Lab Assignment Week 09

CSC/DSCI 1301 – Principles of CS/DS I

Week of October16th, 2023

Introduction

Welcome to the second programming lab of CSC/DSCI 1301! Today we will be covering the following topics:

- Creating & Accessing Nested Lists

Lab policy reminders:

- Attendance is mandatory.
- Labs must be completed **individually.**
- TAs are here to help you. Ask them for help!
- Lab assignments are due at the end of each lab.

Deliverables:

1. Python files for your random walk program.
   a. random_walk.py
2. Screenshots of your program output

If you have any questions, please do not hesitate to ask your TA!

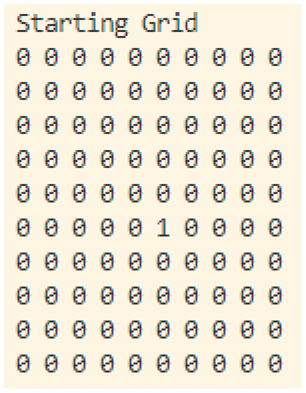
Program 1: random_walk.py

An agent-based model is a computational model for simulating the actions and interactions of autonomous agents to understand the behavior of a system and what governs its outcomes. Agent-based models are a common type of computational model used in Computational Epidemiology for simulating infectious disease spread through contact between individuals. Depending on the context, an agent-based disease simulation can contain dozens of agents when simulating disease spread in a classroom to a computational model with thousands of agents when simulating disease spread on a cruise ship.

For your lab this week, you will need to write a program that simulates the movement of a single autonomous random agent in a 2-dimensional space. The 2-dimensional space will be represented by a finite grid. The random agent will start at the center and move from one position to another in the grid. It can only move in one of the four cardinal directions: Up, Down, Left, and Right. The agent's movement will be entirely determined by a random number generator. Depending on the value random number, the agent will choose to move up, down, left, or right in the grid. For this experiment, you will need to simulate your random agent making 100 random moves.

2-Dimensional Grid

For this experiment, you will create a 2-dimensional grid with a fixed size of **15x15**. You will need to create a 2-dimensional nested list to represent the grid that the agent can move around in. The indexes of the outer list will represent the rows of the grid, while the indexes of the inner list will represent the columns of the list. The nested list index [0][0] will represent the top left corner square of the grid and [14][14] will represent the bottom right corner square. The initial value of all grid squares will be 0. Except for the starting grid square of the agent, that square will be initialized with a value of 1.



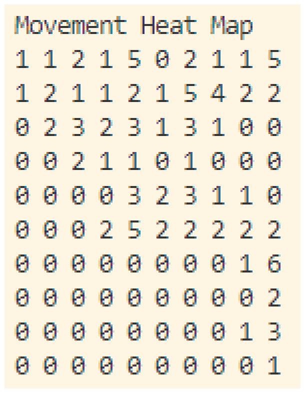
```
Starting Grid
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 1 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
```

Figure 1 Example of a 10x10 Two-Dimensional Grid

Movement Heat Map

Part of the deliverables of your simulation will be printing a movement heat map of the agent. You will need to keep track of the number of times the random agent has visited each grid square during the random walk. When the agent moves into a new square, you will need to increment the value stored in your nested list by 1. At the end of the simulation, you will need to print out the final tally of visits for all squares in the grid.

You will need to develop a print_grid() function that accepts a 2-dimensional nested list of any size and prints the values out in a nicely formatted grid. You will need to call this print_grid() function and pass your nested list as an argument at the end of your program to produce a movement heatmap similar to the one shown below in Figure 2.



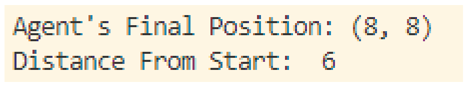
```
Movement Heat Map
1 1 2 1 5 0 2 1 1 5
1 2 1 1 2 1 5 4 2 2
0 2 3 2 3 1 3 1 0 0
0 0 2 1 1 0 1 0 0 0
0 0 0 0 3 2 3 1 1 0
0 0 0 2 5 2 2 2 2 2
0 0 0 0 0 0 0 0 1 6
0 0 0 0 0 0 0 0 0 2
0 0 0 0 0 0 0 0 1 3
0 0 0 0 0 0 0 0 0 1
```

Figure 2 Example of a Movement Heat Map for a 10x10 Grid

Agent Position

You will need to create a container to keep track of the position of the random agent during the simulation. Each time the agent moves, you will need to update their (x, y) position on the grid. Because the grid is finite, you will need to program logic into the autonomous agent's movement to prevent them from moving outside the grid. You may not allow the agent to move beyond the bounds of the 15x15 grid. If the agent's randomly generated moves them out of bounds, then your program should skip that move instead.

Your agent should start the simulation at the center of the grid in square (6, 6). At the end of the simulation, you will need to output the agent's final position after 100 moves and calculate the distance from the agent's final position and their starting position. The distance of the agent from their starting position should be reported using the Manhattan Distance. I.e., the difference between the indexes of the x and y positions.



```
Agent's Final Position: (8, 8)
Distance From Start:  6
```

Figure 3 Example of Agent's Final Position & Manhattan Distance

Agent Movement

Because the agent's movement will be entirely determined by the random number generator, you will need to initialize the random number generator with the following seed: **43543**. This will ensure that the results of the simulation will always be reproducible. Note: This is also how we will verify your program is correct!

You will use the **randInt()** function to generate random numbers between 1 and 20. The agent's movement based on the randomly generated number will be as follows:

- If 1 <= randInt() <= 7, then the agent will move up in the grid.
- If 8 <= randInt() <= 14, then the agent will move down in the grid.
- If 15 <= randInt() <= 17, then the agent will move left in the grid.
- If 18 <= randInt() <= 20, then the agent will move right in the grid.

If the agent's moves will put them out of bounds of the grid, the invalid moves should be skipped until a valid move is randomly selected.

Skills Covered

- Creating & Accessing Nested Lists

Deliverables

For this program you will need to provide the python file containing your code as well as a screenshot of the output of your program. Please name your files as follows:

- Python Files
    - lastname_firstname_filename.py
    - For example: **hawamdeh_faris_random_walk.py**
- Screenshots
    - lastname_firstname_filename.png
    - For example: **hawamdeh_faris_random_walk.png**