University of Utah
School of Computing

CS 1060 Homework #2 Spring 2016

Due Feb 23, 2016 at the start of class

Question 1 (30 points). The SnowFlake Example (please see next page).

Question 2 (30 points, each worth 10 points). The Planet Art Example (see pages 3 - 7). The Python code to be modified is also included as a separate file: PlanetFractalArt.txt.

Please contact TAs Ross and Williams for questions on this assignment.

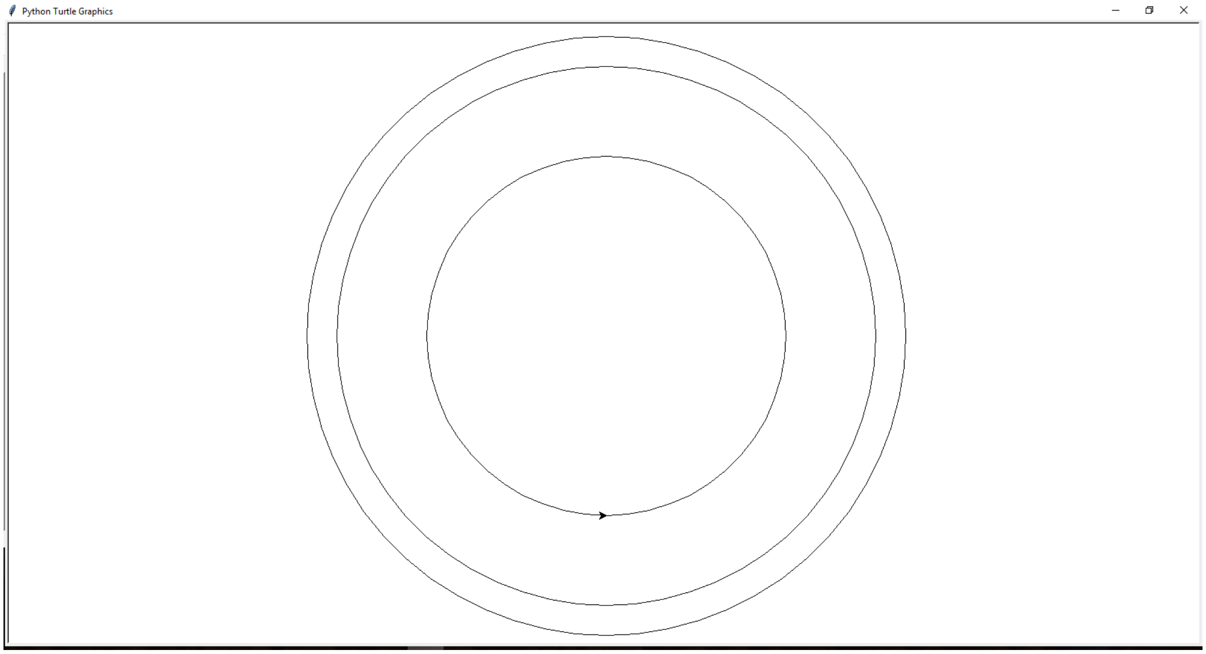
```python
# CS1060 - A2
# Recursion and Fractals

# Copy the following code snippit into http://trinket.io

from turtle import *

def drawSnowFlake(length, depth):
    if depth > 0:
        for i in range(6):
            forward(length)
            drawSnowFlake(length // 3, depth - 1)
            backward(length)
            left(72)

# set turtle_speed equal to 0 to render the fractal as fast as possible.
turtle_speed = 5
speed(turtle_speed)

drawSnowFlake(72,2)

# Answer the following questions by analyzing and manipulating the code snippet

# Question 1 (10 points)
#   1.1 (5 points): The function drawSnowFlake(length, depth) takes in 2
#       arguments: length and depth. Describe in words how these arguments
#       effect the drawn fractal.
#
#   1.2 (5 points): Describe in words what makes this algorithm recursive, what
#       determines the magnitude of recursion, and how the fractal changes when the
#       magnitude of recursion grows.
#
# Question 2 (10 points)
#   Change the "6" in "for i in range(6):" to a number other than 6, and the
#   "72" in "left(72)" into "360/number" where number is the same number specified
#   in range().
#
#   2.1 (10 points): Knowing that 360 is the number of degrees in a circle, explain
#       what drawing a pattern that number of times and turning 360/number degrees
#       each time accomplishes.
#
# Question 3 (10 points)
#   Changing a number in two different places can quickly become tedious. A more
#   convenient way to change this number in two places is to use a variable.
#
#   3.1 (10 points): Rewrite the function drawSnowFlake to have a function
#       signature of drawSnowFlake(length, depth, leaves) and use the added
#       argument leaves instead of a number similar to question 2. Also show how
#       you would call your modified drawSnowFlake function.
```

Planet Fractal Art Description

For this assignment you will try to recreate fractal art images of planets that have a ring. All of the code is given just copy and paste the code into a python interpreter and it should run as it is. If you do not have an interpreter installed on your computer, it is installed on the lab computers in the WEB building, or you can use the following website.

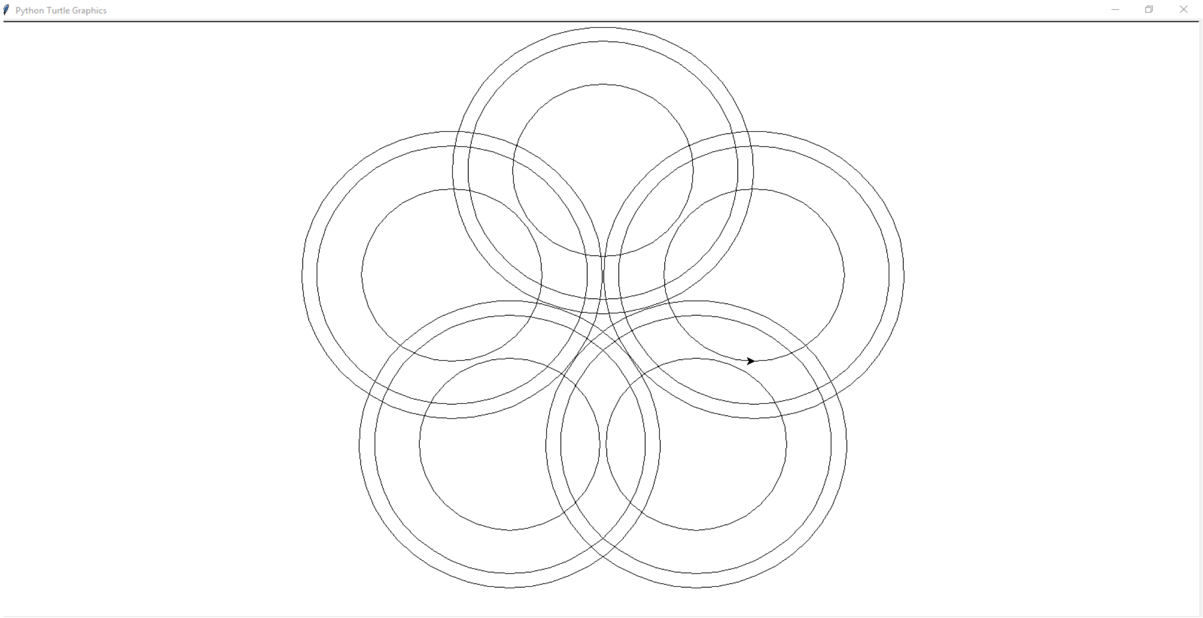
https://trinket.io/python

If you do not know what one planet looks like, then here is an image of just one, all by itself.



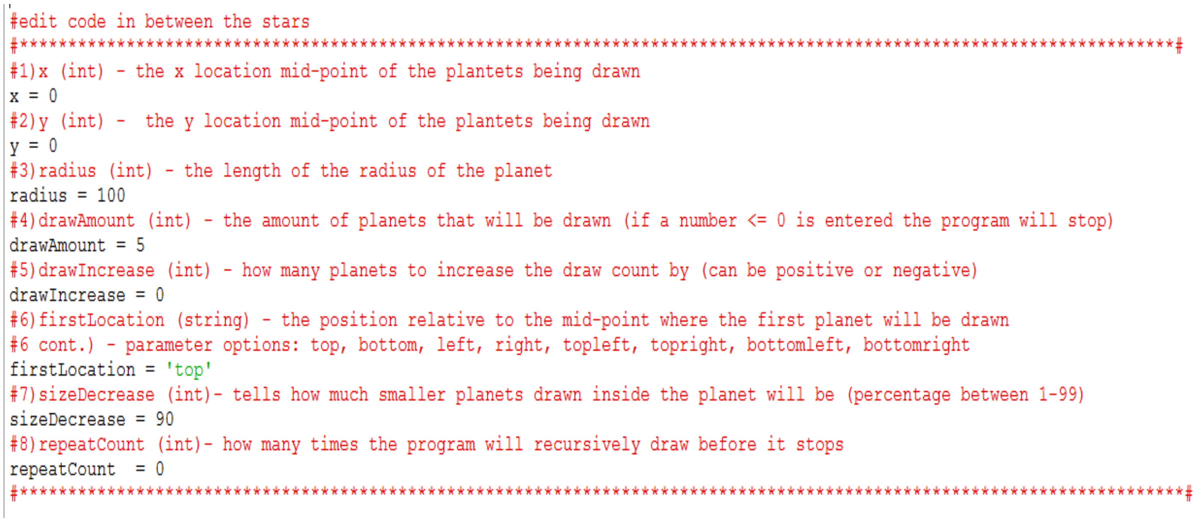
Remember the planet has a ring going around it, so each planet consists of 3 circles.

When you copy and paste the code into an interpreter and run it the program should



create the following image. If you cannot tell it is a picture of five planets.

The code that you will be modifying has been marked and is toward the bottom of the page. Here is an image of the section of code you will be modifying. Please be aware that your



```
#edit code in between the stars
#***********************************************************************************************************************#
#1)x (int) - the x location mid-point of the plantets being drawn
x = 0
#2)y (int) -  the y location mid-point of the plantets being drawn
y = 0
#3)radius (int) - the length of the radius of the planet
radius = 100
#4)drawAmount (int) - the amount of planets that will be drawn (if a number <= 0 is entered the program will stop)
drawAmount = 5
#5)drawIncrease (int) - how many planets to increase the draw count by (can be positive or negative)
drawIncrease = 0
#6)firstLocation (string) - the position relative to the mid-point where the first planet will be drawn
#6 cont.) - parameter options: top, bottom, left, right, topleft, topright, bottomleft, bottomright
firstLocation = 'top'
#7)sizeDecrease (int)- tells how much smaller planets drawn inside the planet will be (percentage between 1-99)
sizeDecrease = 90
#8)repeatCount (int)- how many times the program will recursively draw before it stops
repeatCount  = 0
#************************************************************************************************************************#
```

interpreter might not make your words the same color as the words in my image.

For this assignment you will only need to modify three variables, which are drawAmount, drawIncrease, and repeatCount. Please leave all other variables alone. However if you have a big monitor and want to increase the size of the image feel free to increase the radius variable and the whole picture will proportionally increase with it. Once you have completed the assignment feel free to mess with the other variables as you please.

answers should be given in the following format:

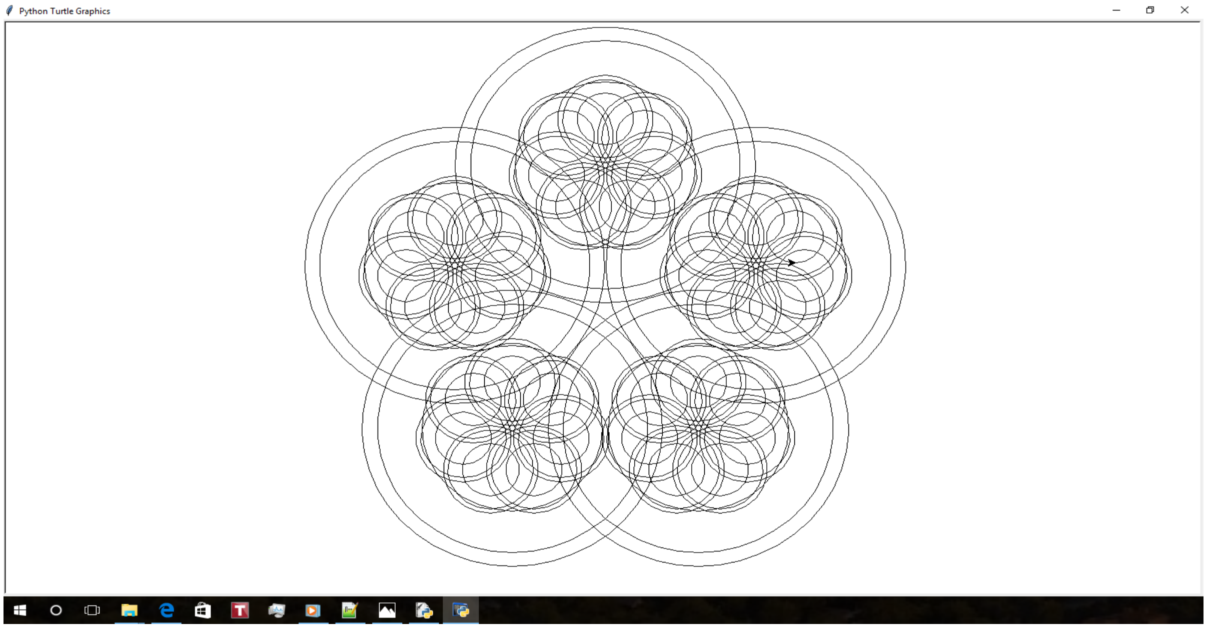
drawAmount = 'place answers here'

drawIncrease = 'place answers here'

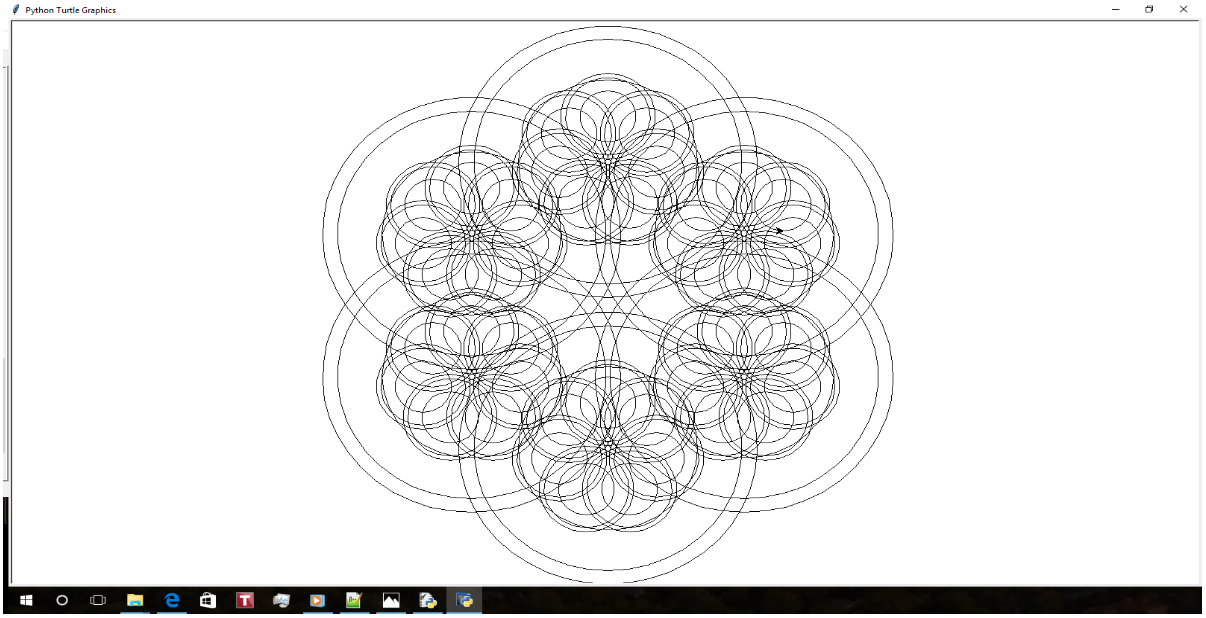
repeatCount = 'place answers here'

Questions

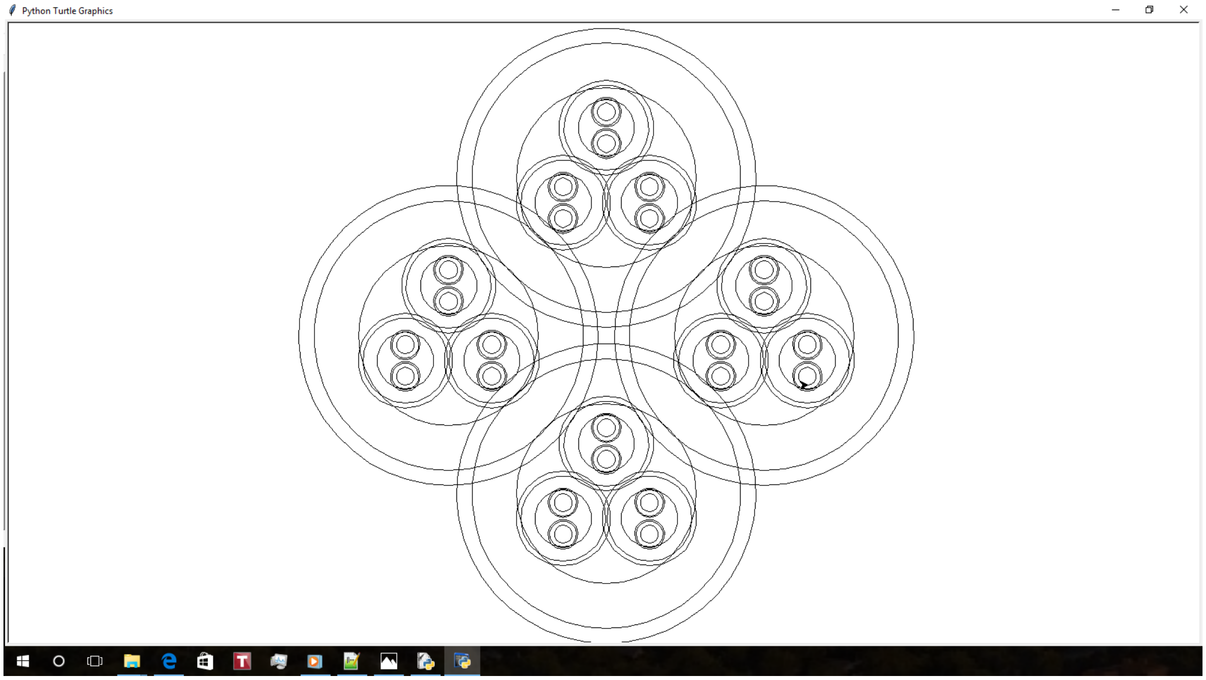
1. Currently the code is set up to draw five planets. Modify the code so that seven planets a inside each planet. The picture should look like this. (10 points)



2. After getting that picture you decide that you would rather have six big planets, and you still want seven on the inside. Modify the code so that there are six planets on the outside and still seven on the inside. The picture should look like this. (10 points)



3. If you have completed the first two questions you should have a pretty good idea of how the three parameters works. Now modify the code to create the following picture.


Python Turtle Graphics