环形链表 II

给定一个链表，返回链表开始入环的第一个节点。 如果链表无环，则返回 `null`。

如果链表中有某个节点，可以通过连续跟踪 `next` 指针再次到达，则链表中存在环。 为了表示给定链表中的环，评测系统内部使用整数 `pos` 来表示链表尾连接到链表中的位置（索引从 0 开始）。如果 `pos` 是 `-1`，则在该链表中没有环。**注意：pos 不作为参数进行传递**，仅仅是为了标识链表的实际情况。

不允许修改 链表。

示例 1：

输入：`head = [3,2,0,-4], pos = 1`

输出：返回索引为 `1` 的链表节点

解释：链表中有一个环，其尾部连接到第二个节点。

示例 2：

输入：`head = [1,2], pos = 0`

输出：返回索引为 `0` 的链表节点

解释：链表中有一个环，其尾部连接到第一个节点。

示例 3：

输入：`head = [1], pos = -1`

输出：返回 `null`

解释：链表中没有环。

```
# Definition for singly-linked list.
# class ListNode:
#     def __init__(self, x):
#         self.val = x
#         self.next = None
```

```python
class Solution:
    def detectCycle(self, head: ListNode) -> ListNode:
        #快慢指针
        '''
        (1)在环形链表中按照快慢指针 找到相交节点（必然会有相遇）
        （2）一个从头节点开始另外一个从相交节点开始 再次相交即为入环节点
        '''
        fast=head
        slow=head
        while(fast and fast.next):
            fast=fast.next.next
            slow=slow.next
            if slow==fast:
                newnode=fast
                slow=head
                while slow!=newnode:
                    slow=slow.next
                    newnode=newnode.next
                return newnode
        return None
```