Daily Coding Problem #168

Problem

This problem was asked by Facebook.

Given an N by N matrix, rotate it by 90 degrees clockwise.

For example, given the following matrix:

```
[[1, 2, 3],
 [4, 5, 6],
 [7, 8, 9]]
```

you should return:

```
[[7, 4, 1],
 [8, 5, 2],
 [9, 6, 3]]
```

Follow-up: What if you couldn't use any extra space?

Solution

Let's look at where each value should go with a few examples:

Consider the example input matrix:

- (0, 0) should go to (0, 2)
- (0, 1) should go to (1, 2)
- (0, 2) should go to (2, 2)
- (1, 0) should go to (0, 1)
- (1, 1) should go to (1, 1)
- (1, 2) should go to (2, 1)
- (2, 0) should go to (0, 0)
- (2, 1) should go to (1, 1)
- (2, 2) should go to (2, 1)

In general, the rule looks like this: `[i][j]` should go to `[j][n - i - 1]`. So we can instantiate a new matrix, loop over each element, and assign it to its new location:

```
def rotate_matrix(matrix):
    n = len(matrix)
```

```
    new_matrix = [[None for _ in range(n)] for _ in range(n)]

    for r, row in enumerate(matrix):
        for c, val in enumerate(row):
            new_matrix[c][n - r - 1] = val

    return new_matrix
```

This takes $O(n^2)$ time and space. How can we do this without using any extra memory?

It would be hard to perform a rotation by only swapping two elements. Instead, we can look at a value and perform a chain of 4 swaps:

- Top-left with bottom-left
- Top-right with top-left
- Bottom-right with top-right
- Bottom-left with bottom-right

We start with the first row and move down until `n // 2`, since the bottom rows should be rotated already by then.

```
def rotate_matrix(matrix):
    n = len(matrix)

    for i in range(n // 2):
        for j in range(i, n - i - 1):
            p1 = matrix[i][j]
            p2 = matrix[j][n - i - 1]
            p3 = matrix[n - i - 1][n - j - 1]
            p4 = matrix[n - j - 1][i]

            matrix[j][n - i - 1] = p1
            matrix[n - i - 1][n - j - 1] = p2

            matrix[n - j - 1][i] = p3
            matrix[i][j] = p4

    return matrix
```

While this still runs in $O(n^2)$ time, we use no extra space as everything is rotated in-place.