Linked List Reversal - SOLUTION

Problem

Write a function to reverse a Linked List in place. The function will take in the head of the list as input and return the new head of the list.

You are given the example Linked List Node class:

In [1]:
```
class Node(object):

    def __init__(self,value):

        self.value = value
        self.nextnode = None
```

Solution

Since we want to do this in place we want to make the funciton operate in O(1) space, meaning we don't want to create a new list, so we will simply use the current nodes! Time wise, we can perform the reversal in O(n) time.

We can reverse the list by changing the next pointer of each node. Each node's next pointer should point to the previous node.

In one pass from head to tail of our input list, we will point each node's next pointer to the previous element.

Make sure to copy current.next_node into next_node **before** setting current.next_node to previous. Let's see this solution coded out:

In [2]:
```
def reverse(head):

    # Set up current,previous, and next nodes
    current = head
    previous = None
    nextnode = None

    # until we have gone through to the end of the list
    while current:

        # Make sure to copy the current nodes next node to a variable next_node
        # Before overwriting as the previous node for reversal
        nextnode = current.nextnode

        # Reverse the pointer ot the next_node
        current.nextnode = previous

        # Go one forward in the list
        previous = current
        current = nextnode
```

```
    return previous
```

Test Your Solution

You should be able to easily test your own solution to make sure it works. Given the short list a,b,c,d with values 1,2,3,4. Check the effect of your reverse function and maek sure the results match the logic here below:

In [3]:
```
# Create a list of 4 nodes
a = Node(1)
b = Node(2)
c = Node(3)
d = Node(4)

# Set up order a,b,c,d with values 1,2,3,4
a.nextnode = b
b.nextnode = c
c.nextnode = d
```

Now let's check the values of the nodes coming after a, b and c:

In [4]:
```
print(a.nextnode.value)
print(b.nextnode.value)
print(c.nextnode.value)
```

```
2
3
4
```

In [5]:
```
d.nextnode.value
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-5-9ee2d814a788> in <module>
----> 1 d.nextnode.value

AttributeError: 'NoneType' object has no attribute 'value'
```

So far so good. Note how there is no value proceeding the last node, this makes sense! Now let's reverse the linked list, we should see the opposite order of values!

In [6]:
```
reverse(a)
```

Out[6]:
```
<__main__.Node at 0x7f4e3414f730>
```

In [7]:
```
print(d.nextnode.value)
print(c.nextnode.value)
print(b.nextnode.value)
```

```
3
2
1
```

In [8]:
```
print(a.nextnode.value) # This will give an error since it now points to None
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-8-14ba01ce8725> in <module>
----> 1 print(a.nextnode.value) # This will give an error since it now points to None

AttributeError: 'NoneType' object has no attribute 'value'
```

Great, now we can see that each of the values points to its previous value (although now that the linked list is reversed we can see the ordering has also reversed)

Good Job!