INTRODUCTION TO COMPUTER SCIENCE I
LAB 8
Turtles and pretty patterns

We're going to use *Swampy*, a collection of new Python functions, to create a basic graphical interface. Doing so, we'll create a few objects and use their methods.

1 Functions to draw patterns

To get started, try the following:

1. Begin, as usual, by opening *IDLE 3*.
2. Create a new module, and save it as `patterns.py`. You should, wherever you store your work, create a new folder named `lab-8`, and place `patterns.py` within that folder.
3. Click on the following link to download a zip-file of the *Swampy* code:

   ```
   https://app.cs.amherst.edu/sfkaplan/courses/2014/spring/
   COSC-111/swampy-2.1.zip
   ```
4. Unzip this file, producing a folder named `swampy`.
5. Move the new `swampy` folder into your `lab-8` folder.
6. Go back to your `patterns.py` module window. Within this module, you should write the following:
   - On the first line, write:

     ```
     from swampy.TurtleWorld import *
     ```

     This magic incantation makes all of the *Swampy* functions available to us.
   - On the bottom lines (where it should stay), write:

     ```
     if __name__ == '__main__':
       main()
     ```

     This is our usual magical incantation to cause Python, upon the loading and running of your module, to start the program by calling the function `main()`.

- Between your `import` statement at the top, and the `if` statement to call `main()` on the bottom, start writing your `main()` function that accepts no parameters. This function should create the special *objects* for *Swampy*. Specifically:

  ```
  def main ():
    world = TurtleWorld()
    turtle = Turtle()
    wait_for_user()
  ```

  If you were to run the program now, this simple `main()` would create a new window with a "turtle" (a critter that moves and draws on the screen) in the middle. The program will then wait for you to close that window; when you do, the program ends. Later, we will add more things for the turtle to do before the `wait_for_user()` function is called.

- Write a `polygon()` function (above the `main()` function). Specifically, it should look something like:

  ```
  def polygon (turtle, sides, size):
  ```

  This function should use the `turtle` to draw a polygon with the given number of `sides`, having the turtle draw each side of the polygon such that it has the given `size`.

- Write a `spiral_polygon()` function, like so:

  ```
  def nested_polygon (turtle, sides, size, decrement):
  ```

  This function should draw one line segment in the shape of the a polygon of the given number of `sides`, and where the `size` of each side of the polygon decreases by `decrement` until the size reaches zero and the `turtle` is in the middle of the spiral.

- **Extra challenge:** Write a `nested_polygon()` function. It should look something like:

  ```
  def nested_polygon (turtle, sides, size, decrement):
  ```

  This function should use the `turtle` to draw a series of polygons with the given number of `sides`. The first polygon should have sides of the given `size`, the next should have sides whose sizes are `size - decrement`, the one after that should have size `size - (2 * decrement)`, and so on, until the sides would have zero size. The polygons must be concentric.[1]

[1] This last part—making them concentric—is tricky! If you can make the polygons *concentrickish*, that's good. If you really want to do it right, figure out just how to move the turtle correctly between the nested polygons such that they are truly centered. There are ways to solve this more difficult version of the problem both with and without trigonometry.

Suggestions. For each of the three drawing functions, write it, test it (by calling it from `main()`, and be sure that it works **before** you move on to the next drawing function.

Once you have all of the functions working, you must change `main()` to do the following:

1. Print the following menu of options to the user:

   ```
       (0) Exit
       (1) Polygon
       (2) Nested polygon
       (3) Spiral polygon
       Your choice:
   ```

2. Ensure that the user enters one of the menu numbers.[2]

3. If the user chooses to exit, then end the program. Otherwise, prompt the user for the necessary information for the drawing (the number of sides, the size, and, if relevant, the decrement).

4. Draw the requested shape.

5. Bring the user back to the menu above, given them a chance to draw a new shape.

2 Submitting your work

As always, go to the course submission page, and submit your completed `patterns.py` for Lab 8

This assignment is due by Wednesday, Apr-02, at 11:59 pm.

[2]Don't worry about the user being silly and entering a non-integer that crashes the program.