Singly Linked List

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

    def __init__(self,value):

        self.value = value
        self.next_node = None
```

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

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

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

In [5]:
```
a.value
```

Out[5]:
```
1
```

In [6]:
```
a.next_node
```

Out[6]:
```
<__main__.Node at 0x1cfa1c91948>
```

In [7]:
```
a.next_node.value
```

Out[7]:
```
2
```

Doubly Linked List

In [8]:
```
class DoublyLinkedListNode(object):

    def __init__(self, value):

        self.value = value
        self.next_node = None
        self.prev_node = None
```

In [9]:
```
d = DoublyLinkedListNode(4)
e = DoublyLinkedListNode(5)
f = DoublyLinkedListNode(6)
```

In [10]:
```
d.next_node = e
e.prev_node = d
```

In [11]:
```
e.next_node = f
f.prev_node = e
```

Linked List cycle check: Circular Linked List

In [12]:
```
def cycle_check(node):

    marker1 = node
    marker2 = node

    while marker2.next_node!= None and marker2 != None:

        marker1 = marker1.next_node
        marker2 = marker2.next_node.next_node

        if marker2 == marker1:
            return True

    return False
```

In [13]:
```
class Node(object):

    def __init__(self,value):

        self.value = value
        self.next_node = None
```

In [14]:
```
!pip install nose
```

```
Requirement already satisfied: nose in e:\users\user.desktop-3hhgvth\anaconda3\envs\mytf
env\lib\site-packages (1.3.7)
```

In [15]:
```
'''
Test Cell
'''

from nose.tools import assert_equal

#Circular List
a = Node(1)
b = Node(2)
c = Node(3)

a.next_node = b
b.next_node = c
c.next_node = a

#Non Circular List
x = Node(4)
y = Node(5)
z = Node(6)

x.next_node = y
y.next_node = z
```

In [16]:
```
class TestCycleCheck(object):

    def test_sol(self,sol):

        assert_equal(sol(a),True)
        assert_equal(sol(x),False)

        print('All Test Cases passed')
```

```
#Run Tests

test_cycle_check = TestCycleCheck()
test_cycle_check.test_sol(cycle_check)
```

```
All Test Cases passed
```

Linked List Reversal

In [17]:
```
class Node(object):

    def __init__(self,value):

        self.value =  value
        self.next_node = None
```

In [18]:
```
def reverse(head_node):

    current_node = head_node
    previous_node = None
    next_node = None

    while current_node:

        next_node = current_node.next_node

        current_node.next_node = previous_node

        previous_node = current_node
        current_node = next_node

    return previous_node
```

In [19]:
```
#Create a list of 4 nodes

a = Node(1)
b = Node(2)
c = Node(3)
d = Node(4)

a.next_node = b
b.next_node = c
c.next_node = d
```

In [20]:
```
print (a.next_node.value)
print (b.next_node.value)
print (c.next_node.value)
```

```
2
3
4
```

In [21]:
```
try:
    print (d.next_node.value)
```

```
except:
    print('Value is None')
```

```
Value is None
```

In [22]:
```
#Reverse the list
```

In [23]:
```
reverse(a)
```

Out[23]:
```
<__main__.Node at 0x1cfa1c56948>
```

In [24]:
```
print (d.next_node.value)
print (c.next_node.value)
print (b.next_node.value)
```

```
3
2
1
```

In [ ]:
```

```