The University of Texas
Computer Science

313E Programming Assignment 1
2D Traversals (Python Review)

This assignment must be completed **individually**. Be sure to include your name and EID in the Python file header.

Complete these steps to prepare for the Assignment.

Check	Description
▢	Download runner.py and traversals.py. You will be working on the traversals.py file.
▢	Place both files in the same folder/directory.
▢	You may not change the file names. Otherwise, the grading script will not work.

Problem Description

Requirements

Write a program to traverse a 2D matrix using loops. Your function's return values must be a list, and the values in the list must be tuples of the coordinates (0 indexed). Your solution should traverse the matrix and find the current position's coordinates based on the given traversal. Specifically:

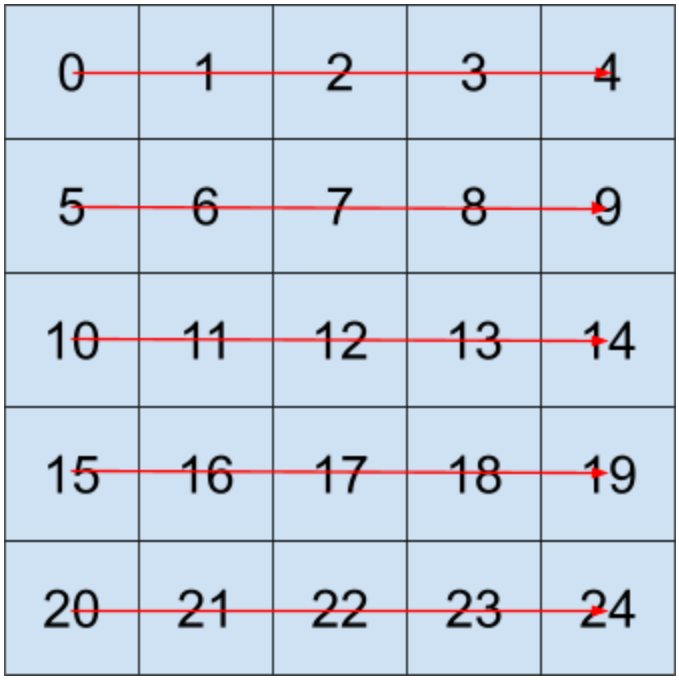
1. The functions should be implemented to handle matrices of sizes 1x1 up to 10x10.
2. The input of each function is a 2D matrix represented as a list of lists.
3. The output of each function should be a list of tuples representing the coordinates of the traversal.
4. **You may not change the names of the functions listed. They must have the functionality as given in the specifications. You can always add more functions than those listed.**
5. **You may not import any external libraries in your solution.**

There are seven traversals you will be implementing

Row-Major Traversal

Iterate from left to right, then top to bottom.

- Example coordinates:
  [(0, 0), (0, 1), (0, 2), (0, 3), (0, 4), (1, 0), (1, 1), (1, 2), (1, 3), (1, 4), (2, 0), (2, 1), (2, 2), (2, 3), (2, 4), (3, 0), (3, 1), (3, 2), (3, 3), (3, 4), (4, 0), (4, 1), (4, 2), (4, 3), (4, 4)]
- Example order:
  0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24



Column-Major Traversal

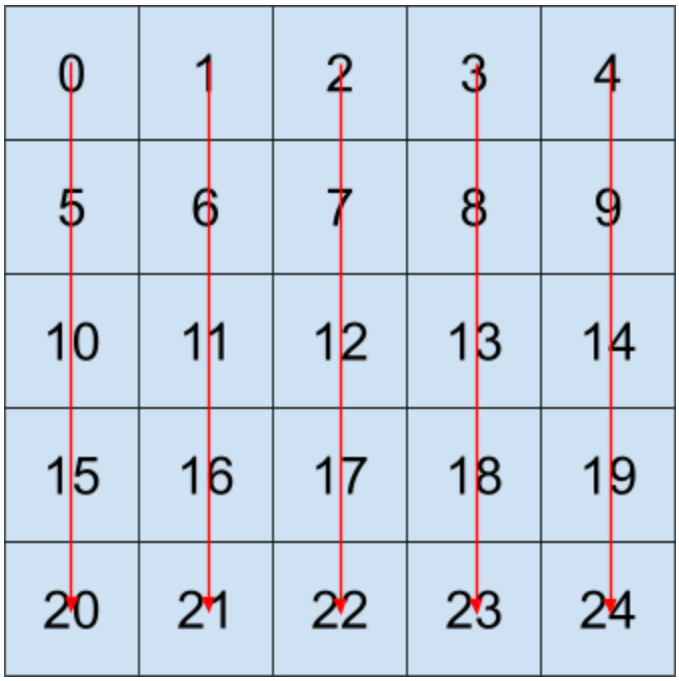
Iterate top to bottom, then left to right.

Example coordinates:

[(0, 0), (1, 0), (2, 0), (3, 0), (4, 0), (0, 1), (1, 1), (2, 1), (3, 1), (4, 1), (0, 2), (1, 2), (2, 2), (3, 2), (4, 2), (0, 3), (1, 3), (2, 3), (3, 3), (4, 3), (0, 4), (1, 4), (2, 4), (3, 4), (4, 4)]

Example order:

0 5 10 15 20 1 6 11 16 21 2 7 12 17 22 3 8 13 18 23 4 9 14 19 24



Row Zig-Zag Traversal

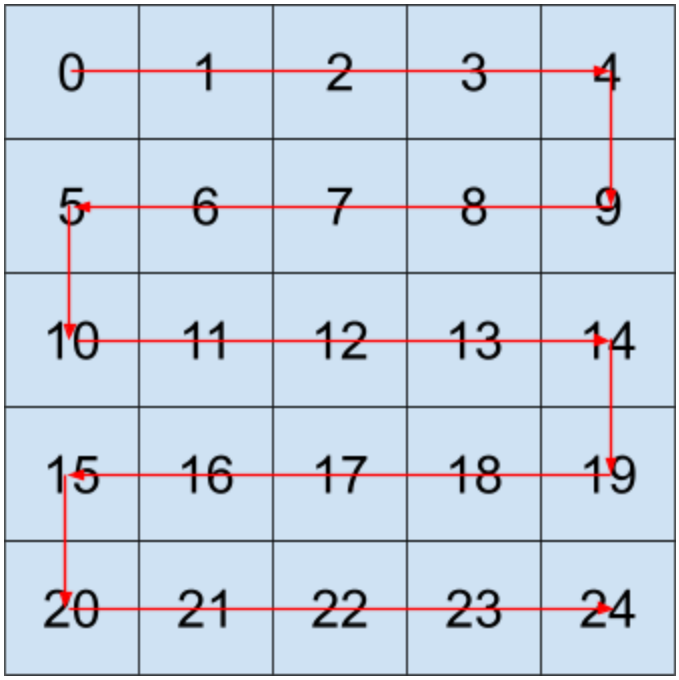
Alternate between iterating left to right and right to left, going from top to bottom.

Example coordinates:

[(0, 0), (0, 1), (0, 2), (0, 3), (0, 4), (1, 4), (1, 3), (1, 2), (1, 1), (1, 0), (2, 0), (2, 1), (2, 2), (2, 3), (2, 4), (3, 4), (3, 3), (3, 2), (3, 1), (3, 0), (4, 0), (4, 1), (4, 2), (4, 3), (4, 4)]

Example order:

0 1 2 3 4 9 8 7 6 5 10 11 12 13 14 19 18 17 16 15 20 21 22 23 24



Column Zig-Zag Traversal

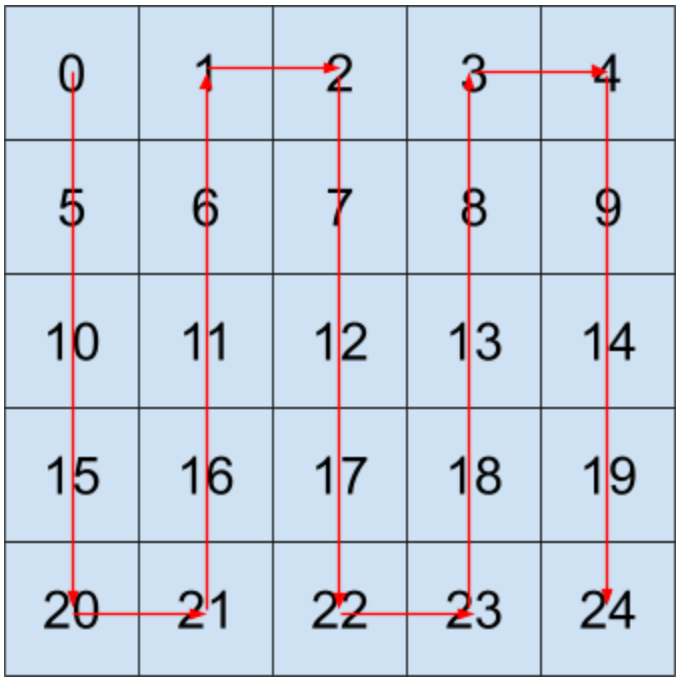
Alternate between iterating top to bottom and bottom to top, going from left to right.

Example coordinates:

[(0, 0), (1, 0), (2, 0), (3, 0), (4, 0), (4, 1), (3, 1), (2, 1), (1, 1), (0, 1), (0, 2), (1, 2), (2, 2), (3, 2), (4, 2), (4, 3), (3, 3), (2, 3), (1, 3), (0, 3), (0, 4), (1, 4), (2, 4), (3, 4), (4, 4)]

Example order:

0 5 10 15 20 21 16 11 6 1 2 7 12 17 22 23 18 13 8 3 4 9 14 19 24



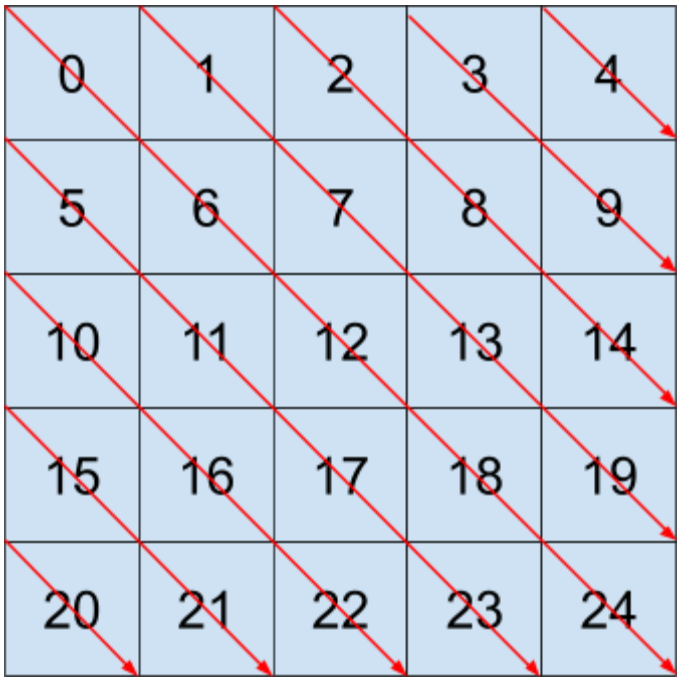
Main Diagonal Traversal

Iterate from the top-right to the bottom-left in the direction of the main diagonal.

Example coordinates:

[(0, 4), (0, 3), (1, 4), (0, 2), (1, 3), (2, 4), (0, 1), (1, 2), (2, 3), (3, 4), (0, 0), (1, 1), (2, 2), (3, 3), (4, 4), (1, 0), (2, 1), (3, 2), (4, 3), (2, 0), (3, 1), (4, 2), (3, 0), (4, 1), (4, 0)]

Example order: 4 3 9 2 8 14 1 7 13 19 0 6 12 18 24 5 11 17 23 10 16 22 15 21 20



Secondary Diagonal Traversal

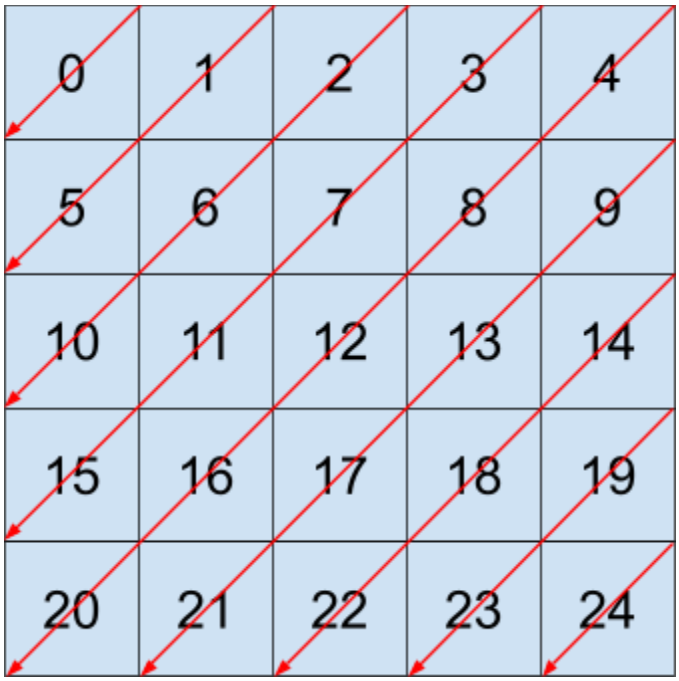
Iterate from the top-left to the bottom-right in the direction of the secondary diagonal.

Example coordinates:

[(0, 0), (0, 1), (1, 0), (0, 2), (1, 1), (2, 0), (0, 3), (1, 2), (2, 1), (3, 0), (0, 4), (1, 3), (2, 2), (3, 1), (4, 0), (1, 4), (2, 3), (3, 2), (4, 1), (2, 4), (3, 3), (4, 2), (3, 4), (4, 3), (4, 4)]

Example order:

0 1 5 2 6 10 3 7 11 15 4 8 12 16 20 9 13 17 21 14 18 22 19 23 24



Spiral Traversal

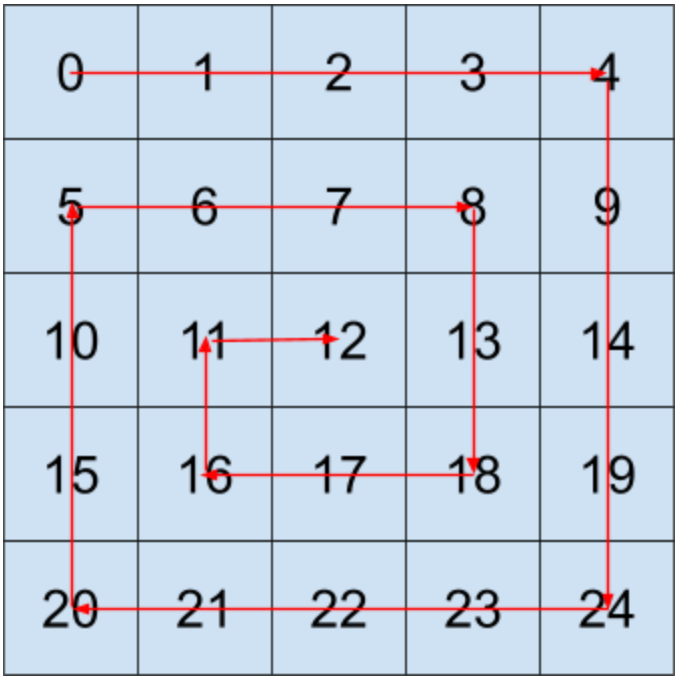
Iterate in a spiral order.

Example coordinates:

[(0, 0), (0, 1), (0, 2), (0, 3), (0, 4), (1, 4), (2, 4), (3, 4), (4, 4), (4, 3), (4, 2), (4, 1), (4, 0), (3, 0), (2, 0), (1, 0), (1, 1), (1, 2), (1, 3), (2, 3), (3, 3), (3, 2), (3, 1), (2, 1), (2, 2)]

Example order:

0 1 2 3 4 9 14 19 24 23 22 21 20 15 10 5 6 7 8 13 18 17 16 11 12



Input

This project uses `argparse` to pass in arguments to set up the grid. `argparse` is a Python module that enables you to pass in arguments using the command-line interface by specifying two dashes, then the argument, and then any possible values (if applicable). Here are the possible arguments you can use:

Argument	Possible Values	Default	Description
rows	1 - 10	5	number of rows in matrix
columns	1 - 10	5	number of columns in matrix
traversal	row, column, row-zigzag, column_zigzag, main, secondary, spiral	row	type of traversal
gui			if present, enables GUI mode. does not take any values
speed	50- 1000, in increments of 50	100	sets the speed of grid updates in milliseconds if using GUI
debug			if present, enables debugging prints. does not take any values

This command sets the matrix row, matrix column, traversal type, enables GUI mode, adjusts the speed, and enables debugging prints:

```
python3 runner.py --rows 5 --columns 5 --traversal spiral --gui --speed 100 --debug
```

For the simplest test:

```
python3 runner.py --traversal row
```

Output

This command is an example of what will be used in the test cases:

```
python3 runner.py --rows 5 --columns 5 --traversal row
```

Your output will look like the following:

```
[
        [   0   1   2   3   4 ]
        [   5   6   7   8   9 ]
        [  10  11  12  13  14 ]
        [  15  16  17  18  19 ]
        [  20  21  22  23  24 ]
]

Traversal: row

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
```

Each test case will print out the grid in this particular format, the traversal passed into argparse, and the grid values after indexing into the grid when performing the traversal.

Grading

Visible Test Cases - 75/100 points

The program output exactly matches the sample solution's output. To receive full credit, the program must produce the correct output based on all requirements in this document and pass all the test cases.

Hidden Test Cases - 15/100 points

The program output exactly matches the sample solution's output. To receive full credit, the program must produce the correct output based on all requirements in this document and pass all the test cases.

Style Grading - 10/100 points

The traversals.py file must comply with the PEP 8 Style Guide.

The TAs will complete a manual code review for each assignment to confirm that you have followed the requirements and directions on this document. Deductions will occur on each test case that fails to follow requirements.

Submission

Follow these steps for submission.

Check	Description
▢	Verify that the code does not have a compile or runtime error. A compile or runtime error will result in a 0 for the assignment as a whole.
▢	Verify that you have no debugging statements left in your code.
▢	Submit **only** traversals.py (the starter code file with your updates) to the given assignment in Gradescope. This will cause the grading scripts to run.
▢	When the grading scripts are complete, check the results. If there are errors, evaluate the script feedback, work on the fix, test the fix, and then re-submit the file to Gradescope. You can submit as many times as necessary before the due date. NOTE: By default, only the most recent submission will be considered for grading. If you want to use a previous submission for your final grade, you must activate it from your submission history before the due date.

Academic Integrity

Please review the Academic Integrity section of the syllabus. We will be using plagiarism checkers, and we will cross-reference your solution with AI-generated solutions to check for similarities. Remember the goal in this class is not to write the perfect solution; we already have many of those! The goal is to learn how to problem solve, so:

- don't hesitate to ask for guidance from the instructional staff.
- discuss with peers about **only** bugs, their potential fixes, design decisions, and project clarifying questions. Do not look at or copy another student's code when doing so.

Attribution

Thanks to Dr. Carol Ramsey and Dr. Kia Teymourian for the project templates.