COMP 142 Project 4: Turtle Maze

Assigned: Friday, October 10
Due: Thursday, October 23, 2014 by 11:55pm (on Moodle)

Normally maze experiments are done with rats and cheese, but all you have is a turtle. I don't know what turtle's eat, so each maze just has a goal marked for the turtle to get to. Don't worry, the turtle is pretty quick for a turtle! The goal of this program is to allow the turtle to search through the maze recursively until he's exhausted all options or found his goal.

Program Description/Specification

Exploring a maze is a great example of a problem easily solved with recursion. To make it easier for you, assume that the maze is divided up into "squares". Each square of the maze is either open or occupied by a section of wall. The turtle can only pass through the open squares of the maze. If the turtle bumps into a wall it must try a different direction. The turtle will require a systematic procedure to find its way to it's "goal square" in the maze. Here is the procedure:

- From the starting position first try going North one square and then recursively try the procedure from there.
- If you are not successful by trying a Northern path as the first step then you will take a step to the South and recursively repeat our procedure.
- If South does not work then you will try a step to the West as your first step and recursively apply your procedure.
- If North, South, and West have not been successful then apply the procedure recursively from a position one step to your East.
- If none of these directions works then there is no way to get to the goal square of the maze and the turtle fails.

Now, that sounds pretty easy, but there are a couple of details to talk about first. Suppose you take your first recursive step by going North. By following your procedure your next step would also be to the North. But if the North is blocked by a wall you must look at the next step of the procedure and try going to the South. Unfortunately that step to the south brings you right back to your original starting place. If you apply the recursive procedure from there you will just go back one step to the North and be in an infinite loop. So, you must have a strategy to remember where you have been. In this case you will assume that you have a bag of bread crumbs you can drop along our way. If you take a step in a certain direction and find that there is a bread crumb already on that square, you know that you should immediately back up and try the next direction in your procedure. Backing up is as simple as returning from a recursive function call. As we do for all recursive algorithms let us review the base cases. Some of them you may already have guessed based on the description in the previous paragraph. In this algorithm, there are four base cases to consider:

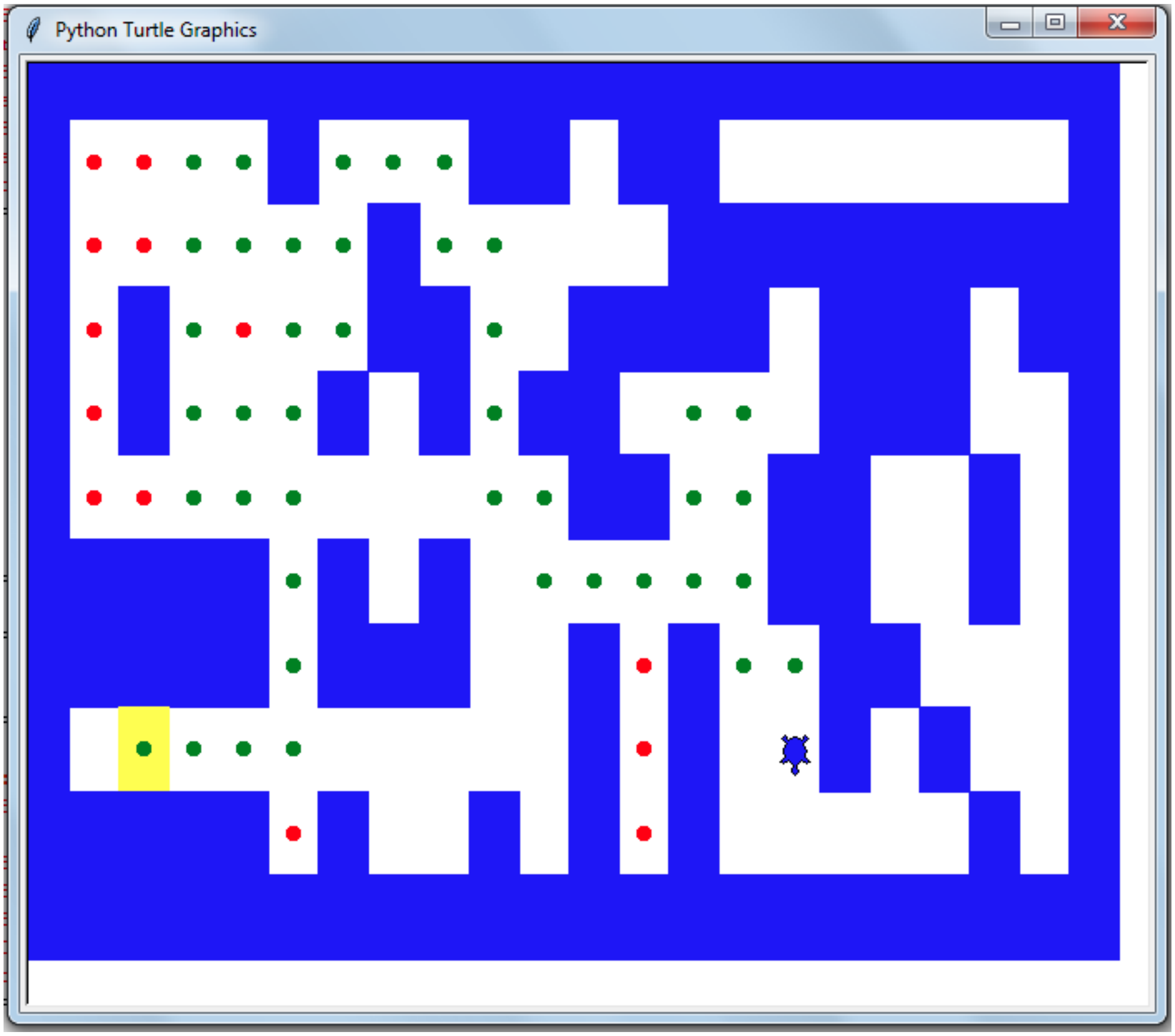
- The turtle has run into a wall. Since the square is occupied by a wall no further exploration can take place.
- The turtle has found a square that has already been explored. We do not want to continue exploring from this position or we will get into a loop.
- The turtle has found the goal square and the turtle is happy!
- We have explored a square unsuccessfully in all four directions.

For your program to work you will need to have a way to represent the maze. To make this even more interesting you are going to use the turtle module to draw and explore the maze so you can watch this algorithm in action. Don't worry, I've provided the maze.py code for you. This file has the TurtleMaze class code as well as the stub with pseudocode fort eh searchFrom function and a main driver function already in it. Your job is to fill in the base cases and the recursive cases for the searchFrom function.

1. You must write a recursive function called searchFrom that takes in a TurtleMaze object and a startRow and startColumn.
2. I have provided the TurtleMaze class code as well as the stub with pseudocode for the searchFrom function. The code also has the main driver function in it already.
3. The code is available here: maze.py
4. The maze will be read in from a file. You can download the maze2.txt file from here. Be sure to put it in the same folder as your Python file.
5. Feel free to create your own maze files and test your code with those as well. At the very least, modify the maze2.txt file and move the start position and goal position around.

Testing your Program

If you follow the instructions and use the same order of directions for exploration, the final maze will look like the image below for the maze2.txt file I've included.



Language

You should write this program in Python (last one in Python).

Coding style, comments, and pledge

Please see Program 1 for details on proper coding style, comments and including the pledge in your program header. Don't forget the program header!
Since I'm providing the class code, I would like you add additional comments to the code to explain each method in the TurtleMaze class. You write should have a high-level comment describing what it does, and each of its method functions should have a comment before it similar to those of regular functions (with a description, parameters and returns).

What to turn in

Through Moodle, turn in your code as a file called `maze_`*`yourLastName_yourFirstName`*`.py`.