Assignment 1

Selene Waide
16201922

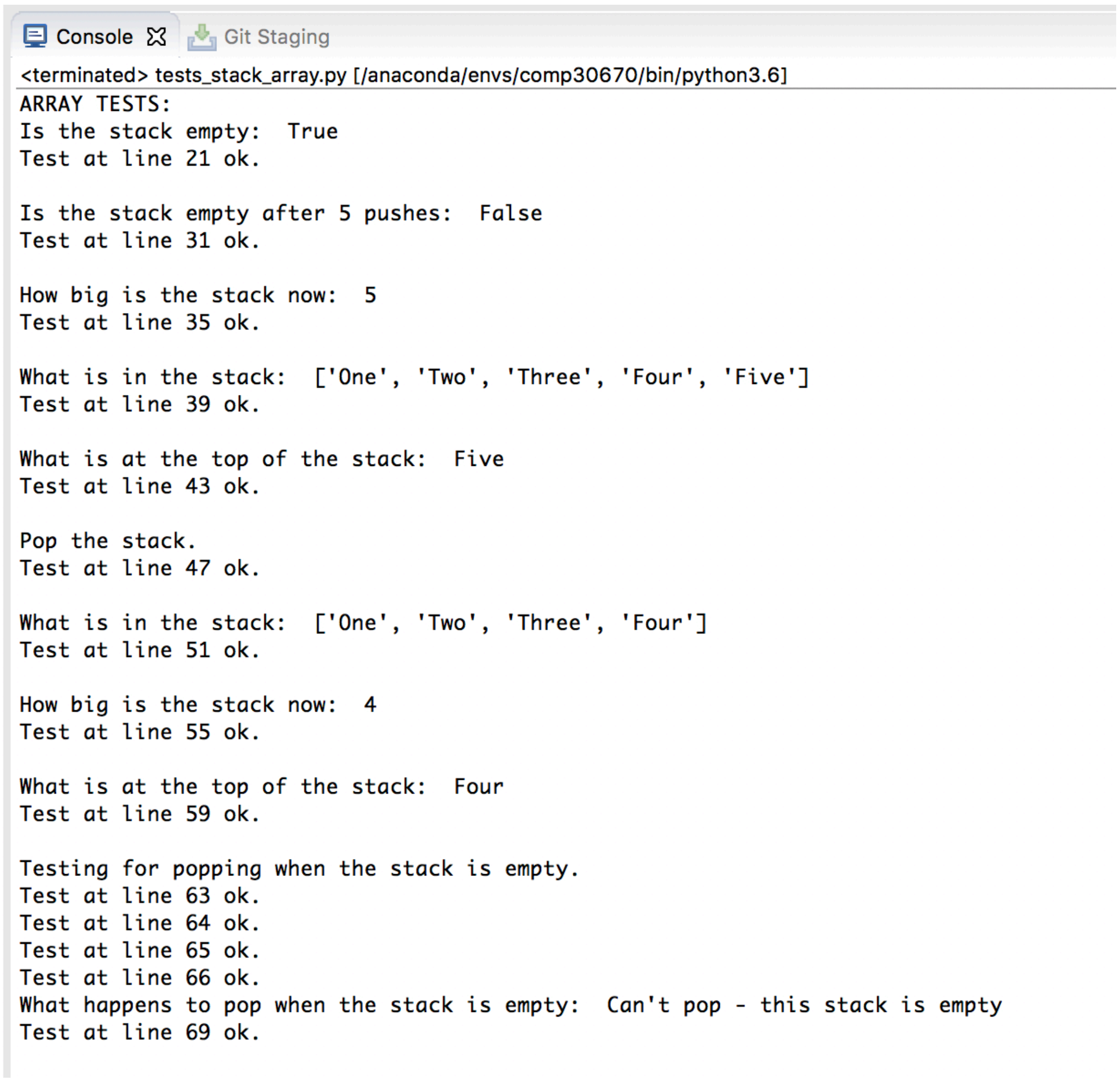
Exercise 1. Stack ADT
The objective is to implement two stack ADTs - one using python array, and another using a linked list. There is a test file for each stack.

Four files are included for this exercise:

- stack_array.py
- stack_linked_list.py
- tests_stack_array.py
- tests_stack_linked_list.py

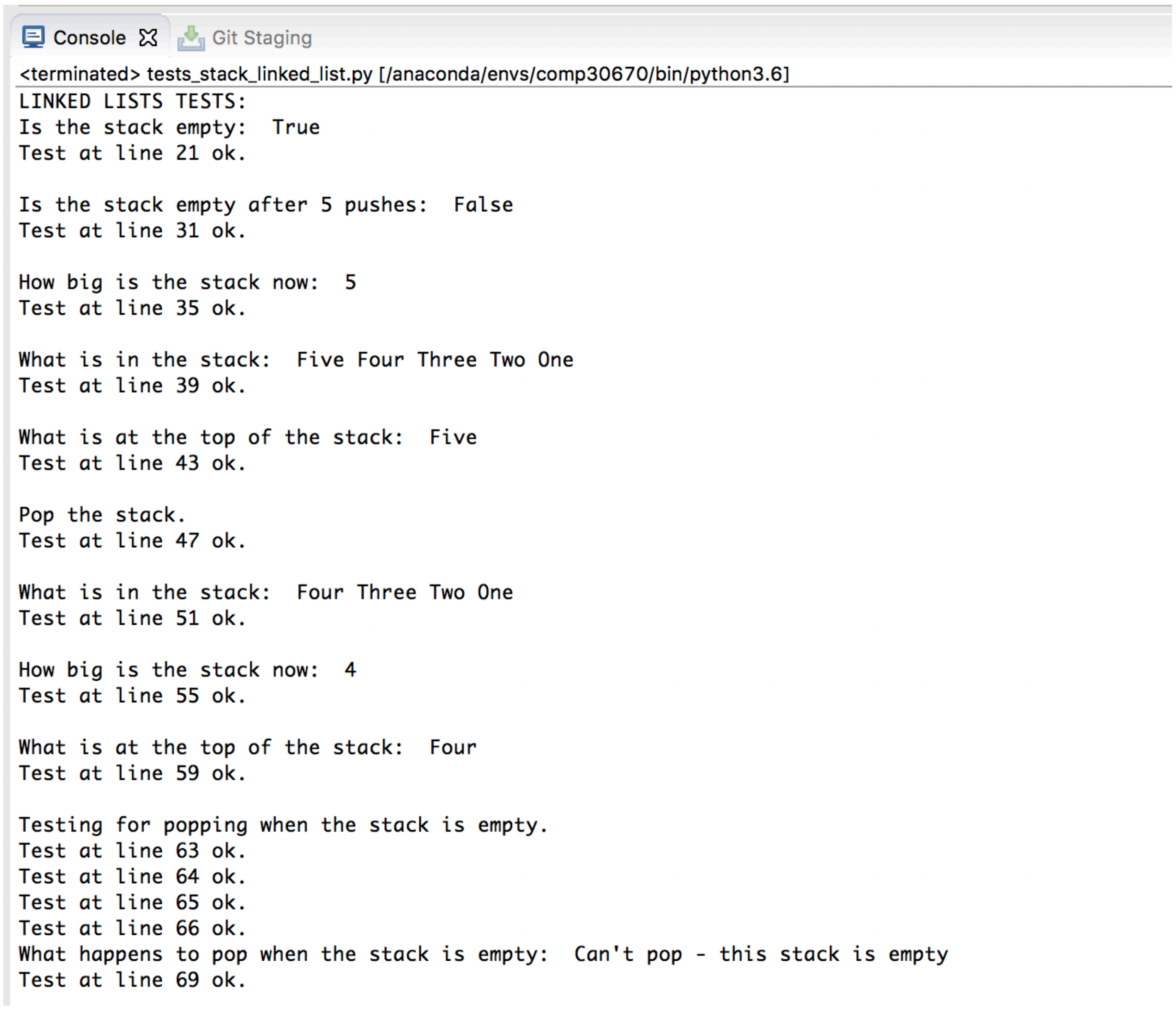
The stack_array.py ADT pushes and pops items to and from the right. The stack_linked_list.py ADT pushes and pops items to and from the left.

When tests_stack_array.py is run, the output is as follows:



```
<terminated> tests_stack_array.py [/anaconda/envs/comp30670/bin/python3.6]
ARRAY TESTS:
Is the stack empty:  True
Test at line 21 ok.

Is the stack empty after 5 pushes:  False
Test at line 31 ok.

How big is the stack now:  5
Test at line 35 ok.

What is in the stack:  ['One', 'Two', 'Three', 'Four', 'Five']
Test at line 39 ok.

What is at the top of the stack:  Five
Test at line 43 ok.

Pop the stack.
Test at line 47 ok.

What is in the stack:  ['One', 'Two', 'Three', 'Four']
Test at line 51 ok.

How big is the stack now:  4
Test at line 55 ok.

What is at the top of the stack:  Four
Test at line 59 ok.

Testing for popping when the stack is empty.
Test at line 63 ok.
Test at line 64 ok.
Test at line 65 ok.
Test at line 66 ok.
What happens to pop when the stack is empty:  Can't pop - this stack is empty
Test at line 69 ok.
```

When tests_stack_linked_list.py is run, the output is as follows:



```
<terminated> tests_stack_linked_list.py [/anaconda/envs/comp30670/bin/python3.6]
LINKED LISTS TESTS:
Is the stack empty:  True
Test at line 21 ok.

Is the stack empty after 5 pushes:  False
Test at line 31 ok.

How big is the stack now:  5
Test at line 35 ok.

What is in the stack:  Five Four Three Two One
Test at line 39 ok.

What is at the top of the stack:  Five
Test at line 43 ok.

Pop the stack.
Test at line 47 ok.

What is in the stack:  Four Three Two One
Test at line 51 ok.

How big is the stack now:  4
Test at line 55 ok.

What is at the top of the stack:  Four
Test at line 59 ok.

Testing for popping when the stack is empty.
Test at line 63 ok.
Test at line 64 ok.
Test at line 65 ok.
Test at line 66 ok.
What happens to pop when the stack is empty:  Can't pop - this stack is empty
Test at line 69 ok.
```

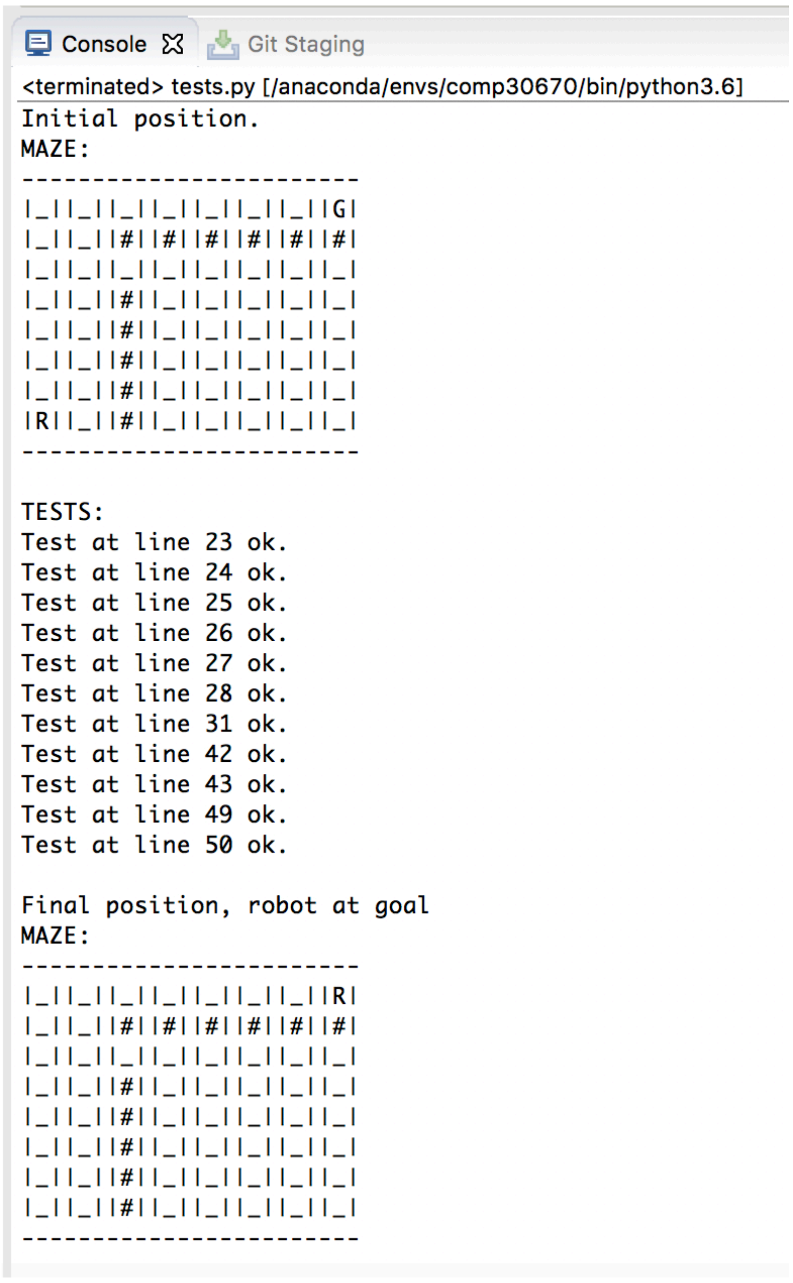
Exercise 2. Creating world and making the robot move:
The objective is to create a two dimensional array, forming a maze with a path and walls. A robot is placed in this maze, and has to navigate to a specific goal.

Three files are involved in this process.

- around_the_maze.py
- tests.py
- world1.dat

"around_the_maze.py" contains functions for creating the maze, finding the robot, seeing whether a cell on the maze is feasible, moving the robot, reaching the goal, and printing the maze. "tests.py" calls these functions and tests each of them. When the functions are called, "world1.dat" file is read in order to build the maze.

When "tests.py" is run, the output is as follows:



```
<terminated> tests.py [/anaconda/envs/comp30670/bin/python3.6]
Initial position.
MAZE:
------------------------
|_||_||_||_||_||_||_||G|
|_||_||#||#||#||#||#||#|
|_||_||_||_||_||_||_||_|
|_||_||#||_||_||_||_||_|
|_||_||#||_||_||_||_||_|
|_||_||#||_||_||_||_||_|
|_||_||#||_||_||_||_||_|
|R||_||#||_||_||_||_||_|
------------------------

TESTS:
Test at line 23 ok.
Test at line 24 ok.
Test at line 25 ok.
Test at line 26 ok.
Test at line 27 ok.
Test at line 28 ok.
Test at line 31 ok.
Test at line 42 ok.
Test at line 43 ok.
Test at line 49 ok.
Test at line 50 ok.

Final position, robot at goal
MAZE:
------------------------
|_||_||_||_||_||_||_||R|
|_||_||#||#||#||#||#||#|
|_||_||_||_||_||_||_||_|
|_||_||#||_||_||_||_||_|
|_||_||#||_||_||_||_||_|
|_||_||#||_||_||_||_||_|
|_||_||#||_||_||_||_||_|
|_||_||#||_||_||_||_||_|
------------------------
```

The tests used here are those given in the assignment. “R” denotes the robot, “G” denotes the goal, walls are “l#l” and a valid path is “l_l” .

To complete this exercise, I started with the “tests.py” file. I took each test one at a time, created the “around_the_maze.py” file and wrote functions in that file to pass the tests in the test file.

I separated the code from the final output, i.e. the maze is actually constructed with zeros and ones, but when I print this array I use a print function to convert those values to the symbols I mention above. This made it easier to solve the problem of moving the robot around the world.

Exercise 3. Finding the shortest path:
The objective is to write a recursive algorithm to enable the robot to navigate the maze to the specified goal.

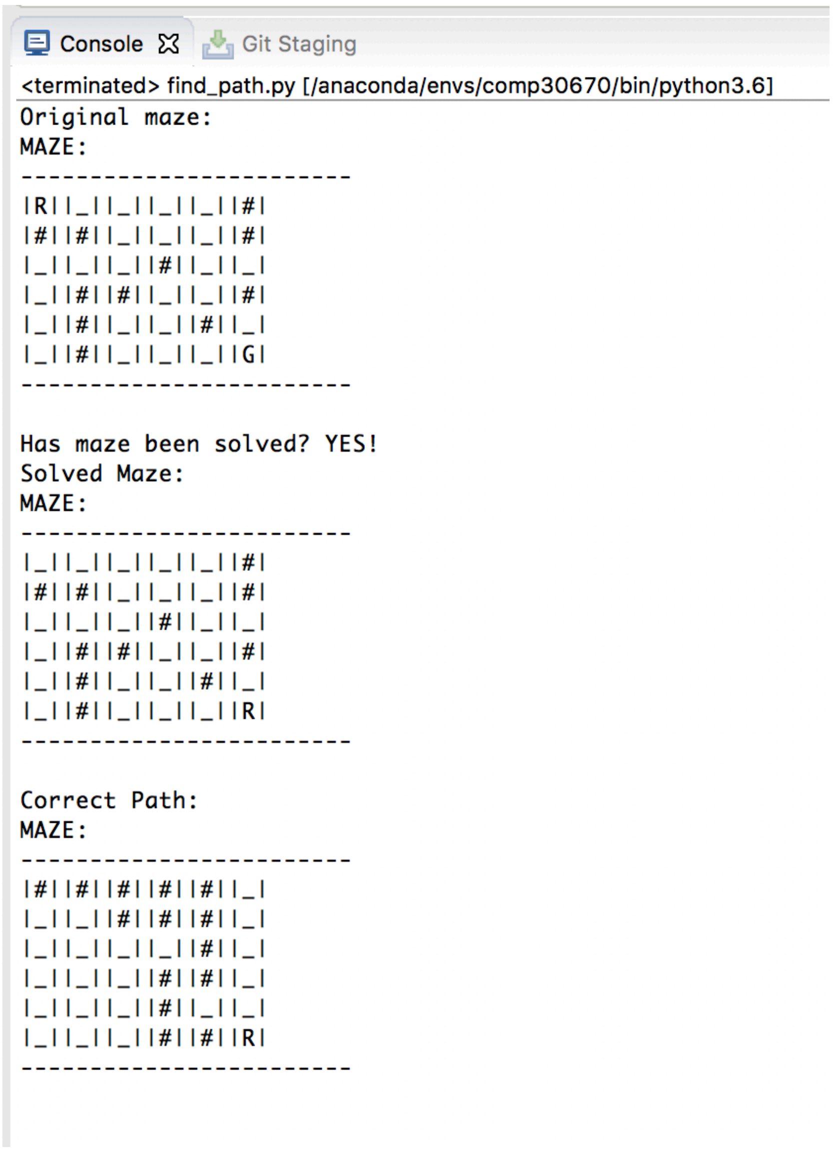
Three files are involved in this process.

- around_the_maze.py
- find_path.py
- world2.dat

“find_path.py” has two functions - solveMaze and recursiveSolve. This file calls functions in the “around_the_maze.py” file. solveMaze builds the maze, and initialised two boolean arrays to track where the robot has been, and the path of the robot. It calls the function recursiveSolve and assigns it to a boolean called “has_maze_been_solved”. This boolean is used to print whether or not the maze has been solved.

recursiveSolve function is used to move the robot around the maze recursively using the functions in “around_the_maze.py”. It returns “true” if goal is reached, and “false” otherwise. This is picked up by “has_maze_been_solved” as discussed above.

When “find_path.py” is run, the output is as follows:



```
<terminated> find_path.py [/anaconda/envs/comp30670/bin/python3.6]
Original maze:
MAZE:
------------------------
|R||_||_||_||_||#|
|#||#||_||_||_||#|
|_||_||_||#||_||_|
|_||#||#||_||_||#|
|_||#||_||_||#||_|
|_||#||_||_||_||G|
------------------------

Has maze been solved? YES!
Solved Maze:
MAZE:
------------------------
|_||_||_||_||_||#|
|#||#||_||_||_||#|
|_||_||_||#||_||_|
|_||#||#||_||_||#|
|_||#||_||_||#||_|
|_||#||_||_||_||R|
------------------------

Correct Path:
MAZE:
------------------------
|#||#||#||#||#||_|
|_||_||#||#||#||_|
|_||_||_||_||#||_|
|_||_||_||#||#||_|
|_||_||_||#||_||_|
|_||_||_||#||#||R|
------------------------
```

The first maze above shows the original setup, before the robot starts moving towards the goal. The next maze is the output once the robot has reached the goal (is possible). The final maze is a rendition of the path the robot takes - provided by the boolean array correctPath.