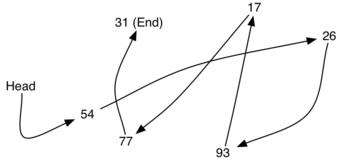
Unordered List

<http://interactivepython.org/courselib/static/pythonds/BasicDS/linkedlists.html>

The Unordered List Abstract Data Type

* List() creates a new list that is empty. It needs no parameters and returns an empty list.
* add(item) adds a new item to the list. It needs the item and returns nothing. Assume the item is not already in the list. Default add item at front.
* remove(item) removes the item from the list. It needs the item and modifies the list. Assume the item is present in the list.
* search(item) searches for the item in the list. It needs the item and returns a boolean value.
* isEmpty() tests to see whether the list is empty. It needs no parameters and returns a boolean value.
* size() returns the number of items in the list. It needs no parameters and returns an integer.
* append(item) adds a new item to the end of the list making it the last item in the collection. It needs the item and returns nothing. Assume the item is not already in the list.
* index(item) returns the position of item in the list. It needs the item and returns the index. Assume the item is in the list.
* insert(pos,item) adds a new item to the list at position pos. It needs the item and returns nothing. Assume the item is not already in the list and there are enough existing items to have position pos.
* pop() removes and returns the last item in the list. It needs nothing and returns an item. Assume the list has at least one item.
* pop(pos) removes and returns the item at position pos. It needs the position and returns the item. Assume the item is in the list.

## Implementing an Unordered List: Linked Lists



### **The Node Class**

**class** **Node**:

**def** \_\_init\_\_(self,initdata):

self.data = initdata

self.next = None

**def** getData(self):

**return** self.data

**def** getNext(self):

**return** self.next

**def** setData(self,newdata):

self.data = newdata

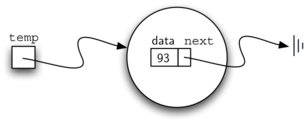
**def** setNext(self,newnext):

self.next = newnext

**>>>** temp = Node(93)

**>>>** temp.getData()

93

 ../_images/node2.png

### **The Unordered List Class**

**class** **UnorderedList**:

**def** \_\_init\_\_(self):

self.head = None

**def** isEmpty(self):

**return** self.head == None

**>>>** mylist = UnorderedList()

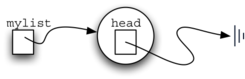


Figure 5: An Empty List



Figure 6: A Linked List of Integers

**def** add(self,item):

temp = Node(item)

temp.setNext(self.head)

self.head = temp

**>>>** mylist.add(31)

**>>>** mylist.add(77)

**>>>** mylist.add(17)

**>>>** mylist.add(93)

**>>>** mylist.add(26)

**>>>** mylist.add(54)

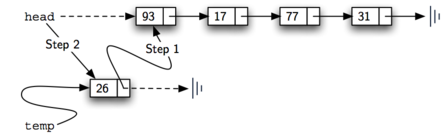


Figure 7: Adding a New Node is a Two-Step Process

**def** size(self):

current = self.head

count = 0

**while** current != None:

count = count + 1

current = current.getNext()

**return** count

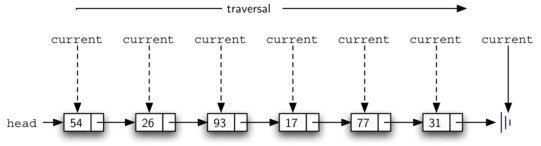


Figure 9: Traversing the Linked List from the Head to the End

**def** search(self,item):

current = self.head

found = False

**while** current != None **and** **not** found:

**if** current.getData() == item:

found = True

**else**:

current = current.getNext()

**return** found

**>>>** mylist.search(17)

True

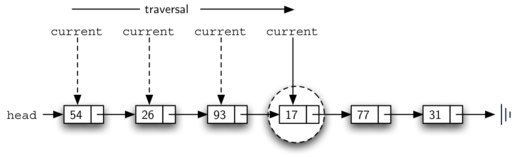


Figure 10: Successful Search for the Value 17

**def** remove(self,item):

current = self.head

previous = None

found = False

**while** **not** found:

**if** current.getData() == item:

found = True

**else**:

previous = current

current = current.getNext()

**if** previous == None:

self.head = current.getNext()

**else**:

previous.setNext(current.getNext())

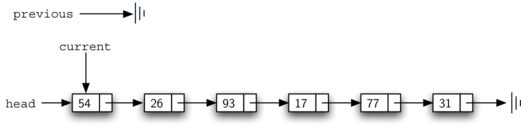


Figure 11: Initial Values for the previous and current References

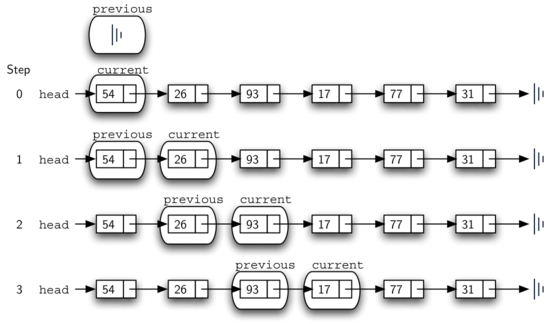


Figure 12: previous and current Move Down the List

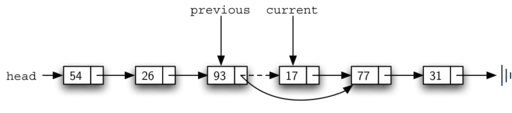


Figure 13: Removing an Item from the Middle of the List

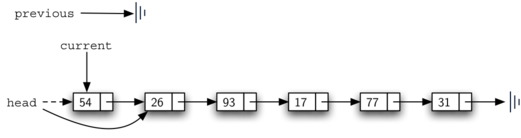


Figure 14: Removing the First Node from the List