EECS268:Lab4

Contents

Due time

This lab is due 1 weeks from the start of your lab.

Overview

I recommend doing this lab in two phases. First, create and test an implementation of a List. Second, make an implementation of the Browser, using your LinkedList.

Create and Test a Linked List

Make sure to...

1. Test each method as you go along
2. Run your code VERY often (if you write even a simple method, run it and test!)
3. I recommend writing a helper class to help speed up tests (e.g. a class that automates populating a LinkedList with value
4. Use the IDLE debugger
   - The debugger can show you exactly what is happening with your list and help track down reasons for errors or exceptions

What not to do...

- Skip testing to try to save time (it doesn't save time in the long run)
- Code an entire class or the entire lab before running for the first time

Here's a listing (pun intended) of the methods your LinkedList will have:

Method	Description
__init__(self)	Initialize list
length(self)	Return length of the list
insert(self, index, entry)	Insert the entry at the index. Valid indices range from 0 to length inclusively. Inserting at index=0 inserts at the front. Inserting at index=length adds to the back. Each insert increases the length by 1.
remove(self, index)	Removes the entry at the index. Valid indices range from 0 to length-1 inclusively. Each remove decreases the length by 1.
get_entry(self, index)	Return the entry at index, raises a RuntimeError otherwise.
set_entry(self, index, entry)	Sets the entry at index, raises a RuntimeError otherwise. Even if successful, the length remains the same.
clear(self)	Empties the list

Web Browser History

You will create a class that mimics the behavior of your web browser's back button, forward button, and address bar.

Here is a listing of methods for the Browser class.

This class can then be used by an Executive class.

Method	Description
__init__(self)	Initialize Browser
navigate_to(self, url)	The browser navigate to the given url
forward(self)	If possible, the browser navigates forward in the history otherwise it keeps focus
back(self)	If possible, the browser navigates backwards in the history otherwise it keeps focus
history(self)	Returns a well formatted string (see below) with the current history.

Reading the history from file

To build up your browser history, you will read in from file. The file name will come in on the command line.

Any given line of the file will contain one of the following entries:

File Entry	Description
NAVIGATE <URL>	Navigates the browser to the given URL. NOTE navigating to a URL retains all URLs accessible from going BACK, but any URLs that would have accessible from going FORWARD are now lost
BACK	A command indicating the web browser is redirected to the previous URL in the history. If there is no URL further back, then the browser stays on the current URL.
FORWARD	A command indicating the web browser is redirected to the next URL in the history. If there is no URL that is next, then the browser stays on the current URL.

HISTORY	Prints the current URL history to the screen using the following format:
	`Oldest`
	`===========`
	`<URL>`
	`<URL>  <==current (assuming this is the current URL)`
	`<URL>`
	`===========`
	`Newest`

Sample input file

```
NAVIGATE http://google.com
NAVIGATE http://reddit.com
NAVIGATE http://facebook.com
NAVIGATE http://myspace.com
HISTORY
BACK
BACK
HISTORY
FORWARD
FORWARD
FORWARD
FORWARD
HISTORY
BACK
BACK
BACK
NAVIGATE http://ku.edu
FORWARD
HISTORY
BACK
HISTORY
```

Output to screen:

```
Oldest
===========
http://google.com
http://reddit.com
http://facebook.com
http://myspace.com  <==current
===========
Newest

Oldest
===========
http://google.com
http://reddit.com   <==current
http://facebook.com
http://myspace.com
===========
Newest

Oldest
===========
http://google.com
http://reddit.com
http://facebook.com
http://myspace.com  <==current
===========
Newest

Oldest
===========
http://google.com
http://ku.edu       <==current
===========
Newest

Oldest
===========
http://google.com   <==current
http://ku.edu
===========
Newest
```

Linked List requirements

Your LinkedList must be made out of your Node class. DO NOT just implement your LinkedList using the existing python lists.

If this class was designed to teach you how to make a pizza, the first step isn't order Dominoes and put your name on the box. We're making it from scratch.

Rubric

- 20pts List Implementation
- 40pts Web Browser Implementation
- 20pts Modularity
  - Sensible class design
  - Completely object oriented (e.g. your main should invoke some kind of executive class)
- 15pts Stability
  - There should be zero unhanded exceptions when provided with a properly formatted file
- 5pts Comments and documentation:
  - docstrings
  - Author, date, last modified comments at the top of each .py file

Submission instructions

All .py and input files should be packaged into a zip or tar.gz file and submitted.

Consult your TA for additional submission instructions.