CSC148 Intro. to Computer Science
Lecture 5: Linked Lists

Amir H. Chinaei, Summer 2016
Office Hours: R 10-12 BA4222
ahchinaei@cs.toronto.edu
http://www.cs.toronto.edu/~ahchinaei/

Course page:

Review
❖ So far
  ▪ class design and implementation
  ▪ composition and inheritance
  ▪ inheriting, extending, and overriding
  ▪ specific examples:
    • Shape: square, right angled triangle
    • Container: stack, sack, queue, etc.
❖ Today
  ▪ linked lists
  ▪ wrappers and helpers

Regular lists vs. linked lists
❖ Regular Python lists:
  ▪ pro(s): it can efficiently be accessed
  ▪ con(s): they allocate large blocks of contiguous memory, which becomes increasingly difficult as memory is in use.
❖ Linked list nodes reserve just enough memory for the object value they want to refer to, a reference to it, and a reference to the next node in the list
  ▪ pro(s): it can efficiently grow and shrink, as needed
  ▪ con(s): ?

Linked list
❖ For now, we implement a linked list as objects (nodes) with a value and a reference to other similar objects

12 → 99 → 37 →

Helper: Node
❖ Examples:

value | next_

12

value

37

99

99 →

Helper: LinkedListNode class
class LinkedListNode:

Node to be used in linked list data structure

--- Public Attributes ---
:type next_: LinkedListNode
  successor to this LinkedListNode
:type value: object
  data this LinkedListNode represents

def __init__(self, value, next_=None):
  Create LinkedListNode self with data value and successor next_

:param value: data of this linked list node
:type value: object
:param next_: successor to this LinkedListNode
:type next_: LinkedListNode

self.value= value
self.next_= next_
**Helper: LinkedListNode class**

What other methods does class node, i.e. LinkedListNode need?

**Wrapper: LinkedList class**

```python
class LinkedList:
    ""
    Collection of LinkedListNodes
    ""
    type front: LinkedListNode
    first node of this LinkedList
    type back: LinkedListNode
    last node of this LinkedList
    param size: int
    number of nodes in this LinkedList
    ""
    def __init__(self):
        ""
        Create an empty linked list.
        ""
        :rtype: None
        self.front, self.back = None, None
        self.size = 0
```

What other methods does class LinkedList need?