Reverse a Linked List

May 3, 2020

1 Reversing a linked list exercise

Given a singly linked list, return another linked list that is the reverse of the first.

In [2]:
```
# Helper Code

class Node:
    def __init__(self, value):
        self.value = value
        self.next = None

class LinkedList:
    def __init__(self):
        self.head = None

    def append(self, value):
        if self.head is None:
            self.head = Node(value)
            return

        node = self.head
        while node.next:
            node = node.next

        node.next = Node(value)

    def __iter__(self):
        node = self.head
        while node:
            yield node.value
            node = node.next

    def __repr__(self):
        return str([v for v in self])
```

1.0.1 Write the function definition here

In [3]:
```
# Solution

    # Time complexity O(N)
    def reverse(linked_list):
        """
        Reverse the inputted linked list

        Args:
           linked_list(obj): Linked List to be reversed
        Returns:
           obj: Reveresed Linked List
        """
        new_list = LinkedList()

        prev_node = None

        """
        A simple idea - Pick a node from the original linked list traversing form the beginn
        prepend it to the new linked list.
        We have to use a loop to iterate over the nodes of original linked list
        """
        # In this "for" loop, the "value" is just a variable whose value will be updated in
        for value in linked_list:
            # create a new node
            new_node = Node(value)

            # Make the new_node.next point to the
            # node created in previous iteration
            new_node.next = prev_node

            # This is the last statement of the loop
            # Mark the current new node as the "prev_node" for next iteration
            prev_node = new_node

        # Update the new_list.head to point to the final node that came out of the loop
        new_list.head = prev_node

        return new_list
```

1.0.2 Let's test your function

In [4]:
```
llist = LinkedList()
for value in [4,2,5,1,-3,0]:
    llist.append(value)

flipped = reverse(llist)
```

```
        is_correct = list(flipped) == list([0,-3,1,5,2,4]) and list(llist) == list(reverse(flipp
        print("Pass" if is_correct else "Fail")
```

```
Pass
```

Show Solution

In [ ]:

In [ ]:
```
# Solution

        # Time complexity O(N)
        def reverse(linked_list):
            """
            Reverse the inputted linked list

            Args:
               linked_list(obj): Linked List to be reversed
            Returns:
               obj: Reveresed Linked List
            """
            new_list = LinkedList()

            prev_node = None

            """
            A simple idea - Pick a node from the original linked list traversing form the beginn
            prepend it to the new linked list.
            We have to use a loop to iterate over the nodes of original linked list
            """
            # In this "for" loop, the "value" is just a variable whose value will be updated in
            for value in linked_list:
                # create a new node
                new_node = Node(value)

                # Make the new_node.next point to the
                # node created in previous iteration
                new_node.next = prev_node

                # This is the last statement of the loop
                # Mark the current new node as the "prev_node" for next iteration
                prev_node = new_node

            # Update the new_list.head to point to the final node that came out of the loop
            new_list.head = prev_node

            return new_list
```