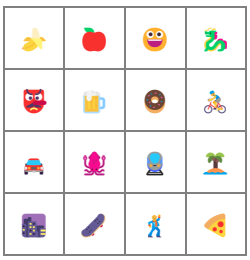
Challenges | Matrix (2D Array) Traversal



For each of the following challenges create a function/method that takes an m × n matrix (2D array) and returns an array containing a record of the given traversal. An approach might be to create separate module files (Python), or class files (Java) for each of the grouped challenges, and access these from a central file/method. Your solution[s] should be dynamic, in that they must be capable of taking any size matrix and use simple constructs that are common to all programming languages.

Row-wise traversal (row by row)

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16	4 3 2 1 8 7 6 5 12 11 10 9 16 15 14 13	16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1	13 14 15 16 9 10 11 12 5 6 7 8 1 2 3 4
Top-left to bottom-right	Top-right to bottom-left	Bottom-right to top-left	Bottom-left to top-right

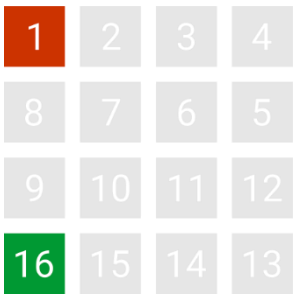
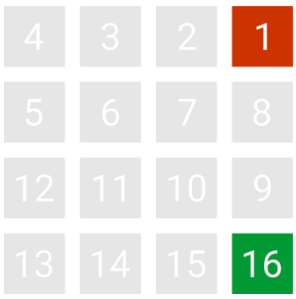
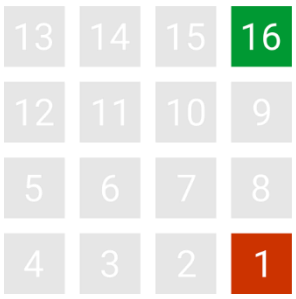
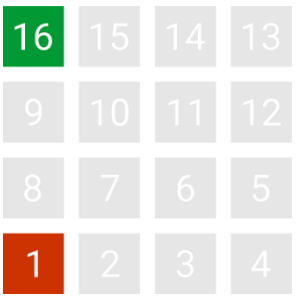
Column-wise traversal (column by column)

1 5 9 13 2 6 10 14 3 7 11 15 4 8 12 16	13 9 5 1 14 10 6 2 15 11 7 3 16 12 8 4	16 12 8 4 15 11 7 3 14 10 6 2 13 9 5 1	4 8 12 16 3 7 11 15 2 6 10 14 1 5 9 13
Top-left to bottom-right	Top-right to bottom-left	Bottom-right to top-left	Bottom-left to top-right

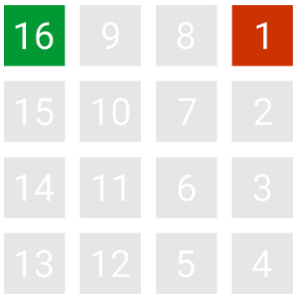
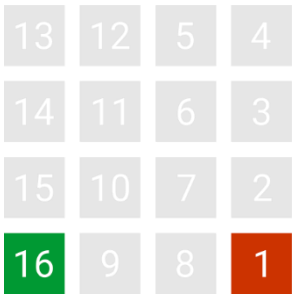
Diagonal traversal (corner to corner)

1 2 3 4	1 2 3 4	4 3 2 1	4 3 2 1
Top-left to bottom-right	Top-right to bottom-left	Bottom-right to top-left	Bottom-left to top-right

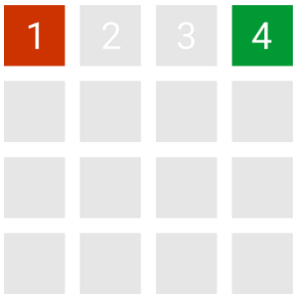
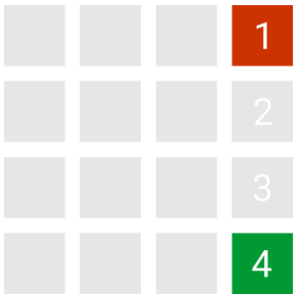
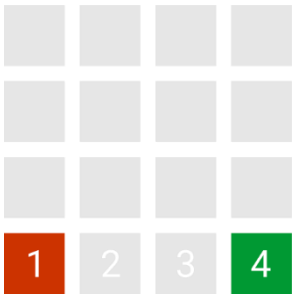
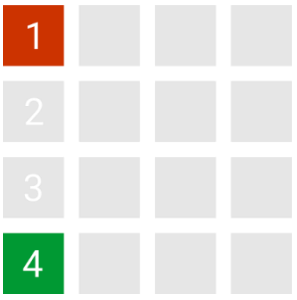
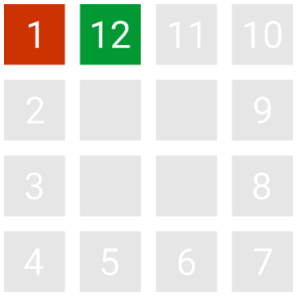
Horizontal snake traversal (row by row)

			
Top-left to bottom-left	Top-right to bottom-right	Bottom-right to top-right	Bottom-left to top-left

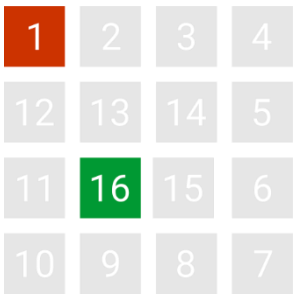
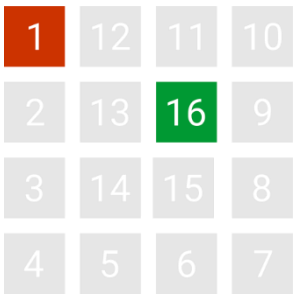
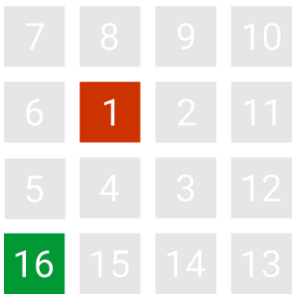
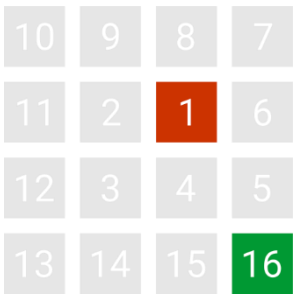
Vertical snake traversal (column by column)

			
Top-left to top-right	Top-left to top-right	Bottom-right to bottom-left	Bottom-left to top-left

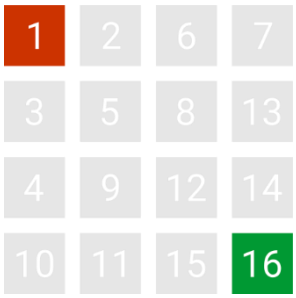
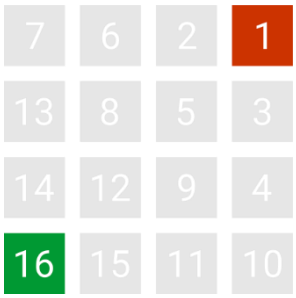
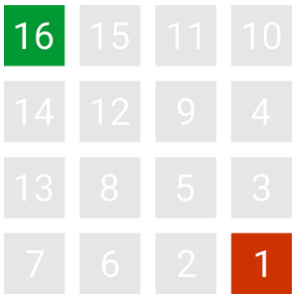
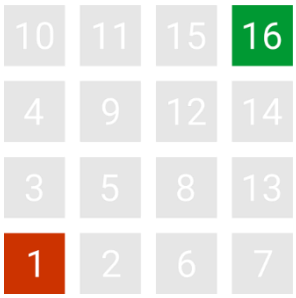
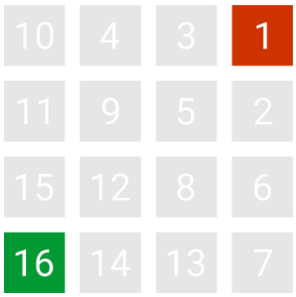
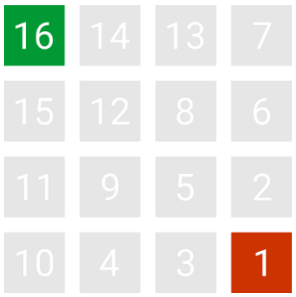
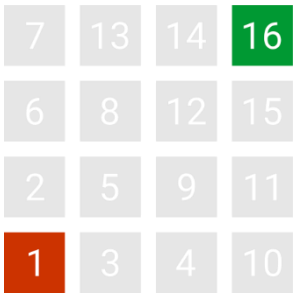
Boundary traversal

			
Top	Right	Bottom	Left
			
Clockwise	Anticlockwise		

Spiral traversal (column by column)

			
Outside in clockwise	Outside in anticlockwise	Inside out clockwise	Inside out anticlockwise

Zigzag traversal

			
Top-left to bottom-right, column first	Top-right to bottom-left, column first	Bottom-right to top-left, column first	Bottom-left to top-right, column first
			
Top-left to bottom-right, row first	Top-right to bottom-left, row first	Bottom-right to top-left, row first	Bottom-left to top-right, row first

Additional challenges (courtesy of ChatGPT)

- **Islands**: Given a matrix that represents the distribution of land and water in a given area, count the number of islands (connected land cells).
- **Adjacent submatrix**: Locate given element[s] in the matrix, and its adjacent elements.
- **Connected submatrix**: Locate given element[s] in the matrix, and elements deemed connected, for example odd or even.
- **Knight's Tour**: Given a chessboard matrix, show the path a given knight would take to visit every square on the board exactly once.
- **Knight's shortest path**: Extend the Knight's Tour problem by finding the shortest path from a starting position to an ending position on the chessboard.
- **Path**: Given matrix that represents a maze, find a path from the top-left to the bottom-right corner.
- **Unique paths**: Given matrix that represents a maze, find the number of unique paths from the top-left to the bottom-right corner, moving only right or down.
- **Magic square**: Determine if a given square matrix is a magic square, meaning the sum of each row, column, and diagonal is the same.
- **Multiplication**: Multiply two matrices.
- **Longest increasing path**: Given a matrix of numbers, find the longest path where a path is a sequence of cells with increasing values.
- **Sparse matrix compression**: Compress a sparse matrix (containing a sizeable number of zeros), converting it to a more memory-efficient representation.
- **Binary connectivity**: Given a matrix that represents a binary image, find the number of connected components of 1s.
- **Rotation**: Rotate a given matrix by 90 degrees clockwise or counterclockwise.