

Homework

This homework is meant to guide you through implementing a data structure. For this homework you will be implementing a linked list. Note that this is a very common problem and you will be able to find the solutions readily on the internet. Try not to.

Singly Linked List

A Singly Linked List is a list comprised of many nodes. Each node contains some data, in our case just an integer, and a pointer to the next node in the list. The first node in the list is known as the *head* node. The last node in the graph has a pointer to a *null* next item. Represented graphically, a Singly Linked List looks like this (note that the *head* has value 5 and the *tail* has value 2):



Your assignment is to implement a LinkedList class with the following interface:

- **__init__(self, value):** Takes a number and sets it as the value at the head of the List
- **length(self):** Returns the length of the list
- **addNode(self, new_value):** Takes a number and adds it to the end of the list
- **addNodeAfter(self, new_value, after_node):** Takes a number and adds it after the *after_node*
- **addNodeBefore(self, new_value, before_node):** Takes a value and adds before the *before_node*
- **removeNode(self, node_to_remove):** Removes a node from the list
- **removeNodesByValue(self, value):** Takes a value, removes all nodes with that value
- **reverse(self):** Reverses the order of the linked list
- **__str__(self):** Displays the list in some reasonable way
- **hasCycle(self):** Bonus: Returns true if this linked list has a cycle. This is non-trivial

For each of the above methods, figure out what the computation complexity of your implementation is and state whether or not you think that is the best possible complexity class. Make sure that your implementation is correct and robust to bad inputs.

You are free to define whatever private helper functions/classes/etc. that you need, but make sure that your implementation has the above public facing

interface. You may NOT use any other data structures to implement this. That means no Lists, Arrays, Tuples, etc. You should use the following as the starter definition for a Node class:

```
1 class Node:
2     def __init__(self, _value=None, _next=None):
3         self.value = _value
4         self.next = _next
5     def __str__(self):
6         return str(self.value)
```

Copyright ©2014 Matt Dickenson

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.