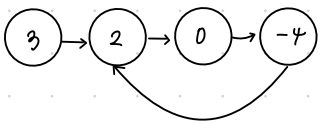
Linked List Cycle

: 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. Internally, pos is used to denote the index of the node that tail's next pointer is connected.

Ex)



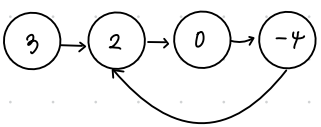
Input: head = [3, 2, 0, -4], pos = 1

Output: true

Idea:

① Check head is None or head.next is None

②



1. Create visited : Memory : O(n)
2. While current node's value is not null,
   if ∃ a same value with current node's value in visited, return True.

Approach 2 : Memory usage: O(1)

Intuition : Imagine two runners running on a track at different speed.

If there is no cycle in the list, the fast pointer will eventually reach the end

If there is a cycle, two runners will eventually meet.