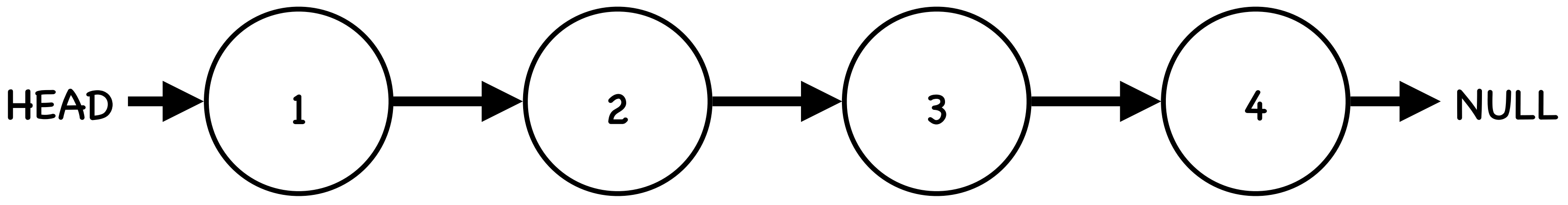
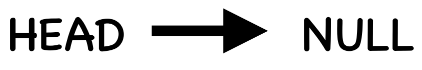
876. Middle of the Linked List



In this case the middle node doesn't exist so
the one we return is the middle plus one so
the middle nade is the one with the value 3

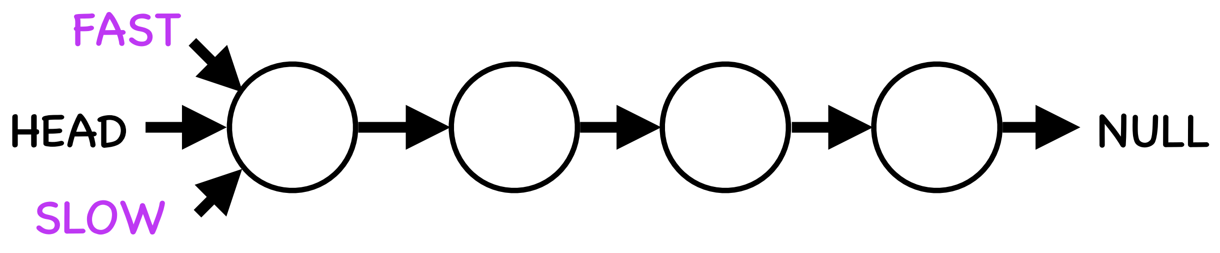
First we have to check if head points to not null value, if it does return null.
Unlike python in Java and other programming languages we would write something like if (head == null) return null;

```
if not head:
    return None
```



If not we'll use the Floyd's algorithm known as well as slow and fast pointers technique. First we'll create two pointers that point to the node that head is pointing at

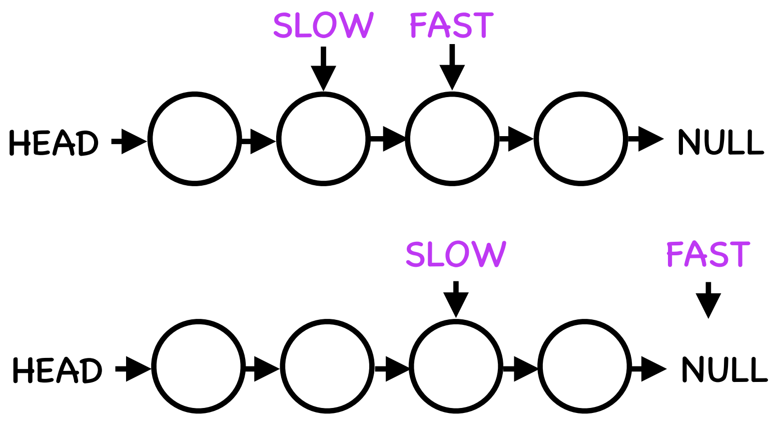
```
slow = head
fast = head
```



We'll advance slow pointer one step and faster point two steps each time while fast and fast.next is different of null and we'll return the node that slow pointer is pointing at.
In Java we would write something like while (fast != null && fast.next != null)

```
while fast and fast.next:
    slow = slow.next
    fast = fast.next.next

return slow
```



We go through the list once and the time complexity is O(n)