[Click on the logo to find **Problem Statement**]

Intuition

This code is designed to reverse a singly linked list in-place. It employs three pointers: `temp`, `before`, and `after`. The algorithm iterates through the linked list, reversing the direction of the pointers at each step. The `before` pointer is used to build the reversed portion of the list, while the `temp` pointer moves forward. The `after` pointer helps in preserving the remaining unreversed part of the list. The process continues until the end of the original list is reached. Finally, the head of the reversed list is set to the last node encountered (`before`). The approach ensures an efficient reversal of the linked list by iteratively adjusting pointers.

Approach

1. **Initialize Pointers:**
    - Initialize three pointers: `before` (initialized to `None`), `temp` (initialized to the head of the linked list), and `after` (initialized to `temp.next`).
2. **Iterative Reversal:**
    - Use a `while` loop to iterate through the list until `temp` becomes `None`.
    - Within the loop:
        - Update `after` to store the next node (`temp.next`) to prevent losing the reference.
        - Update `temp.next` to point back to the `before` node, reversing the link.
        - Move `before` and `temp` pointers one step forward. `before` becomes `temp`, and `temp` becomes `after`.
3. **Return the New Head:**
    - Once the loop exits (`temp` becomes `None`), `before` will be pointing to the new head of the reversed list.
    - Return `before` as the new head of the reversed linked list.

Complexity

- Time complexity: O(n)
- Space complexity: O(1)

Code

```
# Definition for singly-linked list.
# class ListNode:
#     def __init__(self, val=0, next=None):
#         self.val = val
#         self.next = next
class Solution:
    def reverseList(self, head: Optional[ListNode]) -> Optional[ListNode]:
        temp = head
        after = temp
        before = None
        while after is not None:
            after = temp.next
            temp.next = before
            before = temp
            temp =  after
        return before
```

If you want to see more solutions to coding problems, you can visit: