Problem 2: Given *head*, the head of a linked list, determine if the linked list has a cycle in it. There is a cycle in a linked list if there is some node in the list that can be reached again by continuously following the next pointer.

Return *true* if there is a cycle in the linked list. Otherwise, return *false*.

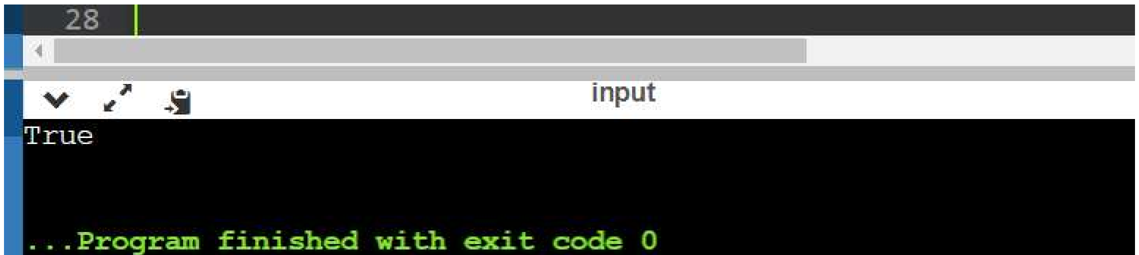
```
class ListNode:
    def __init_(self, val=0, next=None):
        self.val = val
        self.next = next


def hasCycle(head):
    if not head or not head.next:
        return False


    slow =  head
    fast = head.next


    while slow != fast:
        if not fast or not fast.next:
            return False
        slow = slow.next
        fast = fast.next.next
    return  True
head = ListNode(1)
head.next = ListNode(2)
head.next.next = ListNode(3)
head.next.next.next = ListNode(4)
head.next.next.next.next = head.next


has_cycle = hasCycle(head)
print(has_cycle)
```



```
True


...Program finished with exit code 0
```