CSI33 Data Structures

Sharon Persinger

Fall 2019

Day 8 September 23

More list methods

_delete(self, position):

private method to delete item at location position from the list

Used in definitions of

delete and pop

Insert

```
def insert(self, i, x):

    '''inserts x at position i in the list
```

Copy

Copy method makes a shallow copy of the LInkedList

Iterator

We can move down a LinkedList easily:

```
my = LList(list(range(10)))
node = my.head
while node is not None:
    print (node.item)
    node = node.link
```

Iterator

- An iterator is an object that know how to step-through the items in a container object.

- Python provides an iterator for its built-in container objects. You use and iterator when you do a loop through the items in a sequence or dictionary.
- Make a list l1.
- Create an iterator with the iter function
  - It = iter(l1)
- The iter object it has a next function. Call it a few times.

Add an iterator to the LList class

- Create a class LListIterator.
  - Define a constructor and a next method.
- Use this class to define the __iter__ method for the Llist class.
- Now we can iterate through the items of a linked list - for each in …..

Which is better – Python array-based list or linked list?

- Memory? Array-based uses less memory, but both are $\Theta(n)$
- Time – depends on the types of operations
  - With a linked list, recopying data is not necessary when inserting and deleting a known locations. Adding a node at the head, deleting the node at the head both take constant time. Adding a node at the end takes constant time when there is a tail variable.
  - Think about what types of insertions will be done to chose which implementation to use.

Linked List assignment

- Assignment 2 on the webpage.