Homework 2: Turtles and Functions

Comp 123

Susan Fox

1 Overview

This assignment should be done as individual work. Be sure the work is your own. Do not seek out solutions online. DO ask for help from me or our preceptors. DO write down any people or web sites where you get some measure of help.

See the Homework Guidelines document for further information about standards for homework.

For this assignment, you are expected to include a summary, descriptive comment for each function or script you write. This is a standard requirement; starting on homework 3 you will lose 1 point per function for not including this.

I am providing you with a test file for testing the functions you write. Read it carefully, and use it to test your functions, as shown in class.

1.1 Assignment goals

For this assignment, you will practice reading and writing functions, and understanding how functions calling functions works.

1.2 Preparing and handing in the assignment

Download the `hw2Files.zip` archive file. Inside it you will find the `hw2Code.py` and `hw2Tests.py` files to use with this assignment. Put all your programming answers into the `hw2Code.py` file, and hand in just this file when you are done.

To mark the start of each question, leave blank lines, and then give a hash-mark comment with the problem number. Inside the function definition, just after the `def` line, give a summary, descriptive comment using a triple-quoted string. One to two full sentences that describe the function's purpose, its inputs (types and meanings), and any outputs it produces.

Any non-code answers may be included in the Python file as either triple-quoted strings or hash-marked comments.

Using the test file: When you create larger programs, particularly programs built out of functions, you need to thoroughly test each function to make sure that it does what it is supposed to do. This is called "unit testing." For this homework, I am providing you with a test file that contains thorough tests for this assignment's functions. By the end of the semester, you should be able to write your own tests for a given function.

You should open and read through the test file. It may contain some script elements, and it also contains functions, one function for each function in your homework. Each function name in the test file starts with

the word `test`, followed by the name of the function it is testing. Eventually you will be able to understand everything inside the testing functions, but for now just get a general sense: each testing function makes multiple calls to your functions, displaying the results. When possible, it checks the result to see if it is correct, but for visual results (like turtle graphics drawings), it just prints a message describing what you should see.

At the top of the testing file is a function called `runTests`, and inside it are a series of calls to each testing function. They are commented out; when you have finished a problem, then you may test it by uncommenting the call to its testing function, and then running the testing file. If you look at the very bottom of the test file, you will see a call to `runTests`; that ensures that when you run the testing file it will start `runTests`, which will then run the tests you want.

2 Assignment questions

1. (5 pts) Explain the difference between the two functions and calls below (Note that a copy of this is in `hw2Code.py`). What is printed by this? What are the values of `r1` and `r2`? Why do they differ? Put your answer as a comment in `hw2Code.py`.

```
def proc1(x):
    print(3 * x)

def proc2(x):
    return 3 * x

r1 = proc1(5)
r2 = proc2(5)
print(r1, r2)
```

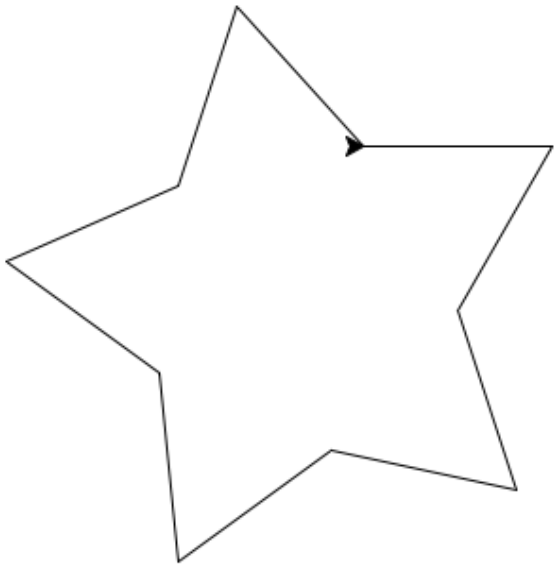
2. (5 pts) Look at the `proc2` definition above. It is made up of the separate elements below. Put a comment in the `hw2Code.py` file, explaining what the purpose and meaning of each element is in defining the function:

   (a) `def`
   (b) `proc2`
   (c) open parenthesis `(`
   (d) `x`
   (e) close parenthesis `)`
   (f) colon `:`
   (g) indentation of statement
   (h) `return`
   (i) `3 * x`

3. (10 pts) Define a function `starOutline` that takes two inputs. The first input should be a `Turtle` object, and the second should be an integer, a side length. The function should tell the input turtle to

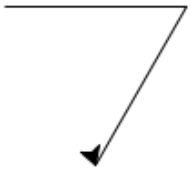
draw the shape at the turtle's current location and heading. Don't move the turtle, and don't make a new screen or turtle inside this function.

This function should draw the outline of a five-pointed star, like the one shown below.



This shape is drawn by a series of moves forward, all exactly the same length. The turtle repeats the same sequence of movements five times, one for each point of the star. It ends exactly where it started, facing exactly the same direction it was at the start.

A single sequence draws a point of the star, and turns the turtle to face the correct direction for the next point of the star. The turtle moves forward, then turns right 120 degrees for the point, moves forward again the same distance, and then turns left 48 degrees for the trough/valley between the two peaks. After a single sequence, the picture should look something like this:

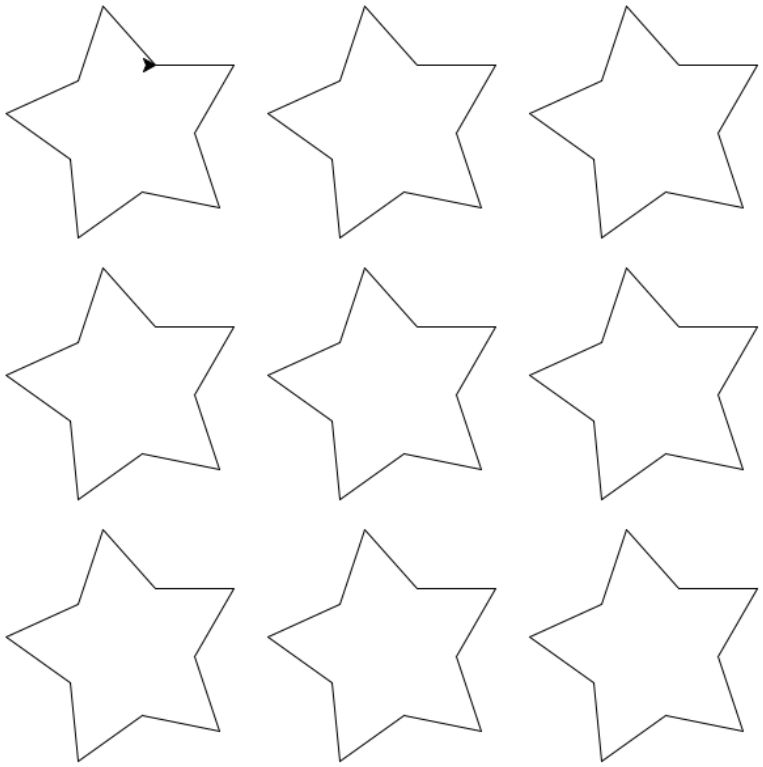


Inside your function, you should first create a `for` loop that repeats five times. Inside the loop, create steps that make the turtle named as an formal parameter (an input) perform the four steps described above: move forward the distance given by the second formal parameter, then turn right 120 degrees, move forward again, then turn left 48 degrees. Use incremental development: write a bit of code, then test it until it works, and only then go on.

Your function is meant to be part of a bigger program. Therefore the following constraints apply: **Do not create a screen or a turtle inside your function, and do not call `exittoclick` inside the function, either.**

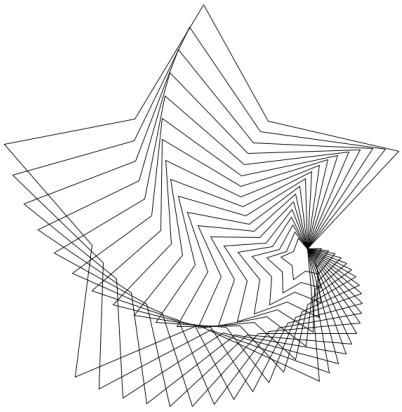
Why would we want a function that doesn't create a screen or a turtle inside it, but rather takes a turtle as its input? Well, maybe we want to draw the shape over and over:

`gridOfStars()`

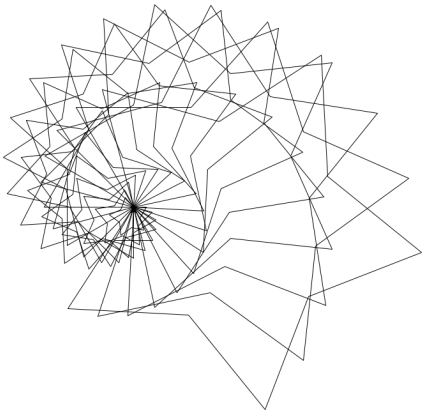


Or draw it in rotation:

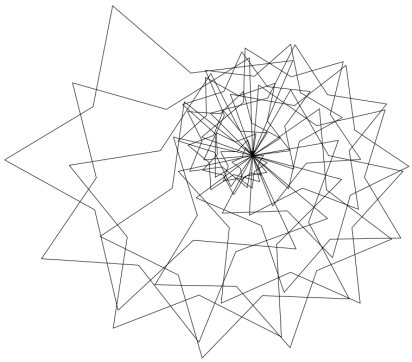
`spiralOfStars(5)`



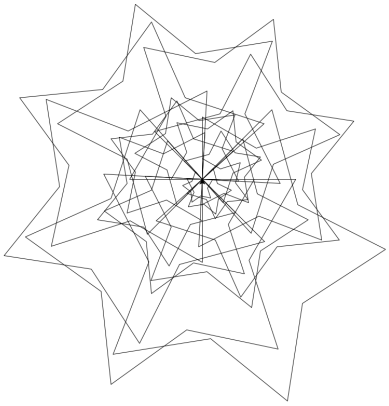
`spiralOfStars(15)`



`spiralOfStars(25)`



`spiralOfStars(45)`



In the `hw2Code.py` file you will find a place to write your function. You will also find two defined functions, `gridOfStars` and `spiralOfStars` that use your function, `starOutline` as a helper. The `spiralOfStars` function can generate all four of the pictures above, by varying the angle it rotates between each star shape.

Testing: Examine the first testing function in the `hw2Tests.py` file. Then do the following to test your function:

(a) Make sure that the `hw2Tests.py` file is saved in the same folder as your solution file.

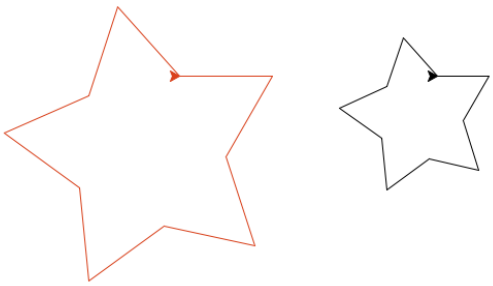
(b) Change the `import` statement at the top of the `hw2Tests.py` file so that the filename there matches your filename. Note that the `import` statement leaves off the `.py` extension.

(c) Uncomment the call to `testStarOutline` in the `runTests` function.

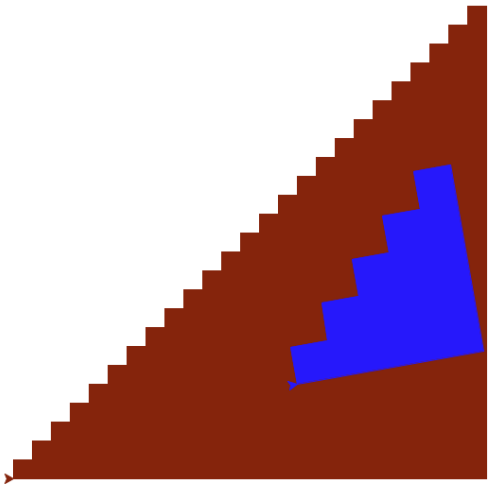
(d) Run the `hw2Tests.py` file, and see what it displays and prints. Note that you will need to type a key into the Python shell to complete the run of each test function.

The test program should display four windows. The first has one turtle that draws the grid of stars shown above. The second window has one turtle that draws the `spiralOfStars(5)`

shown above. The third has one turtle that draws the `spiralOfStars(15)` shown above. The fourth window has two turtles, a black one and a red one, and it should end up looking like this:



4. (10 pts) Examine the function `drawStairs` in the `hw2Code.py` file. This function is supposed to take in three inputs, and draw a filled-in staircase (see the picture below for two examples). The inputs are the turtle object to do the drawing, the number of steps, and the step size (both vertical rise and horizontal step width are the same).



The stairs start where the input turtle is, with the step surfaces parallel to the turtles initial heading (in the picture above, the blue turtle has turned ten degrees to the left).

If you try to run `drawStairs`, it will not work. I have given you a buggy function! The `drawStairs` function has **three** bugs in it. For simplicity each bug occurs on a single line, and fixing the bug requires making changes to that line, not adding or removing any lines.

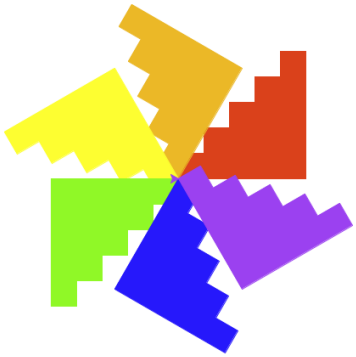
Find the three bugs in `drawStairs`. When you identify a bug, put a comment on that line describing the bug, and then fix the bug.

Remember the different debugging techniques we've talked about, including: looking for red underlining to mark syntax errors, adding print statements to see how far the code is getting, commenting out part of the code to see what other parts do, and working through what the function does line by line to see what it does by hand. Or using the debugger to walk through the program line by line.

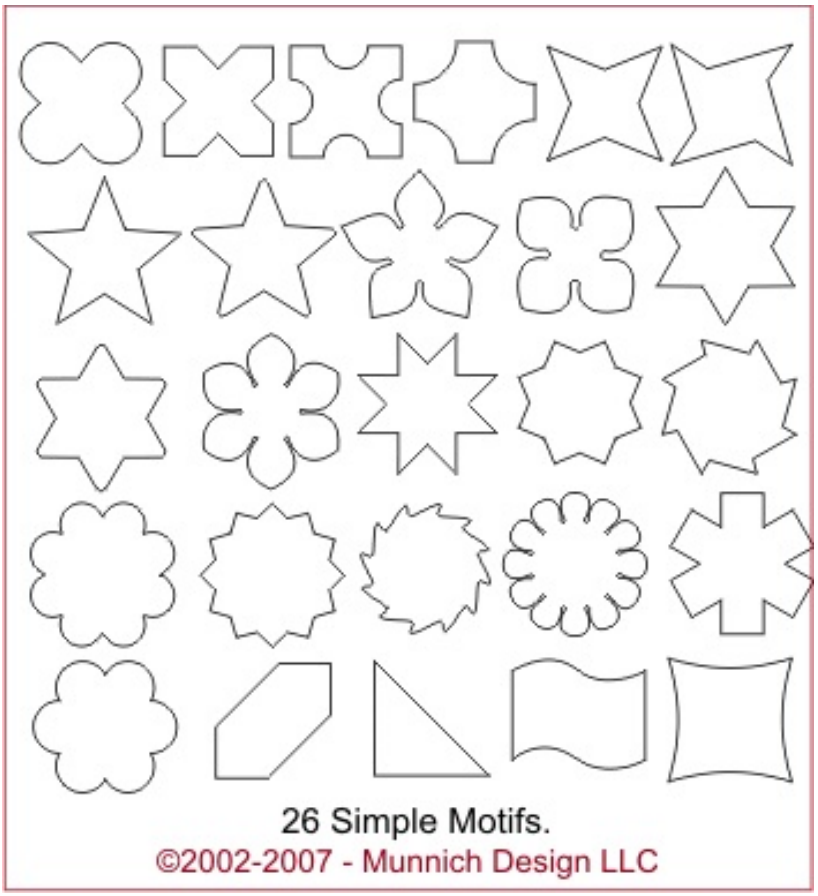
Testing: The testing file has a test function for `drawStairs` as well. To verify that you are right, follow the steps above, but uncomment the `testDrawStairs` call at in the `runTests` function in `hw2Tests.py`.

The testing program should display two windows. The first should have two turtles and three stairs as shown below:

The second window should have one turtle, and it should display a rainbow spiral of stairs, each the same size, but rotated:



Extra credit, for those who need an extra challenge: The two shapes in this assignment were derived from the shapes in the diagram below (found online from a Google search). Pick up to three additional shapes, and figure out how to draw them with turtle graphics. Define a function for each shape, similar to the `starOutline` function above. Make a variation of `spiralOfStars` for each additional shape. For SUPER extra credit, make your new version of `spiralOfStars` take a function as input, and pass it one of the shape functions you've defined, so that one function can be used to draw any of the shapes you've implemented (see me for more information).



26 Simple Motifs.