CSC 106 Assignment 4: A taste of Objects, Recursion, and Fractals

Due: Tue, Nov 18, 2014 at 11:30 pm

Marks: 20 marks.

Learning Goals:
At the end of this assignment you will be able to:

- Fully document a program written in Python
- Use recursion to generate a fractal diagram
- Use the Turtle package to create a graphic design

Submission: This assignment may be done in pairs. You may also work individually if you prefer. The submitted assignment will have two components:
One component will be a **Python file (.py) file called UpsideDown.py**
The other component will be a **Python file (.py) called MyGraphic.py**
Both Python files must be in one of the Version 2.7 "flavours" of Python (e.g. 2.7.8, 2.7.6, etc.)

All files should be attached to the CSC106 Assignment 4 submission page on Connex.

If you are working as a pair, only ONE member of the pair submits the assignment.

A few reminders:
---- Make sure that both partner's NAMES are in the Assignment Text box on Connex. As well, please make sure that the names of the partners are in a comment at the top of the code in each Python program handed in.

- MOST IMPORTANT: ONLY ONE ASSIGNMENT SUBMITTED PER TEAM! –

Resources:

- **You DO NOT need to install or use Python on your own computer -- every machine in the computer science building has Python on it.**
- The main Python resource: https://www.python.org/about/gettingstarted/

Special resources for Mac Users:

- Python 2.7 is already pre-installed on Mac OS X (Mavericks) machines. Although I haven't tested our assignment on an OS X machine, we don't think you'll need anything more than what comes pre-installed. Here is a quick video showing you

how to access and use Python on a Mac (OS X)
https://www.youtube.com/watch?v=BE1wDsLzOJA

- An excellent text editor for Mac Users – check out this great video on how to use TextWrangler for Mac with Python:
https://www.youtube.com/watch?v=et2vjUAz9-k
- If you want more bells and whistles for Python on your Mac this web page give you some instruction for installing Python on your Mac – step by step (caution though – this can get VERY complex and we can't help you out if things go south):
http://www.pyladies.com/blog/Get-Your-Mac-Ready-for-Python-Programming/

Special resources for Windows Users:

- Windows doesn't come with Python installed. If you want to use Python on your own Windows 7 or Windows 8.x machine, you need to install it. It is relatively straightforward, but, again – if something goes wrong, we can't help you out. Install info for Windows 7 (you do not need Mechanize, Requests, Beautiful Soup, or CSV components – no need to download and install):
http://www.anthonydebarros.com/2011/10/15/setting-up-python-in-windows-7/
Install info for Windows 8.1(you do not need Mechanize, Requests, Beautiful Soup, or CSV components talked about towards the end of the instructions, so don't bother downloading and installing them.):
http://www.anthonydebarros.com/2014/02/16/setting-up-python-in-windows-8-1/
- TextPad text editor for use with Windows:
https://www.textpad.com/download/index.html
- NotePad++ http://notepad-plus-plus.org/
- Ninja IDE: http://ninja-ide.org/

Turtle overview

- Sadly, this assignment won't work for online editors like repl.it, so you'll have to use either Python on your own computer, or Python on one of the machines in the CSC labs. Python 2.7.8 is installed in our lab classroom (242), but it is also on all of the other machines on 2nd and 3rd floor. You can drop in to ECS 342 if you want to use a Linux machine, or to any of the 2nd floor Windows labs where Python is on all of the Windows machines. You can use any of the labs on the 2nd floor of ECS if there is not a lab in session.
- We will use the turtle module for drawing graphics in Python. There are lots of excellent references for this module – indeed, you'll recognize many of its actions from things we did in Scratch. Here are some references you might find helpful:
- https://fiftyexamples.readthedocs.org/en/latest/turtle.html (This is Python 3, but, other than the input statement at the bottom of the sample code, works perfectly well in 2.7.x).
- http://interactivepython.org/runestone/static/thinkcspy/toc.html An EXCELLENT book on thinking like a computer scientist – check out the chapter on Python Turtle Graphics
- https://docs.python.org/2/library/turtle.html The official overview of everything "turtle" in the Python environment.

TASK 1: Turn **Sierpinski's Triangle** on its head (15 marks)

Sierpinski Triangle.

1. The following web page has code and explanation for generating Sierpinski's Triangle (a fractal) in Python: http://interactivepython.org/runestone/static/pythonds/Recursion/graphical.html

   Your goal is to make a copy of this code with the following changes:
   A). Use different colours (check out http://www.discoveryplayground.com/computer-programming-for-kids/rgb-colors/)
   B). Make the Sierpinski Triangle twice as big as the original code renders it.
   C). Turn the Sierpinski Triangle upside down (make it draw with the point downwards).
   D). Comment the code. You'll notice that there are no comments in the code. Your goal here is to comment the code to correctly describe what is happening in the following functions and statements (you can do more if you like, but we are marking specifically on these):
   a. import turtle
   b. def drawTriangle(points,color,myTurtle):
   c. myTurtle.fillcolor(color)
   d. myTurtle.up() and myTurtle.down()
   e. myTurtle.goto(points[0][0],points[0][1])
   f. def getMid(p1,p2):
   g. def sierpinski(points,degree,myTurtle):
   h. colormap = ['blue','red','green','white','yellow','violet','orange']
   i. drawTriangle(points,colormap[degree],myTurtle)
   j. if degree > 0:
   k. any one of the calls to sierpinski
   l. myPoints = [[-100,-50],[0,100],[100,-50]]
   m. sierpinski(myPoints,3,myTurtle)

TASK 1 DELIVERABLES: A .py file named UpsideDown.py

Marks will be assigned as follows:
A). 2 marks for different colours

B). 3 marks for making the triangle twice as large

C). 5 marks for turning the triangle on its head (making it draw with the pointy end down)

D). 5 marks for comments. Marks for D break down as follows:
5/5 fully and correctly described with complete clarity – exemplary work – absolutely no question in the mind of the marker you know what is happening at that statement or function. This does not mean writing huge amounts of text for each line of code – these lines of code can be correctly described in not more than one or two sentences.

4.5 / 5 fully and correctly described – excellent work, but with one or two points that are not quite as clear as they could be.
4 / 5 fully and correctly described – very good work, but with more than two points that are not quite a clear as they could be.

3.5 / 5 approximately ¾ of the code is correctly described.

3 / 5 approximately ½ of the code is correctly described.

1 to 2 / 5 approximately ¼ of the code is correctly described.

0 / 5 less than ¼ of the code is correctly described.

(This Task is worth 15 marks).

TASK 2: Create your own graphic (5 marks)

1. Create your own graphic in Python using the Turtle graphics module. You can do whatever you like – perhaps you'll create a forest of recursive trees in different colours, or maybe you'll implement another fractal, or maybe create an abstract image – it is your choice. Your python code will be well documented and will include references (same method as you used in Assignment 3) as to where you got your idea(s) / help for this Task.

If you took major components of code from elsewhere, you will need to reference where you got this code from, and, for a top mark, you'll need to make changes to the code (or do something extra with it) to present it as your own graphic. In comments, you will note what these changes are.

TASK 2 DELIVERABLES: A .py file named MyGraphic.py

Marks will be assigned as follows (yes, there is a bit of subjectivity here – the markers and I will be marking for WOW factor):
5/5 for exemplary work – including exemplary comments.

4.5 / 5 for excellent work – including excellent comments.

4 / 5 for very good work – including clear and complete comments.

3.5 / 5 for good work – including comments that correctly document approximately ¾ of the code.

3 / 5 for satisfactory work – including comments that correctly document approximately ½ of the code.

1 to 2.5 / 5 for giving this component a try – could also be for code that doesn't run, but is otherwise correct.

(This Task is worth 5 marks).