# Detect Loop in linked list

Given a linked list of **N** nodes. The task is to check if the linked list has a loop. Linked list can contain self loop.

**Example 1:**

**Input:** N = 3
value[] = {1,3,4}
x = 2
**Output:** True **Explanation:** In above test case N = 3. The linked list with nodes N = 3 is given. Then value of x=2 is given which means last node is connected with xth node of linked list. Therefore, there exists a loop.

**Example 2:**

**Input:** N = 4
value[] = {1,8,3,4}
x = 0
**Output:** False **Explanation:** For N = 4 ,x = 0 means then lastNode->next = NULL, then the Linked list does not contains any loop.

**Your Task:**
The task is to complete the function **detectloop**() which contains reference to the head as only argument. This function should return 1 if linked list contains loop, else return 0.

**Expected Time Complexity:** O(N)
**Expected Auxiliary Space:** O(1)

**Constraints:**
1 <= N <= 104
1 <= Data on Node <= 103

```
def detectLoop(self, head):
        #code here
        slow = head
        fast = head
```

```
        while slow!=None and fast.next!=None and fast.next.next!=None:
            slow = slow.next
            fast = fast.next.next
            if slow==fast:
                return True
        return False
```