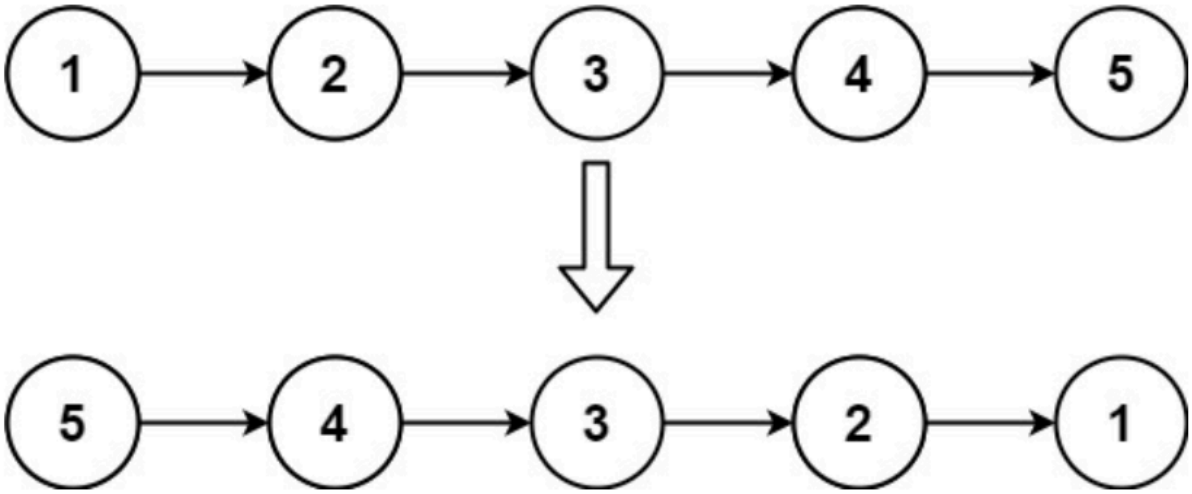
Problem Link:

https://leetcode.com/problems/reverse-linked-list/

Problem Description:

Given the `head` of a singly linked list, reverse the list, and return *the reversed list*.



Problem Approach:

Two pointers, basic logic and visualization.

Solution:

We start by initializing two pointers, previous and current. Current marks the head of the list, and current node of the iteration, and previous marks the element just before the current element. Hence, we initialize them by None and head respectively. We iterate through the complete list, hence, the statement: **while curr:** we take hold of a temporary pointer: next, that stores the address of the next node to the current element, so that we don't break the list during the reversal process. Then, we assign the current pointer's next pointer as previous (Rotating the pointing arrow), then, we move the previous pointer to the current pointer, and the current to the next pointer. Once we're done with the list, the previous pointer would be the head of the new list, and we return it.

Code (Python):

```
def reverseList(self, head: Optional[ListNode]) -> Optional[ListNode]:
        prev = None
        curr = head
        while curr:
            nxt = curr.next
            curr.next = prev
            prev = curr
            curr = nxt
        return prev
```