Daily Coding Problem #127

Problem

This problem was asked by Microsoft.

Let's represent an integer in a linked list format by having each node represent a digit in the number. The nodes make up the number in reversed order.

For example, the following linked list:

```
1 -> 2 -> 3 -> 4 -> 5
```

is the number 54321.

Given two linked lists in this format, return their sum in the same linked list format.

For example, given

```
9 -> 9
```

```
5 -> 2
```

return 124 (99 + 25) as:

```
4 -> 2 -> 1
```

Solution

We can add two numbers using the same process as elementary grade school addition — add the least significant digits with a carry.

We'll start at the head of the two nodes, and compute `sum % 10`. We write this down, move the two nodes up and add a carry if the sum was greater than `10`. A tricky part here is to finding the terminating condition. We can see that this happens when there is no more carry and the two linked lists have reached the end. Since this is tricky to represent in code, we instead extend the nodes one more until they are both `None` and carry is `0` and then simply return `None`.

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

def add(node0, node1, carry=0):
    if not node0 and not node1 and not carry:
        return None

    node0_val = node0.val if node0 else 0
    node1_val = node1.val if node1 else 0
    total = node0_val + node1_val + carry

    node0_next = node0.next if node0 else None
    node1_next = node1.next if node1 else None
    carry_next = 1 if total >= 10 else 0


    return Node(total % 10, add(node0_next, node1_next, carry_next))
```

This will run in O(max(m, n)) time, where m and n are the lengths of the linked lists.