Daily Coding Problem #197

Problem

This problem was asked by Amazon.

Given an array and a number k that's smaller than the length of the array, rotate the array to the right k elements in-place.

Solution

One way to perform a circular shift in Python is to slice off the end of the array and attach it to the front:

```
def rotate(array, k):
    n = len(array) - k
    return array[n:] + array[:n]
```

However, this uses O(n) extra space, as we are creating a new array.

To solve this in place, note that we can think about the array above as consisting of two parts:

- the part before the slice, which we will call `A`, and
- the part after, which we will call `B`

We are looking for a way to turn `AB` into `BA`, but without actually slicing. Is this possible?

It turns out that it is! We can first reverse the elements in `A` to create `A'`, and reverse the elements in `B` to create `B'`. Finally, we can reverse the entire array, which will give us the desired result in O(n) time and no extra space.

To give a concrete example, let's say we want to rotate `[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]` by 3, which will have the same effect as slicing `[7, 8, 9]` and placing it in front. We first rotate the

individual parts, creating:

`[6, 5, 4, 3, 2, 1, 0, 9, 8, 7]`

Finally, we reverse the entire array, to create `[7, 8, 9, 0, 1, 2, 3, 4, 5, 6]`.

Here it is in code:

```
def reverse(array, left, right):
    while left < right:
        array[left], array[right] = array[right], array[left]
        left += 1
        right -= 1

def rotate(array, k):
    n = len(array)
    reverse(array, 0, n - k - 1)
    reverse(array, n - k, n - 1)
    reverse(array, 0, n - 1)
    return array
```