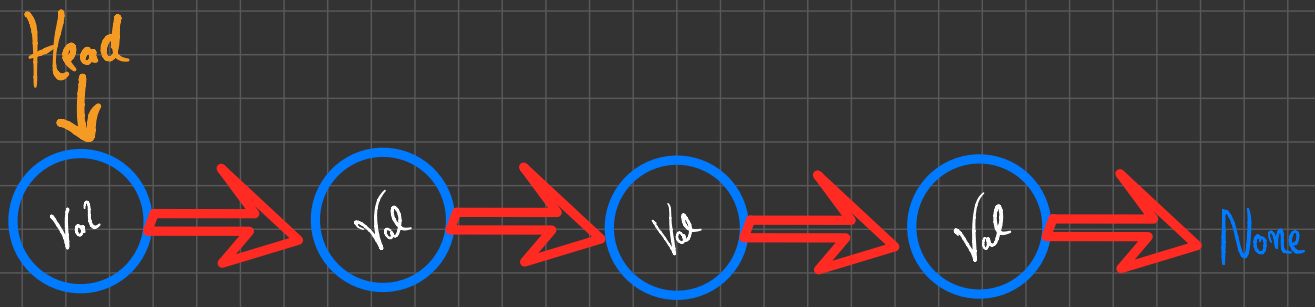
Leetcode
Problem 86: Partition List

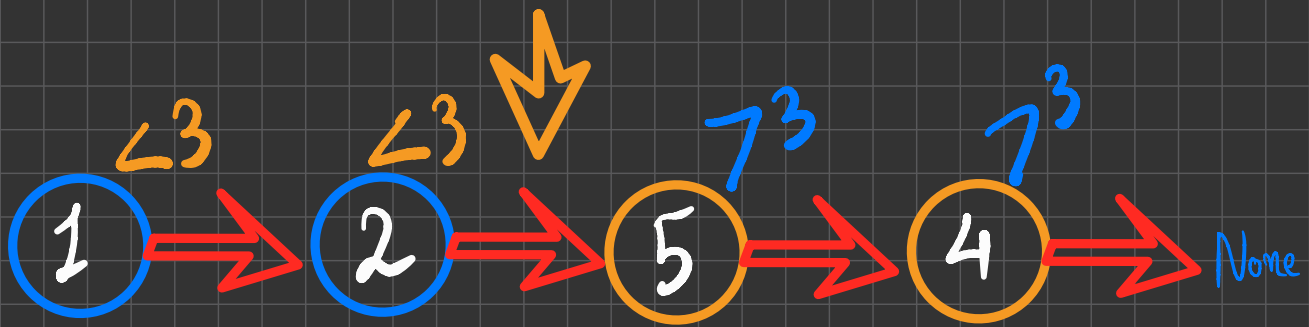
Problem Statement

Given the `head` of a linked list and a value `x`, partition it such that all nodes **less than** `x` come before nodes **greater than or equal** to `x`.

Condition

You should **preserve** the original relative order of the nodes in each of the two partitions.





Solution:

We will initiate two linked lists which will contain node(s) with value(s) lesser or more than and equal to x respectively. Lastly, we will combine both linked lists together to end up with expected output.

Code:

```
class ListNode:
    def __init__(self, val=0, next=None):
        self.val = val
        self.next = next
```

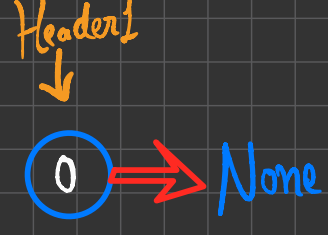
fundamental class for singly linked list

Line defines class Solution which acts as a container for the method 'partition'

```
class Solution:
    def partition(self, head: Optional[ListNode], x: int)
                                    -> Optional[ListNode]:
```

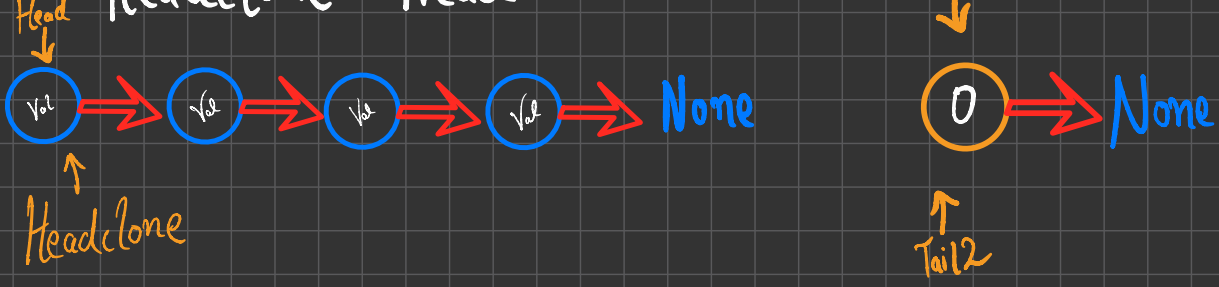
Creates 2 dummy nodes initialised with value of '0'
List 1 holds node(s) with value < x
List 2 holds node(s) with value >= x

```
        header1 = ListNode(0)
        header2 = ListNode(0)
        tail1 = header1
        tail2 = header2
```



Initialise a linked list traversal pointer

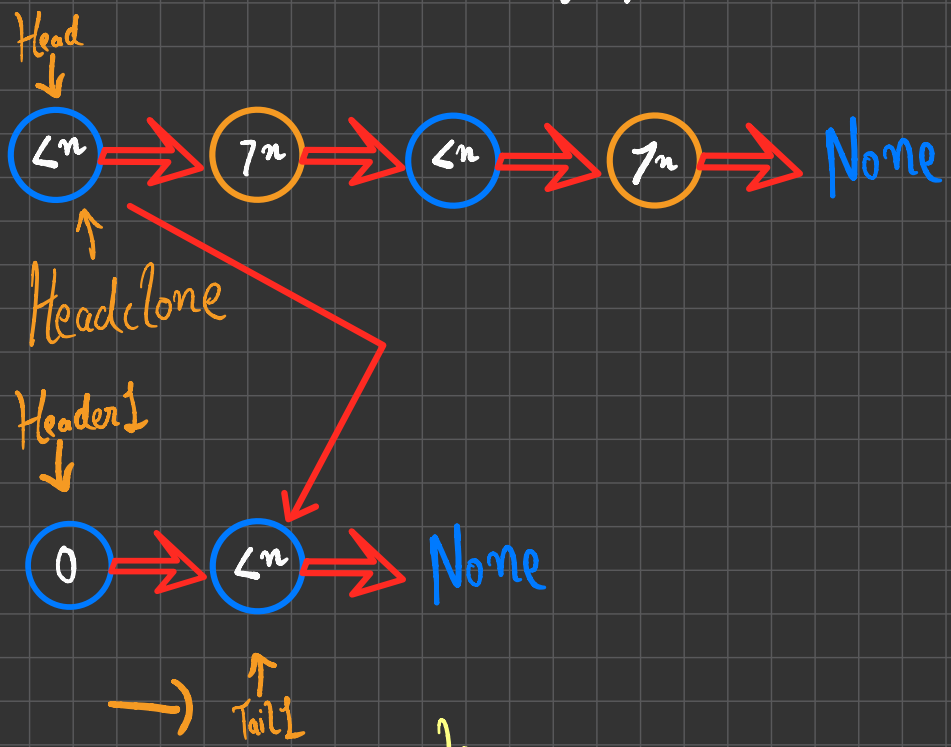
```
        headClone = head
```



Initiate a loop until headclone is not 'None'

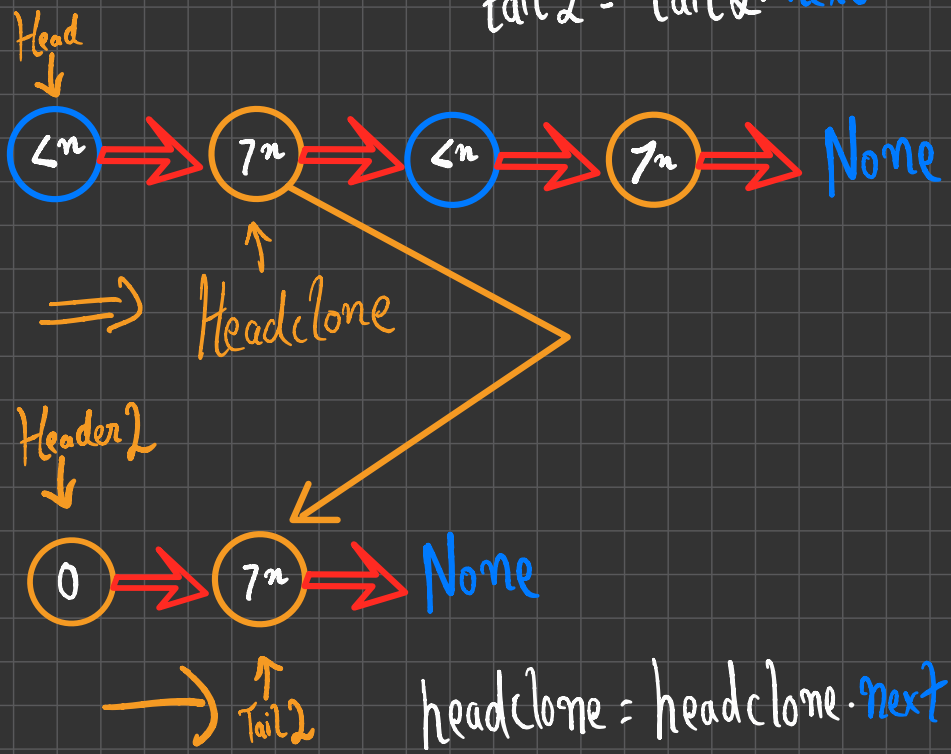
```
while headclone:
    if headclone.val < x:
        tail1.next = headclone
        tail1 = tail1.next
```

If value of node is less than x, add it to the end of list1. Move tail1 to the newly added node.




```
else:
    tail2.next = headclone
    tail2 = tail2.next
```

If value of node is more than x, add it to the end of list2. Move tail2 to the newly added node.



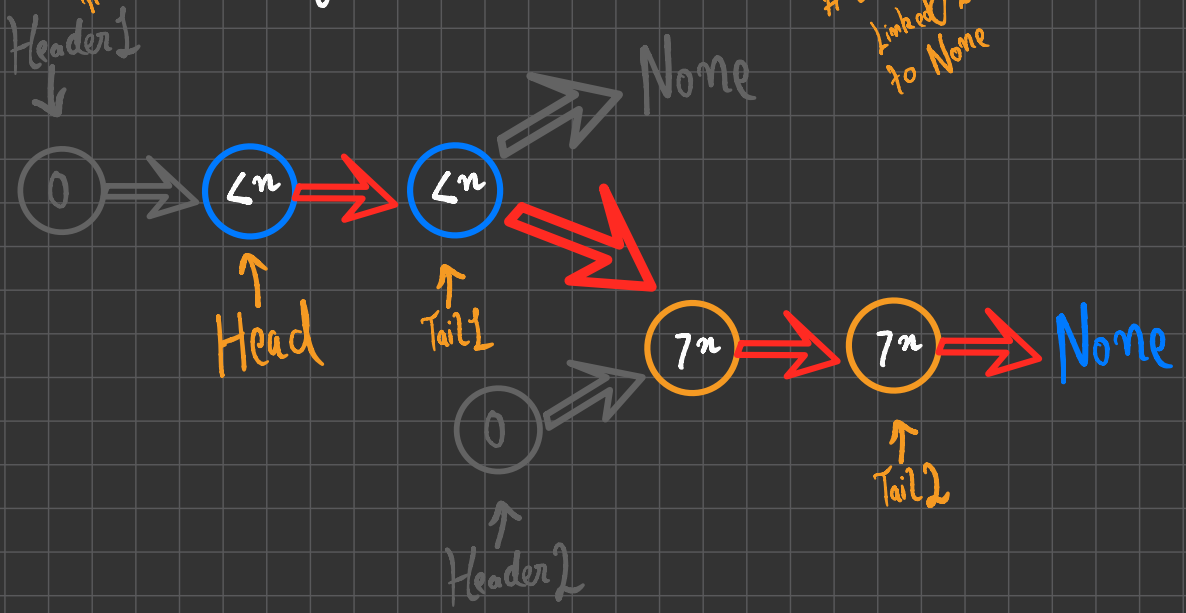

```
headclone = headclone.next
```

Attach List 1's end's to the actual starting node of List 2

tail1.next = header2.next

tail2.next = None

Correctly end the concatenated Linked List by pointing tail 2 to None



self.head = header1.next

return header1.next

```python
# Definition for singly-linked list.
class ListNode:
    def __init__(self, val=0, next=None):
        self.val = val
        self.next = next
class Solution:
    def partition(self, head: Optional[ListNode], x: int) -> Optional[ListNode]:
        header1, header2 = ListNode(0), ListNode(0)
        tail1, tail2 = header1, header2

        headclone = head
        while headclone:
            if headclone.val < x:
                tail1.next = headclone
                tail1 = tail1.next

            else:
                tail2.next = headclone
                tail2 = tail2.next
            headclone = headclone.next

        tail1.next = header2.next
        tail2.next = None
        self.head = header1.next
        return header1.next
```