Reverse Nodes in k-Group

Question: Given a linked list, reverse the nodes of a linked list k at a time and return its modified list. If the number of nodes is not a multiple of k then left-out nodes in the end should remain as it is. You may not alter the values in the nodes, only nodes itself may be changed. Only constant memory is allowed.

For example,

Given this linked list: 1->2->3->4->5

For k = 2, you should return: 2->1->4->3->5

For k = 3, you should return: 3->2->1->4->5.

Solutions:

```
class ListNode(object):
    def __init__(self, x):
        self.val = x
        self.next = None


    def to_list(self):
        return [self.val] + self.next.to_list() if self.next else [self.val]


class Solution(object):
    def reverseKGroup(self, head, k):
        """
        :type head: ListNode
        :type k: int
        :rtype: ListNode
        """
        if not head or k <= 1:
            return head
    dummy = ListNode(-1)
    dummy.next = head
    temp = dummy
    while temp:
        temp = self.reverseNextK(temp, k)
    return dummy.next

def reverseNextK(self, head, k):
    # Check if there are k nodes left
    temp = head
    for i in range(k):
        if not temp.next:
            return None
        temp = temp.next

    # The last node when the k nodes reversed
    node = head.next
    prev = head
    curr = head.next
    # Reverse k nodes
    for i in range(k):
        nextNode = curr.next
        curr.next = prev
        prev = curr
        curr = nextNode
    # Connect with head and tail
    node.next = curr
    head.next = prev
```

```
        return node

if __name__ == "__main__":
    n1 = ListNode(1)
    n2 = ListNode(2)
    n3 = ListNode(3)
    n4 = ListNode(4)
    n5 = ListNode(5)
    n1.next = n2
    n2.next = n3
    n3.next = n4
    n4.next = n5
    r = Solution().reverseKGroup(n1, 3)
    print ( r.to_list() )
```