🎯 Question (LeetCode 189: Rotate Array)

Given an array `nums`, rotate the array to the **right by k steps**, where `k` is non-negative.

Example:

- Input: nums = [1,2,3,4,5,6,7], k = 3
- Output: [5,6,7,1,2,3,4]
- Input: nums = [-1,-100,3,99], k = 2
- Output: [3,99,-1,-100]

🏷️ Rotate Array

📁 1. Definition and Purpose

- Rotate array elements to the right by `k` steps in-place.
- Useful in cyclic data shifting, buffer rotation, and scheduling algorithms.

📁 2. Syntax and Structure (Python)

```
# nums: list of integers
# k: number of rotation steps
```

📁 3. Two Approaches

🧾 Approach 1: Brute Force

- Rotate the array one step at a time, k times.

```
def rotate_bruteforce(nums, k):
    n = len(nums)
    k %= n
    for _ in range(k):
        last = nums[-1]
        for i in range(n-1, 0, -1):
            nums[i] = nums[i-1]
        nums[0] = last
```

- **Time Complexity:** O(n*k)
- **Space Complexity:** O(1)

🧾 Approach 2: Optimized (Reverse Array Method)

- Reverse the whole array.
- Reverse first k elements.
- Reverse remaining n-k elements.
- O(1) extra space and O(n) time.

📂 4. Optimized Pseudocode

```
def reverse(nums, start, end):
    while start < end:
        nums[start], nums[end] = nums[end], nums[start]
        start += 1
        end -= 1

k = k % len(nums)
reverse(nums, 0, len(nums)-1)
reverse(nums, 0, k-1)
reverse(nums, k, len(nums)-1)
```

📂 5. Python Implementation with Detailed Comments

```python
def rotate(nums: list[int], k: int) -> None:
    """
    Rotate array to the right by k steps using reverse method.
    """
    n = len(nums)
    k %= n  # Handle cases where k > n

    def reverse(start: int, end: int) -> None:
        while start < end:
            nums[start], nums[end] = nums[end], nums[start]  # Swap elements
            start += 1
            end -= 1

    reverse(0, n-1)      # Reverse entire array
    reverse(0, k-1)      # Reverse first k elements
    reverse(k, n-1)      # Reverse remaining n-k elements

# Example Usage
nums = [1,2,3,4,5,6,7]
k = 3
rotate(nums, k)
print(nums)  # Output: [5,6,7,1,2,3,4]
```

📂 6. Internal Working

- Reverse entire array to bring last k elements to the front in reverse.
- Reverse subarrays to restore relative order.
- In-place, no extra memory required.

📂 7. Best Practices

- Always reduce k modulo n.
- Use reverse method for large arrays to save time.
- Avoid brute force rotation for large k.

📂 8. Related Concepts

- Array manipulation
- In-place reversal
- Cyclic rotations

📂 9. Complexity Analysis

- **Optimized Approach:**
  - Time: O(n)
  - Space: O(1)
- **Brute Force Approach:**
  - Time: O(n*k)
  - Space: O(1)

📂 10. Practice and Application

- LeetCode: 189 Rotate Array
- Useful in circular buffers, cyclic scheduling, and array-based games.