Singly Linked List Implementation

In this lecture we will implement a basic Singly Linked List.

Remember, in a singly linked list, we have an ordered list of items as individual Nodes that have pointers to other Nodes.

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

    def __init__(self,value):

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

Now we can build out Linked List with the collection of nodes:

In [2]:
```
a = Node(1)
b = Node(2)
c = Node(3)
```

In [3]:
```
a.nextnode = b
```

In [4]:
```
b.nextnode = c
```

In a Linked List the first node is called the **head** and the last node is called the **tail**. Let's discuss the pros and cons of Linked Lists:

Pros

- Linked Lists have constant-time insertions and deletions in any position, in comparison, arrays require O(n) time to do the same thing.
- Linked lists can continue to expand without having to specify their size ahead of time (remember our lectures on Array sizing form the Array Sequence section of the course!)

Cons

- To access an element in a linked list, you need to take O(k) time to go from the head of the list to the kth element. In contrast, arrays have constant time operations to access elements in an array.

Good Job!

That's it for the implementation (pretty simple right?). Up next we will learn about Doubly Linked Lists!