876. Middle of the Linked List

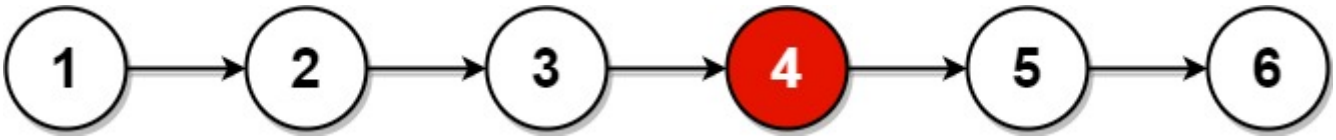
Given the `head` of a singly linked list, return *the middle node of the linked list*.

If there are two middle nodes, return **the second middle** node.

Example 1:


```
Input: head = [1,2,3,4,5]
Output: [3,4,5]
Explanation: The middle node of the list is node 3.
```

Example 2:




```
Input: head = [1,2,3,4,5,6]
Output: [4,5,6]
Explanation: Since the list has two middle nodes with values 3 and 4, we
return the second one.
```

Constraints:

- The number of nodes in the list is in the range `[1, 100]`.
- `1 <= Node.val <= 100`
- ```
  # Definition for singly-linked list.
  # class ListNode:
  #     def __init__(self, val=0, next=None):
  #         self.val = val
  #         self.next = next
  class Solution:
      def middleNode(self, head: Optional[ListNode]) ->
  Optional[ListNode]:

          if head is None or head.next is None:
              return head
  ```
  ```

```
        slow = head
        fast = head

        while fast.next is not None and fast.next.next is not None:
            slow = slow.next
            fast = fast.next.next


        if fast.next is None:
            return slow
        elif fast.next.next is None:
            return slow.next
```