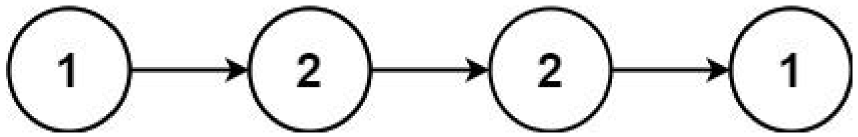
Palindrome Linked List

Question:

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

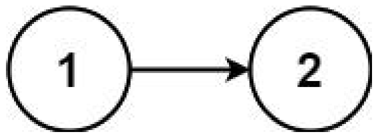
Given the head of a singly linked list, return true if it is a palindrome.

Example 1:



```
Input: head = [1,2,2,1]
Output: true
```

Example 2:



```
Input: head = [1,2]
Output: false
```

Approach 1:

Instead of manipulating the linked list, I just stored the values of the linked list in an array/list and just checked for palindrome directly.
I thought of using a string but since strings are immutable in python, it would require excess memory for inserting and slicing.

Solution 1:

https://gist.github.com/vermaayush680/a8cbeae1968b6f6c325ee9e2102aff1e

```
class Solution:
        def isPalindrome(self, head: Optional[ListNode]) -> bool:
                l=[]
                while head:
                        l+=[str(head.val)]
                        head = head.next
                return l==l[::-1]
```

Approach 2:

After going through the discussions forum, I devised this solution.
I basically traversed half the list by using two pointers: slow and head.
Then reversed the second half and compared it with the first half to see if they are equal or not.

Solution 2:

https://gist.github.com/vermaayush680/2c2b87e18832c5cafab84ab46090c613

```
class Solution:
        def isPalindrome(self, head: Optional[ListNode]) -> bool:
                fast=slow=head
                while fast and fast.next:
                        fast=fast.next.next
                        slow=slow.next

                node = None
                while slow:
                        nxt = slow.next
                        slow.next=node
```

```
            node=slow
            slow=nxt

        while node:
            if node.val != head.val:
                return False
            node=node.next
            head=head.next
        return True
```

Approach 3:

Joining the reversing and traversing the second part, we can reverse the first half while finding the middle and then compare this reverse with the second half.
This removes the need for the middle loop and the reversing and traversing are in the first loop itself.

Solution 3:

https://gist.github.com/vermaayush680/32a9613450d01bb2d3e524c52951b2a8

```
class Solution:
    def isPalindrome(self, head: Optional[ListNode]) -> bool:
        rv=None
        fst=slw=head
        while fst and fst.next:
            fst=fst.next.next
            rv,rv.next,slw=slw,rv,slw.next
        if fst:slw=slw.next
        while rv and rv.val==slw.val:
            rv=rv.next
            slw=slw.next
        return not rv
```