Detect Cycle in a Linked List

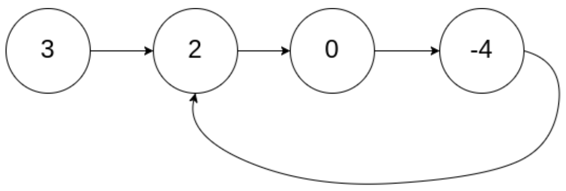
Difficulty	Easy
Category	LinkedList
Question	https://leetcode.com/problems/linked-list-cycle/
Solution	https://youtu.be/gBTe7lFR3vc
Status	Done

Question

> 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 to. **Note that `pos` is not passed as a parameter**.
>
> Return `true` *if there is a cycle in the linked list*. Otherwise, return `false`.

Example

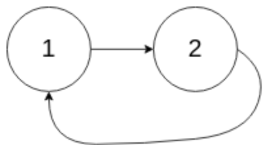
Example 1:



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

```
Explanation: There is a cycle in the linked list, where the tail connects to the 1st node (0-ind
exed).
```

Example 2:



```
Input: head = [1,2], pos = 0
Output: true
Explanation: There is a cycle in the linked list, where the tail connects to the 0th node.
```

Example 3:

```
Input: head = [1], pos = -1
Output: false
Explanation: There is no cycle in the linked list.
```

Idea

💡 Slow-Fast (**Floyd's Tortoise and Hare**) detection method. Fast is double speed, if they meet each other, meaning there’s a loop

Solution

```
class Solution:
    def hasCycle(self, head: Optional[ListNode]) -> bool:
        # Initialize two pointers, 'slow' and 'fast', both starting at the head of the linked list.
        slow, fast = head, head

        # Traverse the linked list using 'fast' pointer, moving 2 steps at a time,
        # and 'slow' pointer, moving 1 step at a time.
        while fast and fast.next:
```

```python
            slow = slow.next  # Move 'slow' one step
            fast = fast.next.next  # Move 'fast' two steps

            # If there is a cycle, at some point, 'slow' and 'fast' will meet at the same node.
            if slow == fast:
                return True  # Return True, indicating the presence of a cycle

        # If 'fast' reaches the end of the list (i.e., 'fast' or 'fast.next' becomes None),
        # it means there is no cycle, and we return False.
        return False  # Return False, indicating no cycle was found.
```