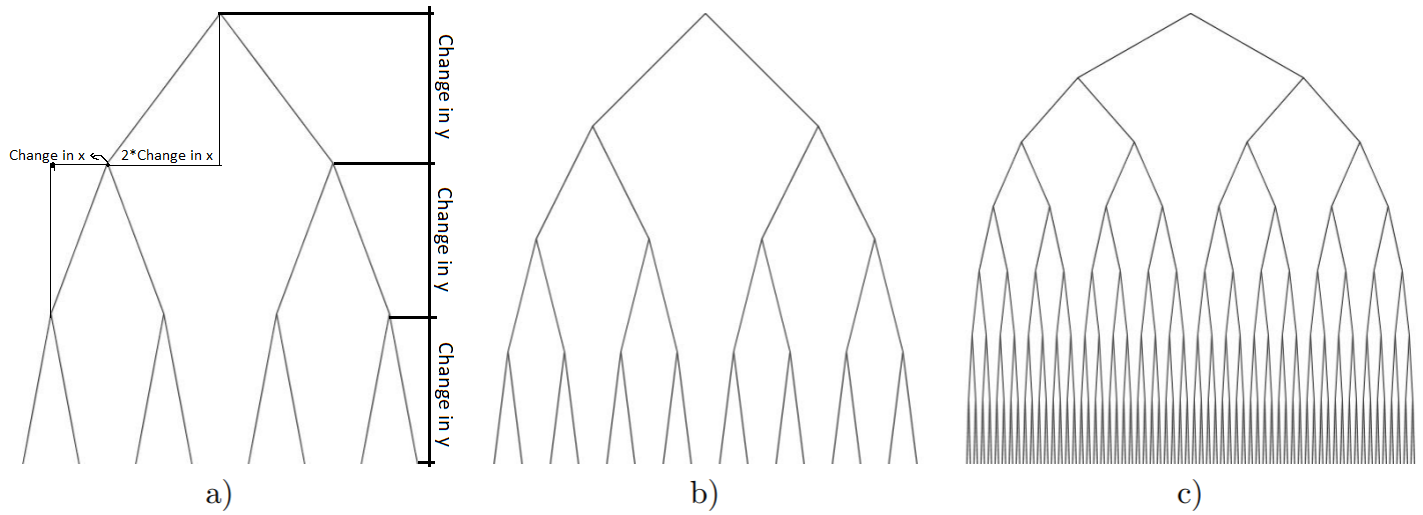
1. Figure b is obtained by making the point of intersection at (0,0), drawing 80 circles approximately and setting weight as .9.

Ex. draw\_touching\_circles(ax, **80**, [2000,0], 2000,**.9**)

1. Figure c is obtained by making the point of intersection at (0,0), drawing 120 circles and setting weight as .95.

Ex. draw\_touching\_circles(ax, **120**, [2000,0], 2000,**.95**)

**Module 5**



Each of these figures are drawn with the following module:

def draw\_bintrees(ax,n,p,deltay):

if n>0:

plt.plot(p[:,0],p[:,1],linewidth=1,color='k') 🡪 Plots the two branches connected by the same point

distance = math.hypot((p[0][0] - p[2][0]),(p[0][1] - p[2][1]))//4 🡪 Calculates the change in x (see figure)

The following two lines create two new branches by subtracting and adding the change in x correspondingly. They also subtract the change in y, that is constant.

q = np.array([[p[0][0]-distance,p[0][1] (deltay)],[p[0][0],p[0][1]],[p[0][0]+distance,p[0][1]-(deltay)]])

r = np.array([[p[2][0]-distance,p[2][1]-(deltay)],[p[2][0],p[2][1]],[p[2][0]+distance,p[2][1]-(deltay)]])

draw\_bintrees(ax,n-1,q,deltay)

draw\_bintrees(ax,n-1,r,deltay)

1. Figure a is obtained by making the first point (0,0) and defining the change in y. Then the method calls itself 3 times.

Ex.

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

draw\_bintrees(ax,**3**,p,orig\_size)

1. Figure b is obtained by making the first point (0,0) and defining the change in y. Then the method calls itself 4 times.

Ex.

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

draw\_bintrees(ax,**4**,p,orig\_size)

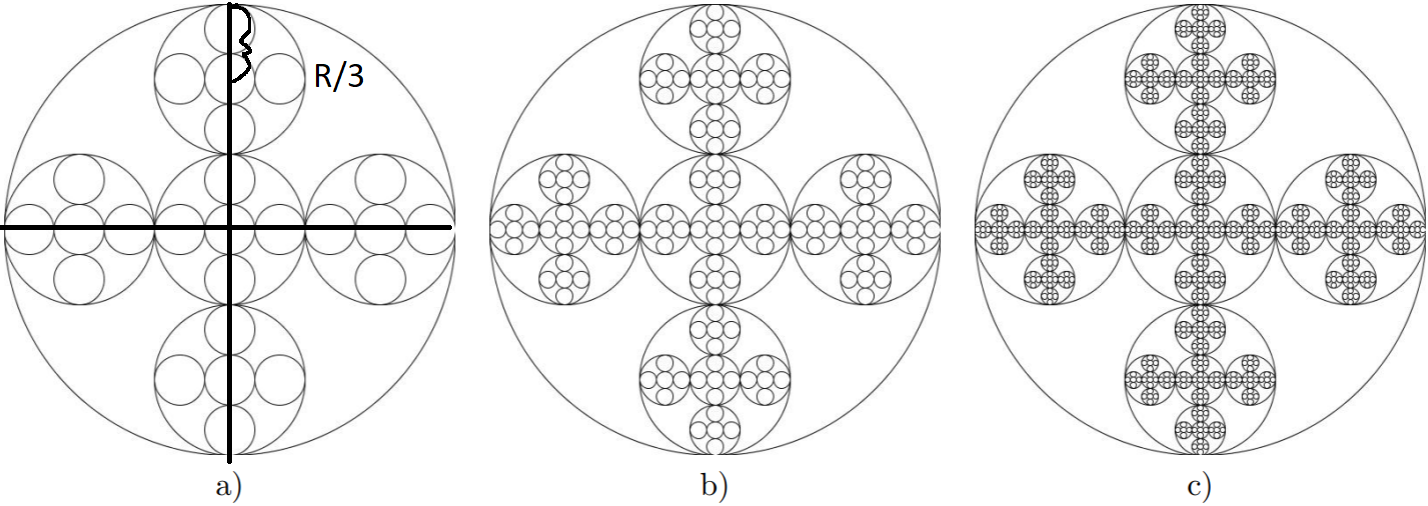
1. Figure a is obtained by making the first point (0,0) and defining the change in y. Then the method calls itself 6 times.

Ex.

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

draw\_bintrees(ax,**6**,p,orig\_size)

**Module 5**



Each of these figures are drawn with the following module:

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

if n>0:

x,y = circle(center,radius) 🡪 Creates the circle

ax.plot(x,y,linewidth=.5,color='k') 🡪 Plots the circle

The following lines create each one of the new circles by shifting the center (up, down, left, right, and center) and calculating a new radius that is a third of the original

draw\_five\_circles(ax,n-1,[center[0],center[1]+(radius-((radius)//3))],(radius)/3) 🡪 Creates the upper circle

draw\_five\_circles(ax,n-1,[center[0]+(radius-((radius)//3)),center[1]],radius/3)

🡪 Creates the right circle

draw\_five\_circles(ax,n-1,[center[0]+((-radius+(radius)//3)),center[1]],radius/3)

🡪 Creates the left circle

draw\_five\_circles(ax,n-1,center,radius/3)

🡪 Creates the center circle

draw\_five\_circles(ax,n-1,[center[0],center[1]+((-radius+(radius)//3))],(radius)/3)

🡪 Creates the bottom circle

1. Figure a is obtained by making the method call itself 3 times.

Ex. draw\_five\_circles(ax, **3**, [0,0], 1000)

1. Figure a is obtained by making the method call itself 4 times.

Ex. draw\_five\_circles(ax, **4**, [0,0], 1000)

1. Figure a is obtained by making the method call itself 5times.

Ex. draw\_five\_circles(ax, **5**, [0,0], 1000)\

# Experimental results

Using the following method calls:

**\*Differences between shapes marked in red**

|  |  |
| --- | --- |
| Input | Output |
| ####### Square A #######  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 , **10** , p , **.8**)  ax.set\_aspect(1.0)  ax.axis('off')  plt.show()  fig.savefig('SquaresA.png') | Square A |
| ######### Square B ################  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 , **10** , p , **.1**)  ax.set\_aspect(1.0)  ax.axis('off')  plt.show()  fig.savefig('SquaresB.png') | Square B |
| ########## Square C ########  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 , **70** , p , **.1**)  ax.set\_aspect(1.0)  ax.axis('off')  plt.show()  fig.savefig('SquaresC.png') | Square C |
| ####### Circle A ######  plt.close("all")  fig, ax = plt.subplots()  draw\_circles(ax, **3**, [1000,0], 1000, **.5**)  ax.set\_aspect(1.0)  ax.axis('off')  plt.show()  fig.savefig('CirclesA.png') | Circles A |
| ######## Circle B #########  plt.close("all")  fig, ax = plt.subplots()  draw\_circles(ax, **50**, [1000,0], 1000, **.83**)  ax.set\_aspect(1.0)  ax.axis('off')  plt.show()  fig.savefig('CirclesB.png') | Squares B |
| ####### Circle C #######  plt.close("all")  fig, ax = plt.subplots()  draw\_circles(ax, **80**, [1000,0], 1000, **.92**)  ax.set\_aspect(1.0)  ax.axis('off')  plt.show()  fig.savefig('CirclesC.png') | Squares C |
| ####### Children Squares A #####  plt.close("all")  orig\_size = 1000  p = np.array([[-orig\_size,-orig\_size],[-orig\_size,orig\_size],[orig\_size,orig\_size],[orig\_size,-orig\_size],[-orig\_size,-orig\_size]])  fig, ax = plt.subplots()  draw\_square\_children(ax, **2**, p)  ax.set\_aspect(1.0)  ax.axis('off')  plt.show()  fig.savefig('ChildSquaresA.png') | ChildSquares A |
| ####### Children Squares B #########  plt.close("all")  orig\_size = 1000  p = np.array([[-orig\_size,-orig\_size],[-orig\_size,orig\_size],[orig\_size,orig\_size],[orig\_size,-orig\_size],[-orig\_size,-orig\_size]])  fig, ax = plt.subplots()  draw\_square\_children(ax, **3**, p)  ax.set\_aspect(1.0)  ax.axis('off')  plt.show()  fig.savefig('ChildSquaresB.png') | ChildSquares B |
| ###### Children Squares C #########  plt.close("all")  orig\_size = 1000  p = np.array([[-orig\_size,-orig\_size],[-orig\_size,orig\_size],[orig\_size,orig\_size],[orig\_size,-orig\_size],[-orig\_size,-orig\_size]])  fig, ax = plt.subplots()  draw\_square\_children(ax, **4**, p)  ax.set\_aspect(1.0)  ax.axis('off')  plt.show()  fig.savefig('ChildSquaresC.png') | ChildSquares C |
| ##### Touching Circles A ######  plt.close("all")  fig, ax = plt.subplots()  draw\_touching\_circles(ax, **10**, [2000,0], 2000, **.62**)  ax.set\_aspect(1.0)  ax.axis('off')  plt.show()  fig.savefig('TouchingCirclesA.png') | TouchingCircles A |
| ###### Touching Circles B ######  plt.close("all")  fig, ax = plt.subplots()  draw\_touching\_circles(ax, **80**, [1000,0], 1000, **.9**)  ax.set\_aspect(1.0)  ax.axis('off')  plt.show()  fig.savefig('TouchingCirclesB.png') | TouchingCircles B |
| ###### Touching Circles C ######  plt.close("all")  fig, ax = plt.subplots()  draw\_touching\_circles(ax, **120**, [1000,0], 1000, **.95**)  ax.set\_aspect(1.0)  ax.axis('off')  plt.show()  fig.savefig('TouchingCirclesC.png') | TouchingCircles C |
| ######### Binary Tree A ###########  plt.close("all")  orig\_size = 1000  p = np.array([[-orig\_size,-orig\_size],[0,0],[orig\_size,-orig\_size]])  fig, ax = plt.subplots()  draw\_bintrees(ax, **3**, p, orig\_size)  ax.set\_aspect(1.0)  ax.axis('off')  plt.show()  fig.savefig('BinTreeA.png') | BinTree A |
| ######### Binary Tree B ############  plt.close("all")  orig\_size = 1000  p = np.array([[-orig\_size,-orig\_size],[0,0],[orig\_size,-orig\_size]])  fig, ax = plt.subplots()  draw\_bintrees(ax, **4**, p, orig\_size)  ax.set\_aspect(1.0)  ax.axis('off')  plt.show()  fig.savefig('BinTreeB.png') | BinTree B |
| ######## Binary Tree C #########  plt.close("all")  orig\_size = 1000  p = np.array([[-orig\_size-500,-orig\_size],[0,0],[orig\_size+500,-orig\_size]])  fig, ax = plt.subplots()  draw\_bintrees(ax, **6**, p, orig\_size)  ax.set\_aspect(1.0)  ax.axis('off')  plt.show()  fig.savefig('BinTreeC.png') | BinTree C |
| ###### Five Circles A ########  plt.close("all")  fig, ax = plt.subplots()  draw\_five\_circles(ax, **3**, [0,0], 1000)  ax.set\_aspect(1.0)  ax.axis('off')  plt.show()  fig.savefig('FiveCirclesA.png') | FiveCircles A |
| ##### Five Circles B #####  plt.close("all")  fig, ax = plt.subplots()  draw\_five\_circles(ax, **4**, [0,0], 1000)  ax.set\_aspect(1.0)  ax.axis('off')  plt.show()  fig.savefig('FiveCirclesB.png') | FiveCircles B |
| #### Five Circles C ####  plt.close("all")  fig, ax = plt.subplots()  draw\_five\_circles(ax, **5**, [0,0], 1000)  ax.set\_aspect(1.0)  ax.axis('off')  plt.show()  fig.savefig('FiveCirclesC.png') | FiveCircles C |

# Appendix

|  |  |
| --- | --- |
|  | # Course: CS2302 |
|  | # Assignment: Lab #1 |
|  | # Instructor: Dr. Olac Fuentes |
|  | # Description: Practice using recursion to draw interesting figures. |
|  | # T.A.: Anindita Nath and Maliheh Zargaran |
|  | # Last modified: 02/08/2019 |
|  | # Purpose: Plot figures using matplotlib and recursion. |
|  |  |
|  | import numpy as np |
|  | import matplotlib.pyplot as plt |
|  | import math |
|  |  |
|  | #Draws the rotating squares |
|  | def draw\_squares(ax,n,p,w): |
|  | if n>0: |
|  | i1 = [1,2,3,0,1] |
|  | q = p\*w + p[i1]\*(1-w) |
|  | ax.plot(p[:,0],p[:,1],linewidth=.5,color='k') |
|  | draw\_squares(ax,n-1,q,w) |
|  |  |
|  | #Draws the 4 squares |
|  | def draw\_square\_children(ax,n,p): |
|  | if n>0: |
|  | #Draws the square |
|  | ax.plot(p[:,0],p[:,1],linewidth=.5,color='k') |
|  | #calculates the distance that the new points will be shifted to |
|  | distance = math.hypot((p[1][0] - p[2][0]),(p[1][1] - p[2][1]))//4 |
|  | #creates each one of the new squares by shifting the original point |
|  | draw\_square\_children(ax,n-1,np.array([[p[0][0]-distance,p[0][1]-distance],[p[0][0]-distance,p[0][1]+distance],[p[0][0]+distance,p[0][1]+distance],[p[0][0]+distance,p[0][1]-distance],[p[0][0]-distance,p[0][1]-distance]])) |
|  | draw\_square\_children(ax,n-1,np.array([[p[1][0]-distance,p[1][1]-distance],[p[1][0]-distance,p[1][1]+distance],[p[1][0]+distance,p[1][1]+distance],[p[1][0]+distance,p[1][1]-distance],[p[1][0]-distance,p[1][1]-distance]])) |
|  | draw\_square\_children(ax,n-1,np.array([[p[2][0]-distance,p[2][1]-distance],[p[2][0]-distance,p[2][1]+distance],[p[2][0]+distance,p[2][1]+distance],[p[2][0]+distance,p[2][1]-distance],[p[2][0]-distance,p[2][1]-distance]])) |
|  | draw\_square\_children(ax,n-1,np.array([[p[3][0]-distance,p[3][1]-distance],[p[3][0]-distance,p[3][1]+distance],[p[3][0]+distance,p[3][1]+distance],[p[3][0]+distance,p[3][1]-distance],[p[3][0]-distance,p[3][1]-distance]])) |
|  |  |
|  | #Creates the circle |
|  | 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 |
|  |  |
|  | #Draws the concentric circles |
|  | def draw\_circles(ax,n,center,radius,w): |
|  | if n>0: |
|  | x,y = circle(center,radius) |
|  | ax.plot(x,y,linewidth=.5,color='k') |
|  | draw\_circles(ax,n-1,center,radius\*w,w) |
|  |  |
|  | #Draws the circles that intersect the same point (0,0) |
|  | def draw\_touching\_circles(ax,n,center,radius,w): |
|  | if n>0: |
|  | #Create circle and plot |
|  | x,y = circle(center,radius) |
|  | ax.plot(x,y,linewidth=.5,color='k') |
|  | #shifts the radius towards the right w proportion |
|  | center = np.array([center[0]\*w,0]) |
|  | #New radius for the new circle |
|  | radius = center[0] |
|  | draw\_touching\_circles(ax,n-1,center,radius,w) |
|  |  |
|  | #Draws five circles inside one |
|  | def draw\_five\_circles(ax,n,center,radius): |
|  | if n>0: |
|  | #create and plot circle |
|  | x,y = circle(center,radius) |
|  | ax.plot(x,y,linewidth=.5,color='k') |
|  | #Create each one of the new circles by shifting the center and calculating a new radius that is a third of the original |
|  | draw\_five\_circles(ax,n-1,[center[0],center[1]+(radius-((radius)//3))],(radius)/3) |
|  | draw\_five\_circles(ax,n-1,[center[0]+(radius-((radius)//3)),center[1]],radius/3) |
|  | draw\_five\_circles(ax,n-1,[center[0]+((-radius+(radius)//3)),center[1]],radius/3) |
|  | draw\_five\_circles(ax,n-1,center,radius/3) |
|  | draw\_five\_circles(ax,n-1,[center[0],center[1]+((-radius+(radius)//3))],(radius)/3) |
|  |  |
|  | #Draws binary trees |
|  | def draw\_bintrees(ax,n,p,deltay): |
|  | if n>0: |
|  | #Plots the two branches connected by the same point |
|  | plt.plot(p[:,0],p[:,1],linewidth=1,color='k') |
|  | #Calculates the change in x that is a fourth of the original distance |
|  | distance = math.hypot((p[0][0] - p[2][0]),(p[0][1] - p[2][1]))//4 |
|  | #Creates two new branches by subtracting and adding the change in x that is the distance. It subtracts the change in y, that is the height of the first two branches. |
|  | q = np.array([[p[0][0]-distance,p[0][1]-(deltay)],[p[0][0],p[0][1]],[p[0][0]+distance,p[0][1]-(deltay)]]) |
|  | r = np.array([[p[2][0]-distance,p[2][1]-(deltay)],[p[2][0],p[2][1]],[p[2][0]+distance,p[2][1]-(deltay)]]) |
|  | draw\_bintrees(ax,n-1,q,deltay) |
|  | draw\_bintrees(ax,n-1,r,deltay) |
|  |  |
|  | try: |
|  | ################################ Square A ################################ |
|  | 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,10,p,.8) |
|  | ax.set\_aspect(1.0) |
|  | ax.axis('off') |
|  | plt.show() |
|  | fig.savefig('SquaresA.png') |
|  |  |
|  | ################################ Square B ################################ |
|  | 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,10,p,.1) |
|  | ax.set\_aspect(1.0) |
|  | ax.axis('off') |
|  | plt.show() |
|  | fig.savefig('SquaresB.png') |
|  |  |
|  | ################################ Square C ################################ |
|  | 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,70,p,.1) |
|  | ax.set\_aspect(1.0) |
|  | ax.axis('off') |
|  | plt.show() |
|  | fig.savefig('SquaresC.png') |
|  |  |
|  | ################################ Circle A ################################ |
|  | plt.close("all") |
|  | fig, ax = plt.subplots() |
|  | draw\_circles(ax, 3, [1000,0], 1000,.5) |
|  | ax.set\_aspect(1.0) |
|  | ax.axis('off') |
|  | plt.show() |
|  | fig.savefig('CirclesA.png') |
|  |  |
|  | ################################ Circle B ################################ |
|  | plt.close("all") |
|  | fig, ax = plt.subplots() |
|  | draw\_circles(ax, 50, [1000,0], 1000,.83) |
|  | ax.set\_aspect(1.0) |
|  | ax.axis('off') |
|  | plt.show() |
|  | fig.savefig('CirclesB.png') |
|  |  |
|  | ################################ Circle C ################################ |
|  | plt.close("all") |
|  | fig, ax = plt.subplots() |
|  | draw\_circles(ax, 80, [1000,0], 1000,.92) |
|  | ax.set\_aspect(1.0) |
|  | ax.axis('off') |
|  | plt.show() |
|  | fig.savefig('CirclesC.png') |
|  |  |
|  | ################################ Children Squares A ################################ |
|  | plt.close("all") |
|  | orig\_size = 1000 |
|  | p = np.array([[-orig\_size,-orig\_size],[-orig\_size,orig\_size],[orig\_size,orig\_size],[orig\_size,-orig\_size],[-orig\_size,-orig\_size]]) |
|  | fig, ax = plt.subplots() |
|  | draw\_square\_children(ax,2,p) |
|  | ax.set\_aspect(1.0) |
|  | ax.axis('off') |
|  | plt.show() |
|  | fig.savefig('ChildSquaresA.png') |
|  |  |
|  | ################################ Children Squares B ################################ |
|  | plt.close("all") |
|  | orig\_size = 1000 |
|  | p = np.array([[-orig\_size,-orig\_size],[-orig\_size,orig\_size],[orig\_size,orig\_size],[orig\_size,-orig\_size],[-orig\_size,-orig\_size]]) |
|  | fig, ax = plt.subplots() |
|  | draw\_square\_children(ax,3,p) |
|  | ax.set\_aspect(1.0) |
|  | ax.axis('off') |
|  | plt.show() |
|  | fig.savefig('ChildSquaresB.png') |
|  |  |
|  | ################################ Children Squares C ################################ |
|  | plt.close("all") |
|  | orig\_size = 1000 |
|  | p = np.array([[-orig\_size,-orig\_size],[-orig\_size,orig\_size],[orig\_size,orig\_size],[orig\_size,-orig\_size],[-orig\_size,-orig\_size]]) |
|  | fig, ax = plt.subplots() |
|  | draw\_square\_children(ax,4,p) |
|  | ax.set\_aspect(1.0) |
|  | ax.axis('off') |
|  | plt.show() |
|  | fig.savefig('ChildSquaresC.png') |
|  |  |
|  | ################################ Touching Circles A ################################ |
|  | plt.close("all") |
|  | fig, ax = plt.subplots() |
|  | draw\_touching\_circles(ax, 10, [2000,0], 2000,.62) |
|  | ax.set\_aspect(1.0) |
|  | ax.axis('off') |
|  | plt.show() |
|  | fig.savefig('TouchingCirclesA.png') |
|  |  |
|  | ################################ Touching Circles B ################################ |
|  | plt.close("all") |
|  | fig, ax = plt.subplots() |
|  | draw\_touching\_circles(ax, 80, [1000,0], 1000,.9) |
|  | ax.set\_aspect(1.0) |
|  | ax.axis('off') |
|  | plt.show() |
|  | fig.savefig('TouchingCirclesB.png') |
|  |  |
|  | ################################ Touching Circles C ################################ |
|  | plt.close("all") |
|  | fig, ax = plt.subplots() |
|  | draw\_touching\_circles(ax, 120, [1000,0], 1000,.95) |
|  | ax.set\_aspect(1.0) |
|  | ax.axis('off') |
|  | plt.show() |
|  | fig.savefig('TouchingCirclesC.png') |
|  |  |
|  | ################################ Binary Tree A ################################ |
|  | plt.close("all") |
|  | orig\_size = 1000 |
|  | p = np.array([[-orig\_size,-orig\_size],[0,0],[orig\_size,-orig\_size]]) |
|  | fig, ax = plt.subplots() |
|  | draw\_bintrees(ax,3,p,orig\_size) |
|  | ax.set\_aspect(1.0) |
|  | ax.axis('off') |
|  | plt.show() |
|  | fig.savefig('BinTreeA.png') |
|  |  |
|  | ################################ Binary Tree B ################################ |
|  | plt.close("all") |
|  | orig\_size = 1000 |
|  | p = np.array([[-orig\_size,-orig\_size],[0,0],[orig\_size,-orig\_size]]) |
|  | fig, ax = plt.subplots() |
|  | draw\_bintrees(ax,4,p,orig\_size) |
|  | ax.set\_aspect(1.0) |
|  | ax.axis('off') |
|  | plt.show() |
|  | fig.savefig('BinTreeB.png') |
|  |  |
|  | ################################ Binary Tree C ################################ |
|  | plt.close("all") |
|  | orig\_size = 1000 |
|  | p = np.array([[-orig\_size-500,-orig\_size],[0,0],[orig\_size+500,-orig\_size]]) |
|  | fig, ax = plt.subplots() |
|  | draw\_bintrees(ax,6,p,orig\_size) |
|  | ax.set\_aspect(1.0) |
|  | ax.axis('off') |
|  | plt.show() |
|  | fig.savefig('BinTreeC.png') |
|  |  |
|  | ################################ Five Circles A ################################ |
|  | plt.close("all") |
|  | fig, ax = plt.subplots() |
|  | draw\_five\_circles(ax, 3, [1000,0], 1000) |
|  | ax.set\_aspect(1.0) |
|  | ax.axis('off') |
|  | plt.show() |
|  | fig.savefig('FourCirclesA.png') |
|  |  |
|  | ################################ Five Circles B ################################ |
|  | plt.close("all") |
|  | fig, ax = plt.subplots() |
|  | draw\_five\_circles(ax, 4, [1000,0], 1000) |
|  | ax.set\_aspect(1.0) |
|  | ax.axis('off') |
|  | plt.show() |
|  | fig.savefig('FourCirclesB.png') |
|  |  |
|  | ################################ Five Circles C ################################ |
|  | plt.close("all") |
|  | fig, ax = plt.subplots() |
|  | draw\_five\_circles(ax, 5, [1000,0], 1000) |
|  | ax.set\_aspect(1.0) |
|  | ax.axis('off') |
|  | plt.show() |
|  | fig.savefig('FourCirclesC.png') |
|  |  |
|  | except TypeError: |
|  | print("There's an error in a method call!") |