Computer Science I — CSCI-141
Hexagons — Homework 1

08/17/2021

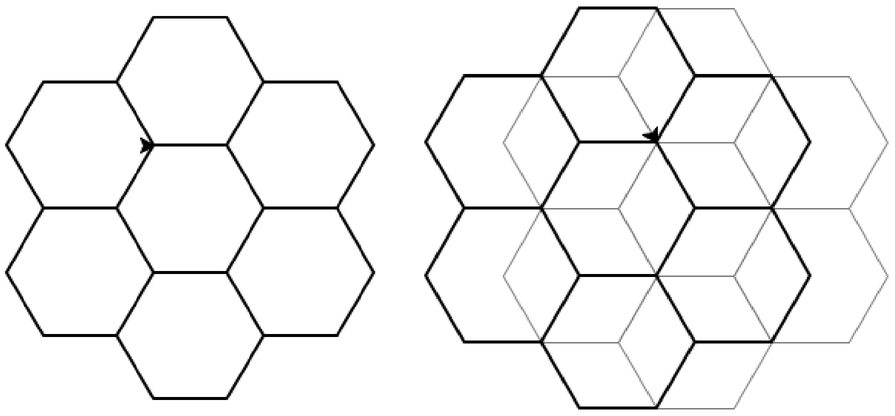
1 Problem

Write functions and compose a program that draws the second figure shown below. To do this the program will need to execute the same sequence of operations twice. The sequence is to draw one small hexagon, have the turtle move the length of the first side it drew, turn right 60 degrees, and then repeat these steps five more times.

Doing the above creates the drawing shown on the left below. Before the program executes the sequence a second time, have the turtle turn to the right by 60 degrees. Then re-executing the same function will create the drawing on the right.

All the lines that the program actually draws will be identical! The drawing from the first phase is dimmed to gray in appearance to highlight the new drawing.

When drawing operations have finished, the program must **pause** until the user closes the canvas window.



2 Tips

- Note that the center hexagon is not explicitly drawn; it is formed by the drawing of the outer hexagons.
- In the interest of continuous drawing, the program is allowed to make the turtle redraw lines it has already drawn; this is not real ink. See a sample animation movie at:
  `https://www.cs.rit.edu/~csci141/Homeworks/01/execution.mov`
- To make the turtle wait until the user closes the canvas window, use the turtle module function `done()`.

- For code-writing style rules, see the guide posted in the **Resources** section of the course web site are followed. Here is the link to the resources:
  `https://www.cs.rit.edu/~csci141/resources.html`

3 Specifications

3.1 Requirements

1. The program file name must be **hexagons.py**, and case is sensitive.
2. The implementation should contain a function to draw a single hexagon, one or more functions to set up the canvas and turtle state, a main function, and a function to reposition the turtle to the next place to draw.
3. The program begins by initializing the drawing canvas and turtle state.
4. The pen "size" (width of lines) shall be 2, and all sides drawn shall be 60 pixels in length.
5. Drawing begins with the turtle facing East.
   The sequence is to draw one small hexagon, have the turtle move the length of the first side it drew, turn right 60 degrees, and then repeat these steps five more times. Before the program executes the sequence a second time, have the turtle turn to the right by 60 degrees. Re-executing the same function will create the final drawing on the right.
6. The first line drawn is the bottom edge of the top hexagon in the first set of six hexagons.
7. The call `turtle.done()` should be the last statement the program executes. Before that line, there must be a `print` function call to tell the user how to close the drawing window (i.e. click on the "X" button in the window titlebar).

3.2 Constraints

Constraints are requirements that the program **implementation must not do**.

Programming constructs not yet presented in this course are prohibited, and here are some examples:

- The functions defined in the module may not have parameters.
- Using repetition constructs (recursion or loops) is prohibited.
- Using variables and assignment statements is prohibited.

The course will introduce these and other practices in the coming weeks.

3.3 Program Invocation and Operation

If using `Idle`, run the program to produce the picture like this:

Run → Run Module

If using PyCharm, right-click on the program source and choose this:

Run hexagons.py.

After running it once, the program's name should be visible near the top of the tool; just click the adjacent green triangle.

The program also should be runnable from the command line of a terminal (or console) window as shown here (how a grader might run the program):

```
python3 hexagons.py
```

When run, the program must pause after drawing the figure and wait for the user to close the canvas window before it terminates.

3.4 Grading

- 25%: Output Functionality

  The implementation produces the expected picture, and the figure fits within the canvas.

- 60%: Design
  - 10%: Multiple user-defined functions work together to create the solution.
  - 30%: The implementation reuses at least two user-defined functions to enable reuse and minimize repeated code. Reuse happens when the code calls the same function more than once.
  - 10%: At least one function sets up the initial canvas and turtle state.
  - 10%: A "main" function exists to drive the overall program operation.

  A 50% penalty will be levied if the code solution uses a single sequence of commands without defining functions or reusing some of them.

- 10%: Documentation and Style
  The file module has a *docstring* at the top identifying the file's name and full, first and last author name.
  Each function has a *docstring* containing a sentence describing its purpose. This documentation helps others understand how they may reuse the function. Below is an example of function docstring content.

```
def draw_triangle():
    """
    Make the turtle draw a triangle 100 pixels on each side.
    :precondition: Pen is down.
    :precondition: Turtle is facing the direction in which the first
                   side will be drawn.
    :postcondition: Turtle's state (location, direction, pen position)
                    is the same as when this function started.
    """
    # ... the body of function would go here ...
```

The entire program file should be written in a correct, standard style, which starts with a *file docstring* describing what the program does. The file docstring must also contain the *full*, first and last name of the author. Study and apply the style-related content in the course resources

    `https://www.cs.rit.edu/~csci141/resources.html`

- 5%: Submission Conformance

  The program file name must be `hexagons.py`, and the program file must be submitted inside a zip file named `hw01.zip`. See the Submission section for details.

3.5 Submission

To compress the file into a **zip** file, *zip* the documented program code into a file called **hw01.zip**

Note: It may seem pointless to zip a single file, but this requirement is for consistent grading. Some assignments will contain more than one file, and it is best that they be zipped together.

Other compression formats, e.g., **7zip**, **rar**, and **tgz** ("tar-gzip") are prohibited.

Submit the zip file by uploading to the Homework 1 assignment drop box in MyCourses. Below are some ways to make a zip file.

- On a Mac: in the Finder window, select the file, right-click, and choose:
  Compress hexagons.py.
- On Windows: in a File Manager window, select the file, right-click, and choose:
  Send to → Compressed (zipped) folder.
- On Linux (and on a Mac): in a terminal window, type: `zip hw01.zip hexagons.py`

3.5.1 Submission Upload Steps

1. Login to MyCourses and go to the course page.
2. Navigate to the Assignments (Dropbox) area.
3. Locate the correct assignment folder.
4. Upload your file following the directions.
5. Press the SUBMIT button.
6. Press the DONE button near the bottom of the page.
7. Check for a confirmation email.
8. Save the confirmation email (proof of submission).