Toshiki Kamio

CS 2302 Lab#1

Spring 2019

1. Introduction

This report is for CS2302 Lab1. There are four questions and each of them requires generating three figures using recursion.

1. Proposed solution design and implementation

The solution consists of two parts. The first part is to create a list which store x- and y- coordinations of expected figure to draw. The second part calls the first part and plot the figure. The second part also call itself with slight change of conditions to complete drawing the expected figure recursively.

1. Experimental results

Experiment environment is as below.

Computer: ASUS Vivobook 15

Processor: AMD Ryzen 5 2500U with Radeon Vega Mobile Gfx 2.00GHz

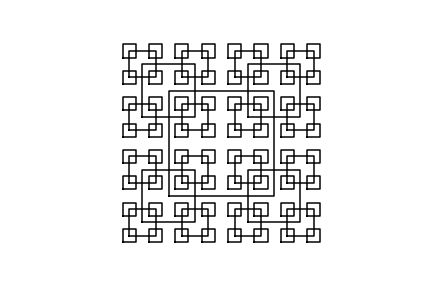
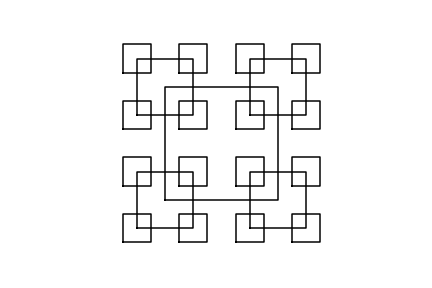
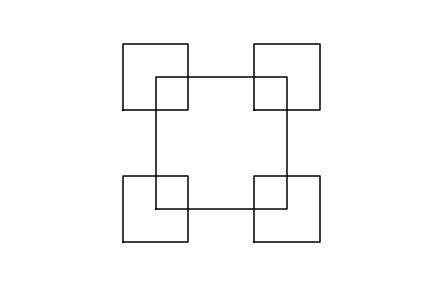
Memory: 8.00 GB (6.95 GB usable)

OS: Windows 10 Home

Python version: Python 3.7

The outputs are as following.

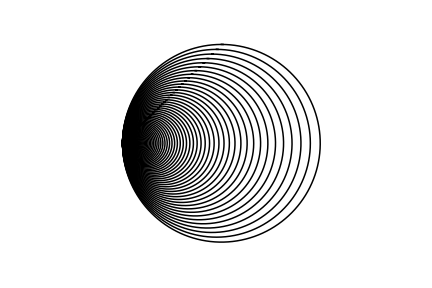
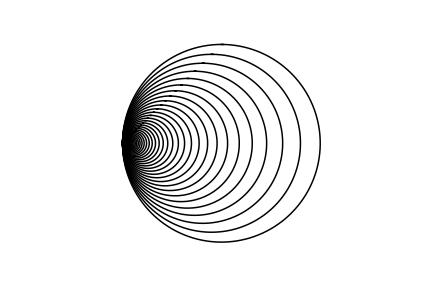
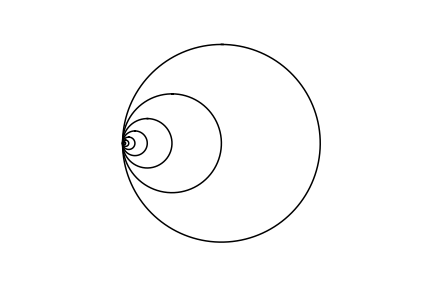
-1. a), b), c)



Runtime was,

a) 0.0156317(sec) b) 0.031261 (sec) c) 0.116158 (sec)

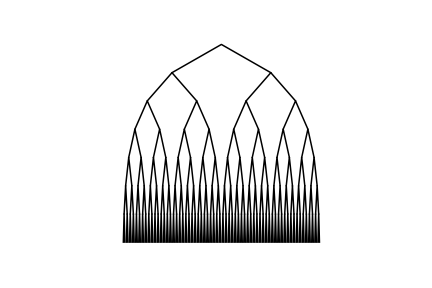
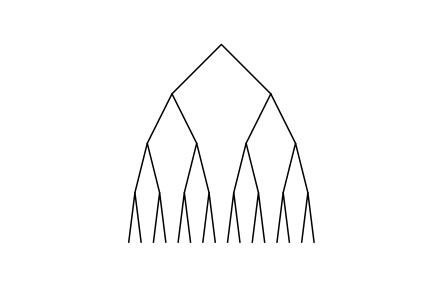
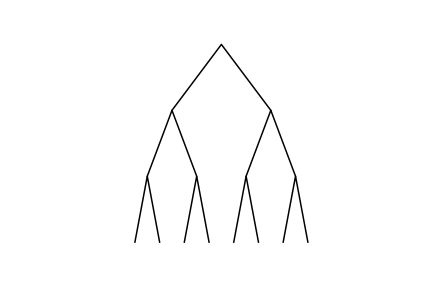
-2. a), b), c)



Runtime was,

a) 0.0534494 (sec) b) 0.0466363 (sec) c) 0.100327 (sec)

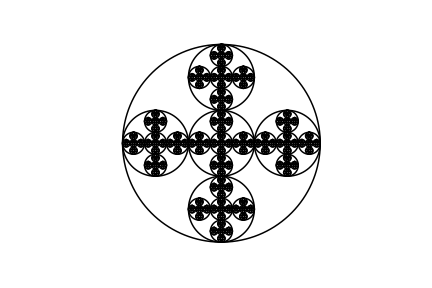
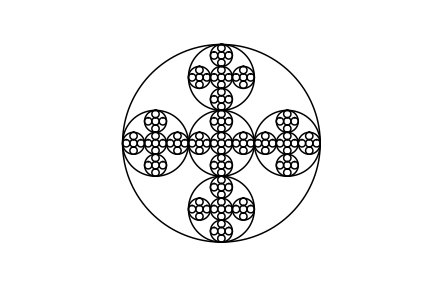
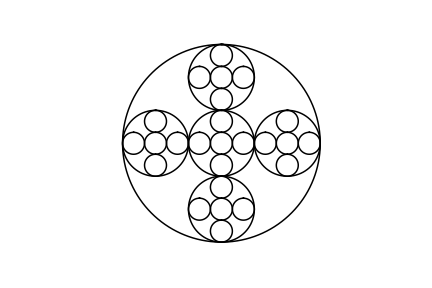
-3. a), b), c)



Runtime was,

a) 0.015619 (sec) b) 0.0156298 (sec) c) 0.169053 (sec)

-4. a), b), c)



Runtime was,

a) 0.0312605 (sec) b) 0.147423 (sec) c) 1.21575 (sec)

1. Conclusion

What I learned from this project was to use Python and to draw figures using programming. Both were first time for me. Also, I couldn’t find good online compiler supporting matplotlib and numpy at the same time. I bought a new laptop to finish this lab. That was an expensive lesson.

1. Appendix

Source codes for each figure are following.

-1. a)

'''

This program writes a recursive method to draw given figures in CS2302 Lab1,

Question 1.

'''

import matplotlib.pyplot as plt

# Create a list of x- and y- coordination based on center and side

def square(center, side):

x = [center[0]-1\*side/2, center[0]-1\*side/2, center[0]+side/2, center[0]+side/2, center[0]-1\*side/2]

y = [center[1]-1\*side/2, center[1]+side/2, center[1]+side/2, center[1]-1\*side/2, center[1]-1\*side/2]

return x, y

'''

draw\_squares draws squares recursively. The first square is drawn based on the

given center and side. The consecutive squares are drawn from bottom left,

top left, top right, and bottom right cornor respectively. The user can specify

how the consecutive squares change their side by variable w.

'''

def draw\_squares(ax,n,center,side,w):

if n>0:

# Create and draw a square based on given center and side.

x, y = square(center, side)

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

# Repeat to each corner of the square

draw\_squares(ax, n-1, [center[0]-side/2, center[1]-side/2], side\*w, w)

draw\_squares(ax, n-1, [center[0]-side/2, center[1]+side/2], side\*w, w)

draw\_squares(ax, n-1, [center[0]+side/2, center[1]+side/2], side\*w, w)

draw\_squares(ax, n-1, [center[0]+side/2, center[1]-side/2], side\*w, w)

plt.close("all")

center = [0, 0]

side = 800

weight = 0.5

numRepeat = 2

fig, ax = plt.subplots()

draw\_squares(ax, numRepeat, center, side, weight)

ax.set\_aspect(1.0)

ax.axis('off')

plt.show()

fig.savefig('Lab1\_1\_a.png')

-1. b)

'''

This program write a recursive method to draw given figures in CS2302 Lab1,

Question 1.

'''

import matplotlib.pyplot as plt

# Create a list of x- and y- coordination based on center and side

def square(center, side):

x = [center[0]-1\*side/2, center[0]-1\*side/2, center[0]+side/2, center[0]+side/2, center[0]-1\*side/2]

y = [center[1]-1\*side/2, center[1]+side/2, center[1]+side/2, center[1]-1\*side/2, center[1]-1\*side/2]

return x, y

'''

draw\_squares draws squares recursively. The first square is drawn based on the given center

and side. The second to fifth squares are drawn at bottom left, top left, top right, and bottom

right cornor respectively. The user can specify how the second to fifth squares change their side

by variable w.

'''

def draw\_squares(ax,n,center,side,w):

if n>0:

# Create and draw a square base on given center and side.

x, y = square(center, side)

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

# Repeat to each corner of the square

draw\_squares(ax, n-1, [center[0]-side/2, center[1]-side/2], side\*w, w)

draw\_squares(ax, n-1, [center[0]-side/2, center[1]+side/2], side\*w, w)

draw\_squares(ax, n-1, [center[0]+side/2, center[1]+side/2], side\*w, w)

draw\_squares(ax, n-1, [center[0]+side/2, center[1]-side/2], side\*w, w)

plt.close("all")

center = [0, 0]

side = 800

weight = 0.5

numRepeat = 3

fig, ax = plt.subplots()

draw\_squares(ax, numRepeat, center, side, weight)

ax.set\_aspect(1.0)

ax.axis('off')

plt.show()

fig.savefig('Lab1\_1\_b.png')

-1. c)

'''

This program write a recursive method to draw given figures in CS2302 Lab1,

Question 1.

'''

import matplotlib.pyplot as plt

# Create a list of x- and y- coordination based on center and side

def square(center, side):

x = [center[0]-1\*side/2, center[0]-1\*side/2, center[0]+side/2, center[0]+side/2, center[0]-1\*side/2]

y = [center[1]-1\*side/2, center[1]+side/2, center[1]+side/2, center[1]-1\*side/2, center[1]-1\*side/2]

return x, y

'''

draw\_squares draws squares recursively. The first square is drawn based on the given center

and side. The second to fifth squares are drawn at bottom left, top left, top right, and bottom

right cornor respectively. The user can specify how the second to fifth squares change their side

by variable w.

'''

def draw\_squares(ax,n,center,side,w):

if n>0:

# Create and draw a square base on given center and side.

x, y = square(center, side)

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

# Repeat to each corner of the square

draw\_squares(ax, n-1, [center[0]-side/2, center[1]-side/2], side\*w, w)

draw\_squares(ax, n-1, [center[0]-side/2, center[1]+side/2], side\*w, w)

draw\_squares(ax, n-1, [center[0]+side/2, center[1]+side/2], side\*w, w)

draw\_squares(ax, n-1, [center[0]+side/2, center[1]-side/2], side\*w, w)

plt.close("all")

center = [0, 0]

side = 800

weight = 0.5

numRepeat = 4

fig, ax = plt.subplots()

draw\_squares(ax, numRepeat, center, side, weight)

ax.set\_aspect(1.0)

ax.axis('off')

plt.show()

fig.savefig('Lab1\_1\_c.png')

-2. a)

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

if n>0:

x,y = circle(center,radius)

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

center[0]=center[0]-radius\*(1-w)

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

plt.close("all")

fig, ax = plt.subplots()

draw\_circles(ax, 50, [100,0], 100,.5)

ax.set\_aspect(1.0)

ax.axis('off')

plt.show()

fig.savefig('Lab1\_2\_a.png')

-2. b)

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

if n>0:

x,y = circle(center,radius)

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

center[0]=center[0]-radius\*(1-w)

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

plt.close("all")

fig, ax = plt.subplots()

draw\_circles(ax, 50, [100,0], 100,.9)

ax.set\_aspect(1.0)

ax.axis('off')

plt.show()

fig.savefig('Lab1\_2\_b.png')

-2. c)

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

if n>0:

x,y = circle(center,radius)

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

center[0]=center[0]-radius\*(1-w)

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

plt.close("all")

fig, ax = plt.subplots()

draw\_circles(ax, 100, [100,0], 100,.95)

ax.set\_aspect(1.0)

ax.axis('off')

plt.show()

fig.savefig('Lab1\_2\_c.png')

-3. a)

'''

This program write a recursive method to draw given figures in CS2302 Lab1,

Question 3.

'''

import matplotlib.pyplot as plt

'''

Create a list of x- and y- coordination based on number of iteration,

the coordination of parent point, and depth and width, which define the size of

canvas.

'''

def binary\_tree(n, parent, depth, width):

x = [parent[0]-width/4, parent[0], parent[0]+width/4]

y = [parent[1]-depth/n, parent[1], parent[1]-depth/n]

return x, y

'''

draw\_tree draws binary tree recursively. r represents the remaining number of

iteration and n represents the number of iteration in total. parent, depth,

and width are as same as binary\_tree, the coordination of parent point and the

size of drawing canvas.

'''

def draw\_tree(ax, r, n, parent, depth, width):

if r>0:

# Create and draw a binary tree based on given variables.

x, y = binary\_tree(n, parent, depth, width)

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

# Repeat to left and right of the binary tree.

draw\_tree(ax, r-1, n, [parent[0]-width/4, parent[1]-depth/n], depth, width/2)

draw\_tree(ax, r-1, n, [parent[0]+width/4, parent[1]-depth/n], depth, width/2)

plt.close("all")

fig, ax = plt.subplots()

remIterate = 3

numIterate = remIterate

parentCoordinate = [0, 0]

depth = 500

width = 500

draw\_tree(ax, remIterate, numIterate, parentCoordinate, depth, width)

ax.set\_aspect(1.0)

ax.axis('off')

plt.show()

fig.savefig('Lab1\_3\_a.png')

-3. b)

'''

This program write a recursive method to draw given figures in CS2302 Lab1,

Question 3.

'''

import matplotlib.pyplot as plt

'''

Create a list of x- and y- coordination based on number of iteration,

the coordination of parent point, and depth and width, which define the size of

canvas.

'''

def binary\_tree(n, parent, depth, width):

x = [parent[0]-width/4, parent[0], parent[0]+width/4]

y = [parent[1]-depth/n, parent[1], parent[1]-depth/n]

return x, y

'''

draw\_tree draws binary tree recursively. r represents the remaining number of

iteration and n represents the number of iteration in total. parent, depth,

and width are as same as binary\_tree, the coordination of parent point and the

size of drawing canvas.

'''

def draw\_tree(ax, r, n, parent, depth, width):

if r>0:

# Create and draw a binary tree based on given variables.

x, y = binary\_tree(n, parent, depth, width)

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

# Repeat to left and right of the binary tree.

draw\_tree(ax, r-1, n, [parent[0]-width/4, parent[1]-depth/n], depth, width/2)

draw\_tree(ax, r-1, n, [parent[0]+width/4, parent[1]-depth/n], depth, width/2)

plt.close("all")

fig, ax = plt.subplots()

remIterate = 4

numIterate = remIterate

parentCoordinate = [0, 0]

depth = 500

width = 500

draw\_tree(ax, remIterate, numIterate, parentCoordinate, depth, width)

ax.set\_aspect(1.0)

ax.axis('off')

plt.show()

fig.savefig('Lab1\_3\_b.png')

-3. c)

'''

This program write a recursive method to draw given figures in CS2302 Lab1,

Question 3.

'''

import matplotlib.pyplot as plt

'''

Create a list of x- and y- coordination based on number of iteration,

the coordination of parent point, and depth and width, which define the size of

canvas.

'''

def binary\_tree(n, parent, depth, width):

x = [parent[0]-width/4, parent[0], parent[0]+width/4]

y = [parent[1]-depth/n, parent[1], parent[1]-depth/n]

return x, y

'''

draw\_tree draws binary tree recursively. r represents the remaining number of

iteration and n represents the number of iteration in total. parent, depth,

and width are as same as binary\_tree, the coordination of parent point and the

size of drawing canvas.

'''

def draw\_tree(ax, r, n, parent, depth, width):

if r>0:

# Create and draw a binary tree based on given variables.

x, y = binary\_tree(n, parent, depth, width)

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

# Repeat to left and right of the binary tree.

draw\_tree(ax, r-1, n, [parent[0]-width/4, parent[1]-depth/n], depth, width/2)

draw\_tree(ax, r-1, n, [parent[0]+width/4, parent[1]-depth/n], depth, width/2)

plt.close("all")

fig, ax = plt.subplots()

remIterate = 7

numIterate = remIterate

parentCoordinate = [0, 0]

depth = 500

width = 500

draw\_tree(ax, remIterate, numIterate, parentCoordinate, depth, width)

ax.set\_aspect(1.0)

ax.axis('off')

plt.show()

fig.savefig('Lab1\_3\_c.png')

-4. a)

'''

This program write a recursive method to draw given figures in CS2302 Lab1,

Question 4.

'''

import matplotlib.pyplot as plt

import numpy as np

import math

# Create a list of x- and y- coordination based on center and 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

'''

draw\_circles draws circles recursively. The circle is drawn based on given

center and radius. Additional five circles are drawn inside the first cirle

in order of center, left, right, bottom, and top respectively.

'''

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

if n>0:

# Create and draw a circle based on given center and siradius.

x,y = circle(center,radius)

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

# Repeat to each inside position of the circle

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

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

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

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

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

plt.close("all")

fig, ax = plt.subplots()

numIterate = 3

center = [0, 0]

radius = 100

draw\_circles(ax, numIterate, center, radius)

ax.set\_aspect(1.0)

ax.axis('off')

plt.show()

fig.savefig('Lab1\_4\_a.png')

-4. b)

'''

This program write a recursive method to draw given figures in CS2302 Lab1,

Question 4.

'''

import matplotlib.pyplot as plt

import numpy as np

import math

# Create a list of x- and y- coordination based on center and 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

'''

draw\_circles draws circles recursively. The circle is drawn based on given

center and radius. Additional five circles are drawn inside the first cirle

in order of center, left, right, bottom, and top respectively.

'''

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

if n>0:

# Create and draw a circle based on given center and siradius.

x,y = circle(center,radius)

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

# Repeat to each inside position of the circle

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

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

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

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

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

plt.close("all")

fig, ax = plt.subplots()

numIterate = 4

center = [0, 0]

radius = 100

draw\_circles(ax, numIterate, center, radius)

ax.set\_aspect(1.0)

ax.axis('off')

plt.show()

fig.savefig('Lab1\_4\_b.png')

-4. c)

'''

This program write a recursive method to draw given figures in CS2302 Lab1,

Question 4.

'''

import matplotlib.pyplot as plt

import numpy as np

import math

# Create a list of x- and y- coordination based on center and 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

'''

draw\_circles draws circles recursively. The circle is drawn based on given

center and radius. Additional five circles are drawn inside the first cirle

in order of center, left, right, bottom, and top respectively.

'''

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

if n>0:

# Create and draw a circle based on given center and siradius.

x,y = circle(center,radius)

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

# Repeat to each inside position of the circle

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

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

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

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

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

plt.close("all")

fig, ax = plt.subplots()

numIterate = 5

center = [0, 0]

radius = 100

draw\_circles(ax, numIterate, center, radius)

ax.set\_aspect(1.0)

ax.axis('off')

plt.show()

fig.savefig('Lab1\_4\_c.png')