CMPT 120—LAB WEEK 4 (Jan. 27)

In Lectures 7 and 8, you were introduced to the turtle module in Python which allows for graphical drawing. You were given a few examples of how to use the turtle, including some very complicated example, the Koch snowflake, that will ***not*** *be investigated here.*

This lab will aim to use the turtle to do some drawings.

GOALS OF THIS LAB:

- Get comfortable using a Python module.
- Understand how to write a small program from instructions.
-

General instructions:

- It is recommended you do these exercises in teams of 2 people each. Make sure to take a partner that has roughly the same experience and understand of the topic as you. (If you team up with someone who comprehends the material much better than you, then chances are you will not learn anything.)

- Save your work for future reference and to show the TAs and instructor, so as to show you have actively participated in the lab.

- **Save your files in your home folder U:** (files saved on the Desktop may get lost). Python files should have the **.py** extension written explicitly—the *extension* is the end of filename.

Submission of this lab:

- This week, submission will be done through Coursys, which is available https://courses.cs.sfu.ca/

- You are asked to submit one Python source code file per exercise, containing everything you have done in that exercise. If your code does not work, you may comment (use the # at the beginning of the line) and try to explain what you were attempting to do.

- **FINAL QUESTION: you must also submit a screenshot of any one of your drawings which you are the proudest about. If you do not know how to take a screenshot, look it up on Internet, for instance here:** http://www.wikihow.com/Take-a-Screenshot-in-Microsoft-Windows

—PART 1: EXPLORATION—

Exercises in this first part encourage you to explore the behavior of Python.

EXERCISE 1: discovering the functionalities of the turtle module
The purpose of this exercise is for you to familiarize yourself with Python's self-document system.

Python modules have a high quality official documentation, which may be accessed one of several ways (all of which will give the same information).

- The first way is through the Python website, by going to the Python documentation website (docs.python.org) and ***making sure you are getting the documentation for your version of Python, 2.7.6,*** you end up at this link:

  http://docs.python.org/2.7/

  Type "turtle" in the Quick Search box (on the left side bar), and search. It will provide all functions linked to the turtle module, and the very first page in the results is the turtle module's complete documentation:

  http://docs.python.org/2.7/library/turtle.html

  There is an introduction that provides you with a lot of overwhelming information, but if you scroll down, you will see the "Turtle methods", which will list all methods you can use to control the turtle.

- The other way of getting documentation from Python is through the interactive Python shell. Once you have imported the module, you may then ask for help:

```
>>> import turtle as t
>>> help(t)
```

Listing 1: importing the turtle module and getting documentation

Note that if you import turtle using a short name (as discussed in Lecture 8), you should ask for help on the short name:

```
>>> import turtle as t
>>> help(t)
```

Listing 2: importing the turtle module using a short name

You may also ask for help on a specific function:

```
>>> import turtle
>>> help(turtle.forward)
Help on function forward in module turtle:

forward(distance)
    Move the turtle forward by the specified distance.

    Aliases: forward | fd

    Argument:
    distance -- a number (integer or float)

    Move the turtle forward by the specified distance,
    in the direction the turtle is headed.

    Example:
    >>> position()
    (0.00, 0.00)
    >>> forward(25)
    >>> position()
    (25.00,0.00)
    >>> forward(-75)
    >>> position()
    (-50.00,0.00)
```

Listing 3: asking for documentation on a function in the Python shell

So you have means of finding out how to use the functionalities of the turtle that we have not covered in class (changing colors, filling regions, etc.). For a quick list of functions, you may also look at the slides of Lecture 8.

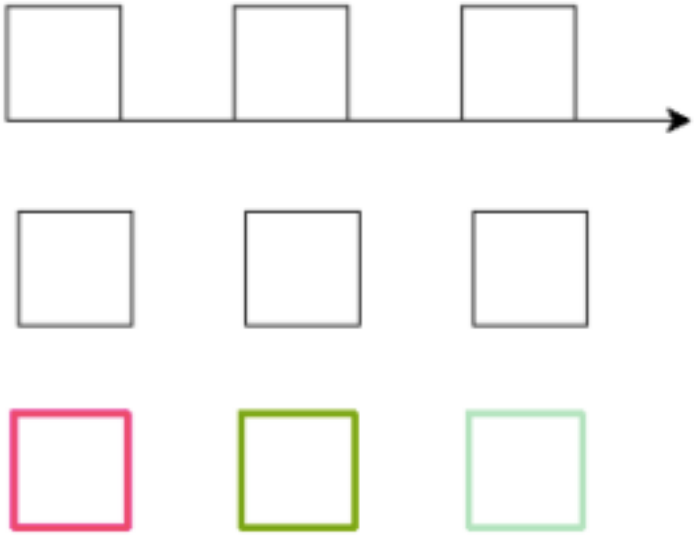
Try the following:

1. Redraw the examples seen during Lecture 7.

2. Try using colors in those examples (there are several options to refer to colors, and you may for instance, for convenience, use string descriptions such as “blue”, “red”, etc.).

3. Investigate the notion of “pen up” and “pen down” to be able to draw non-continuous drawings.

4. Write programs to draw the following (you may consider that the whole thing is one same drawing, or you may consider that each row is a different program/drawing):

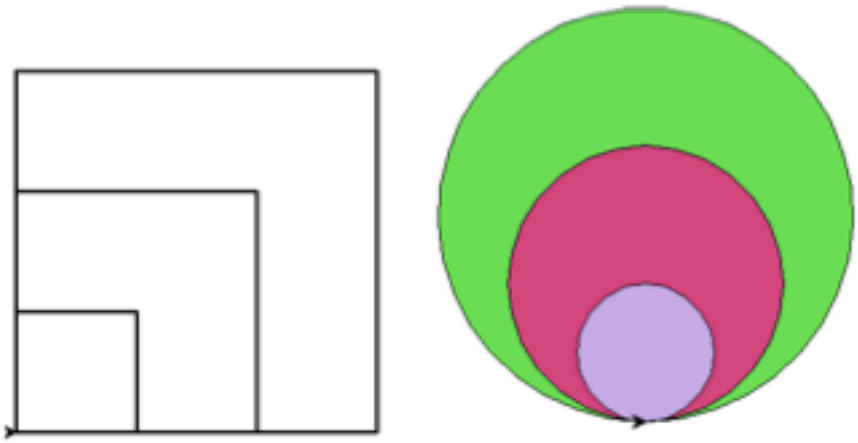


EXERCISE 2: drawing using loops, part 1
In this example, you should explore on finding ways of drawing using loops.

In this exercise, you are simply asked to use:

- a function that draws a shape (either one that you have defined, or one that already exists in the turtle module)
- use a loop and draw objects of different size.

For instance, you can take inspiration on the following (do whatever you want with the colors):



—PART 2: DESIGN AND PROGRAMMING—

Exercises in this second part encourage you to write your own code from scratch.

Notes and recommendations for all exercises of this type (in this lab, and in your future work):

- **To test and debug,** include enough printing messages so that you may see intermediate and final results and also from where the printing is done.

- Alternatively if your program is running fine (that is, there is no syntax error), but rather your problem is that your problem is not computing the result you intend it to, you can also use the Python Tutor website to see step by step what is happening.

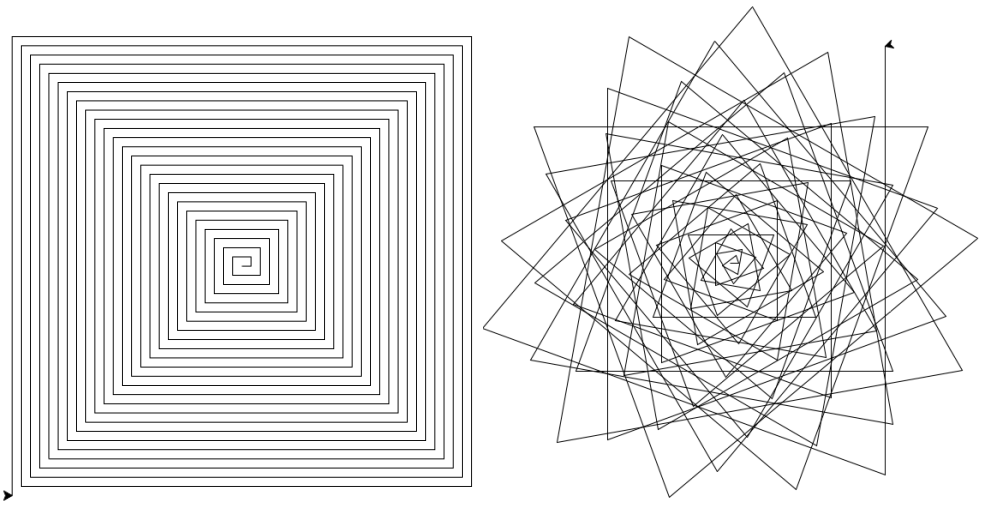
- Always **consider different cases and special limit cases**: when your parameters have values that require special case.

- Make sure to initialize all your variables before using them.

- Don't forget you can put comments in your code, by putting a sharp # at the beginning of the line.

- In a program, do not write sentences that are so long that you cannot see them in the IDLE editor original screen.

EXERCISE 3: spirals
The purpose of the exercise is for you to define a function and use a "for" loop in the context of a turtle drawing.



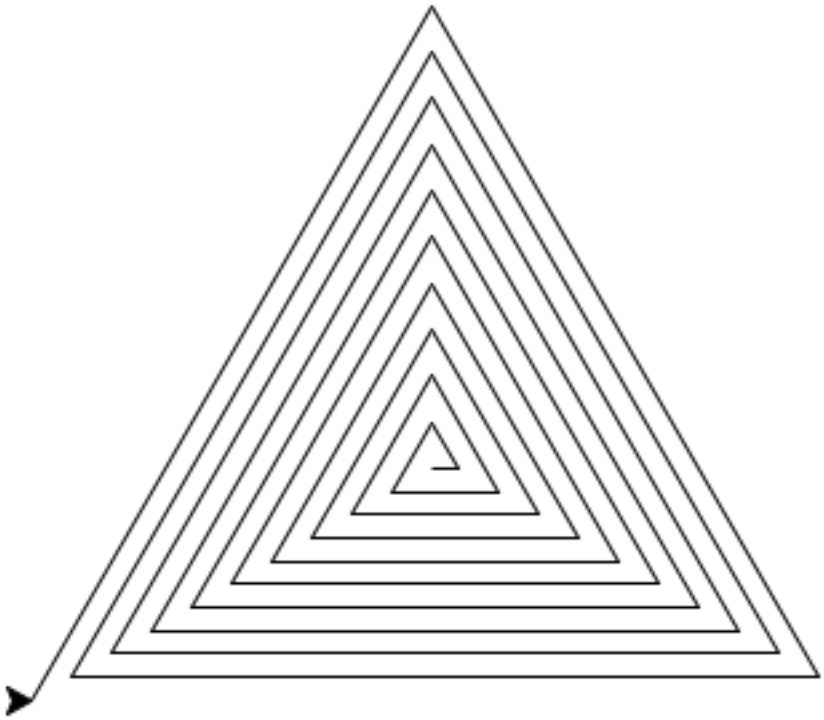
1. Write a function “draw_square_spiral” that takes as a parameter the number of lines in the spiral (for instance, the one on the left drawing has 100 lines) and draws a square spiral accordingly.

(The challenge in all of these questions is how you modify the length of the line as the iterations go on.)

2. Write other functions by modifying some of the parameters, such as the angle the turtle turns, call this function “draw_spiral”, and have it take two parameters, one for the number of lines, and the other for the angle at which it should turn.

IF YOU HAVE TIME (advanced):

3. Triangle spirals are a bit more difficult. You can first try to do an “equilateral triangle spiral”, in which all angles are the same.



Once that is done, you may try to do the harder task of doing non-equilateral triangles (triangles in which all angles aren’t the same).

4. Finally, you might try to do spirals of more complex forms (like stars…).