Remove loop in Linked List

You are given a linked list of **N** nodes. Remove the loop from the linked list, if present.
Note: Xis the position of the node to which the last node is connected to. If it is 0, then there is no loop.

Example 1:

Input: N = 3
value[] = {1,3,4}
X = 2
Output: 1 **Explanation:** The link list looks like 1 -> 3 -> 4
^ |
|____|
A loop is present. If you remove it
successfully, the answer will be 1.

Example 2:

Input: N = 4
value[] = {1,8,3,4}
X = 0
Output: 1 **Explanation:** The Linked list does not
contains any loop.

Your Task:
You don't need to read input or print anything. Your task is to complete the function **removeLoop**()
which takes the head of the linked list as input parameter. Simply remove the loop in the list (if present)
without disconnecting any nodes from the list. The driver code will print **1 **if your code is correct.

Expected time complexity : O(n)
**Expected auxiliary space : **O(1)

Constraints:
1 ≤ N ≤ 104

```
def removeLoop(self, head):
        # code here
        # remove the loop without losing any nodes
        if head == None:
            return
        if head.next == head:
            head.next = None
```

```
            return None
        slow = head
        fast = head
        while slow and fast:
            slow = slow.next
            fast = fast.next
            if fast == None:
                break
            fast = fast.next
            if slow == fast:
                break
        if fast==None:
            return
        else:
            slow = head
            //For circular linked list
            if slow==fast:
                fast = head
                while fast.next!=slow:
                    fast = fast.next
                fast.next = None
                return
            else:
                # prev = None
                while slow.next!=fast.next:
                    slow = slow.next
                    prev = fast
                    fast = fast.next
                fast.next = None
                return
```