Linked List Cycle II

Question: Given a linked list, return the node where the cycle begins. If there is no cycle, return null.

Solutions:

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


class Solution:
    """
    @param head: The first node of the linked list.
    @return: the node where the cycle begins. If there is no cycle, return null
    """
    def detectCycle(self, head):
        if head == None or head.next == None:
            return None
        slow = fast = head
        while fast and fast.next:
            slow = slow.next
            fast = fast.next.next
            if fast == slow:
                break
        if slow == fast:
```

```
            slow = head
            while slow != fast:
                slow = slow.next
                fast = fast.next
            return slow
        return None

if __name__ == '__main__':
    ll, ll.next, ll.next.next = ListNode(2), ListNode(4), ListNode(3),
    ll.next.next.next = ll.next
    print( Solution().detectCycle(ll).val )
```