Daily Coding Problem #84

Problem

This problem was asked by Amazon.

Given a matrix of 1s and 0s, return the number of "islands" in the matrix. A 1 represents land and 0 represents water, so an island is a group of 1s that are neighboring whose perimeter is surrounded by water.

For example, this matrix has 4 islands.

```
1 0 0 0 0
0 0 1 1 0
0 1 1 0 0
0 0 0 0 0
1 1 0 0 1
1 1 0 0 1
```

Solution

This problem can be solved by keeping track of a `visited` table that keeps track of the land we've visited. Then, every time we see a piece of land that hasn't been visited, we can floodfill explore.

This takes O(N) (where N is the number of cells) since each cell is only visited twice: once in our outer for loop and then once in our `fill`.

```
def num_islands(board):
    num_rows = len(board)
    num_cols = len(board[0])
    count = 0

    visitied = [[False for _ in range(num_cols)] for _ in range(num_rows)]
    for row in range(len(board)):
        for col in range(len(board[row])):
            if board[row][col] == 1 and not visitied[row][col]:
                fill(board, visitied, row, col)
                count += 1
    return count

def fill(board, visitied, row, col):
    moves = [(0, 1),
             (0, -1),
             (1, 0),
             (-1, 0)]
    visitied[row][col] = True

    for move_row, move_col in moves:
```

```
        new_row, new_col = (row + move_row, col + move_col)
        if (inside_board(board, new_row, new_col) and
            board[new_row][new_col] == 1 and
            not visitied[new_row][new_col]):

            fill(board, visitied, new_row, new_col)

def inside_board(board, row, col):
    return 0 <= row < len(board) and 0 <= col < len(board[0])
```