Singly Linked List Cycle Check - SOLUTION

Problem

Given a singly linked list, write a function which takes in the first node in a singly linked list and returns a boolean indicating if the linked list contains a "cycle".

A cycle is when a node's next point actually points back to a previous node in the list. This is also sometimes known as a circularly linked list.

You've been given the Linked List Node class code:

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

    def __init__(self,value):

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

Solution

To solve this problem we will have two markers traversing through the list. **marker1** and **marker2**. We will have both makers begin at the first node of the list and traverse through the linked list. However the second marker, marker2, will move two nodes ahead for every one node that marker1 moves.

By this logic we can imagine that the markers are "racing" through the linked list, with marker2 moving faster. If the linked list has a cylce and is circularly connected we will have the analogy of a track, in this case the marker2 will eventually be "lapping" the marker1 and they will equal each other.

If the linked list has no cycle, then marker2 should be able to continue on until the very end, never equaling the first marker.

Let's see this logic coded out:

In [2]:
```
def cycle_check(node):

    # Begin both markers at the first node
    marker1 = node
    marker2 = node

    # Go until end of list
    while marker2 != None and marker2.nextnode != None:

        # Note
        marker1 = marker1.nextnode
        marker2 = marker2.nextnode.nextnode

        # Check if the markers have matched
        if marker2 == marker1:
            return True
```

```
    # Case where marker ahead reaches the end of the list
    return False
```

Test Your Solution

In [3]:
```
"""
RUN THIS CELL TO TEST YOUR SOLUTION
"""
from nose.tools import assert_equal

# CREATE CYCLE LIST
a = Node(1)
b = Node(2)
c = Node(3)

a.nextnode = b
b.nextnode = c
c.nextnode = a # Cycle Here!


# CREATE NON CYCLE LIST
x = Node(1)
y = Node(2)
z = Node(3)

x.nextnode = y
y.nextnode = z


##############
class TestCycleCheck(object):

    def test(self,sol):
        assert_equal(sol(a),True)
        assert_equal(sol(x),False)

        print("ALL TEST CASES PASSED")

# Run Tests

t = TestCycleCheck()
t.test(cycle_check)
```

```
ALL TEST CASES PASSED
```

Good Job!