Loops and Functions

Andrew Rosen

Read this paragraph. For each of the following problems, write a function that solves the problem. Demo each function you write by calling it. You should either do these all in the same file or in one file for the text based problems and another for the turtle problems. **DO NOT USE THE INPUT FUNCTION INSIDE THE FUNCTIONS YOU WRITE**

Hints 1: Creating a Function in Python

You can create your own functions in Python. This is essentially making your own custom command in Python, adding another verb to the Python language. Let's look at the following program.

```
def manyHellos(n):
        for i in range(n):
                print("Hello")


hellos = int(input("How many hellos do you want?"))
manyHellos(hellos)
```

This program will print out "Hello" a number of times equal to the user's input. The first line defines a new function (command) for your python program, called `manyHellos`. The `manyHellos` function requires the user to pass in a number as an argument in order for it to work. By the way, you shouldn't use `hellos` as a variable in your program; that will tell me you're just copy/pasting. Also note that the input statement is outside the function; this is ideal place to put it since it can easily be replaced by a literal for easy testing without changing the function.

If you need to review how to write functions, refer to the linked readings: `https://runestone.academy/runestone/assignments/doAssignment?assignment_id=42928`

Hints 2: Printing with sep and end

There are three special things we can do with print in python.

Printing will multiple arguments

The first is giving a print function call multiple arguments:

```
print("Hello",'World', 1)
```

This will print `Hello World 1`. *By default*, your arguments are printed with a single space between them.

sep

We can use the `sep` parameter to override the default behavior of printing multiple arguments. Let learn by looking at a few examples:

```
print("Hello",'World', 1)              # will print Hello World 1
print("Hello",'World', 1, sep='_')          # will print Hello_World_1
print("Hello",'World', 1, sep='taco') # will print HellotacoWorldtaco1
print("Hello",'World', 1, sep='')           # will print HelloWorld1
print("Hello",'World', 1, sep=' ')          # the default behavior
```

end

We can change how print ends a line by using the `end` parameter.

```
for i in range(5):
        print(i,end="") # will print 01234
print() #goes to a new line of output
```

You can combine `sep` and `end` in the same statement if needed.

1 Part 1: Printing with Text

1.1 99 Bottles of Beer

Write a function that uses a for loop to print out the the lyrics of the infamous "99 Bottles of Beer on the Wall" drinking song. However, this function should take in an `int` as a parameter and start the lyrics from there. For example, if the function is called with 10 as the parameter, the output should be:

```
10 bottles of beer on the wall, 10 bottles of beer
Take one down, pass it around, 9 bottles of beer on the wall

9 bottles of beer on the wall, 9 bottles of beer
Take one down, pass it around, 8 bottles of beer on the wall

... (output continues in the same pattern) ...

1 bottles of beer on the wall, 1 bottles of beer
Take one down, pass it around, 0 bottles of beer on the wall
```

1.2 Multiplication Table

Write a function which, given an integer n as an input, prints out an $n \times n$ multiplication table.

If n is 4, print out a 4×4 multiplication table like below

```
1    2    3    4
2    4    6    8
3    6    9    12
4    8    12    16
```

If n is 5, you want to print out:

```
1    2    3    4    5
2    4    6    8    10
3    6    9    12    15
4    8    12    16    20
5    10    15    20    25
```

Try to get the spacing as neat as you can!

1.3 Summation of squares

Write a function which, given an integer n, uses a for loop to print out the sum of all numbers squared from 1 to n. For example, if the given integer is 5, the program should print out 55, as $1^2 + 2^2 + 3^2 + 4^2 + 5^2 = 55$.

2 Teenage Mutant Function Turtles

Each of these problems will build off of the turtle skills we have learned in class to create more and more complex figures.

A few pointers before we get to the actual problems:

- Some of the functions involve a lot of drawing. You can speed it up by using the `myTurtle.speed(0)` command, where *myTurtle* is whatever variable you have for your turtle. Click this text to learn more.
- Don't use the `input` function in your functions.
- The function code below includes the word `pass` below the header. This is to ensure your code does not crash if you try to run your program immediately after copy pasting. You should delete it when you being coding the functions.

2.1 Stairs

2.1.1 DrawSquare

Create a function called drawSquare, which takes in a turtle and a number called *squareSize* and uses that turtle to draw a square with each side *squareSize* pixels long.

Your function should start like this:

```
# myTurtle is the turtle doing the drawing
# size is the length of each of the sides of the square.
def drawSquare(myTurtle, squareSize):
        pass
```

2.1.2 Drawing a Row of Squares

Write a function called `drawRow` that draws a row of squares. `drawRow` should call (use) `drawSquare` to accomplish this task. See the comments in the code for more details.

```
# draws a row of squares
# myTurtle is the turtle doing the drawing
# length is how many squares are in the row
# squareSize is the length of each of the sides of each square.
def drawRow(myTurtle, length, squareSize):
        pass
```

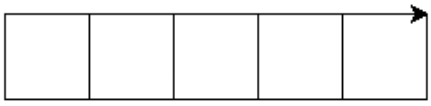

Figure 1: The output of `drawRow(bob,5,50)`.

2.1.3 Drawing a Grid

Write a function called `drawGrid` that draws a square grid of squares. `drawGrid` should call (use) `drawRow` to accomplish this task. See the comments in the code for more details.

```
# myTurtle is the turtle doing the drawing
# the grid will be "size" squares wide and "size" squares tall
# squareSize is the length of each of the sides of each individual square.
def drawGrid(myTurtle, size, squareSize):
        pass
```

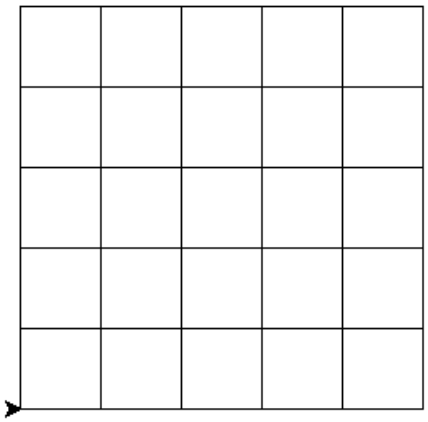

Figure 2: Output of `drawGrid(bob,5,50)`.

2.1.4 Drawing a Stair of Squares

Write a function called `drawSquareStairs` that draws a staircase made of squares, like in Figure 3. `drawSquareStairs` should call (use) `drawRow` to accomplish this task. See the comments in the code for more details.

```
# myTurtle is the turtle doing the drawing
# height is how tall the stairs are going to be
# squareSize is the length of each of the sides of each square.
def drawSquareStairs(myTurtle, height, squareSize):
```

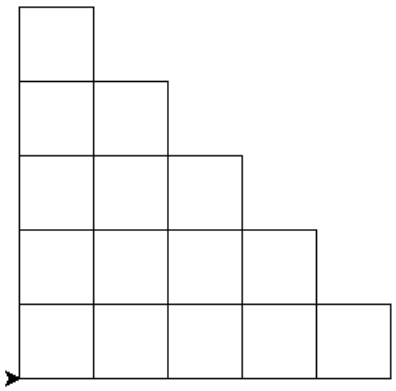

Figure 3: Output of `drawSquareStairs(bob,5,50)`.

2.2 Spirals

2.2.1 N Sided Polygon

Create a function called drawNgon, which takes in a turtle and two numbers - *numSides* and *sideLength* - and uses that turtle to to draw a polygon with *numSides* sides with each side *sideLength* pixels long.

Your function should start like this:

```
def drawNgon(myTurtle, numSides, sideLength):
        pass
```

This is very similar to the draw shape problem from the previous week.

Calling the function like so- `drawNgon(bob,6,100)` - where bob is the name of the turtle, would result in the following output:

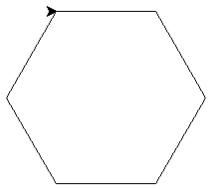

Figure 4: The output for `drawNgon(bob,6,100)`

2.2.2 Super Spiral

Create a function that can draw a spiral made of shapes using the following function. This should use the previous function. Feel free to add additional parameters if needed.

```
# draws a spiral of shapes.
# this is done by drawing a single polygon, rotating the turtle a bit,
# drawing another polygon until the spiral is completed.
# numSides defines the shape of the polygons
# sideLength defines how big each polygon is
# numShapes defines how many polygons make up the spiral
def drawNgonSpiral(myTurtle, numSides, sideLength, numShapes):
        pass
```

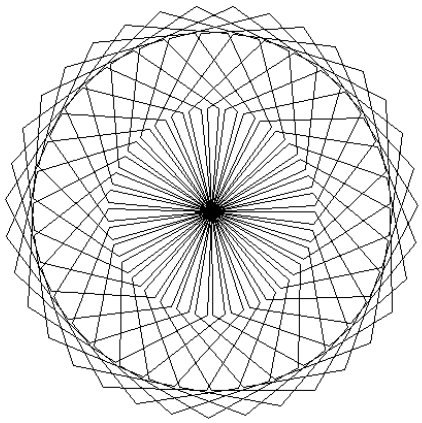

Figure 5: `drawNgonSpiral(bob,6,100,35)` creates a spiral of 35 hexagons. Each hexagon is offset by 20.57 degrees ($720 \div 35 \approx 20.57$). Your output does not need to match my output; this is here to give you a better idea of what to expect.

3 Bonus (5 pts each)

3.1 Hourglass

Write a function that creates the following figure of an hourglass. This function takes no inputs. You must use for loops to draw the figure

```
|"""""""""""|
 \:::::::::/
  \:::::::/
   \:::::/
    \:::/
     |||
    /:::\
   /:::::\
  /:::::::\
 /:::::::::\
|"""""""""""|
```

Use for loops to print out spaces and colons. I highly recommend writing 2 separate loops, one for the top part and one for bottom part.

3.2 Slash Figure

Write a function, which given an `int` n, prints out a slash-based ASCII art of size n. Below is an example of what the output looks like at size 4:

```
!!!!!!!!!!!!!!
\\!!!!!!!!!!//
\\\\!!!!!!////
\\\\\\!!//////
```

And size 6

```
!!!!!!!!!!!!!!!!!!!!!!
\\!!!!!!!!!!!!!!!!!!//
\\\\!!!!!!!!!!!!!!////
\\\\\\!!!!!!!!!!!!//////
\\\\\\\\!!!!!!!!////////
\\\\\\\\\\!!!!!!!!!!
```

And size 7:

```
!!!!!!!!!!!!!!!!!!!!!!!!!!
\\!!!!!!!!!!!!!!!!!!!!!!//
\\\\!!!!!!!!!!!!!!!!!!////
\\\\\\!!!!!!!!!!!!!!!!//////
\\\\\\\\!!!!!!!!!!!!////////
\\\\\\\\\\!!!!!!!!!!!!
```

3.3 Super Duper Spiral

Modify `drawNgon` and `drawNgonSpiral` or create new functions so that the turtle gradually changes the color of the spiral as it draws the shape, like in Figure 6.

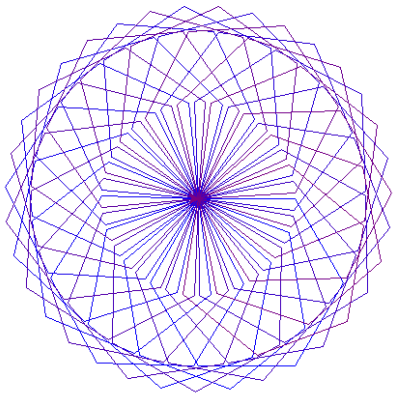

Figure 6: The output of drawNgonSpiral(bob,3,100,3) with color.

4 Grading

Each problem is worth 20 points, broken down as follows:

12 points The problem is solved as directed. Partial credit may be given for partial solutions at the grader's discretion.

3 points The code is properly indented and easy to read.

5 points The problem is in a function.

This means there are a total of 180 points, excluding the extra credit.