CS 1110, LAB 6: LISTS AND FOR-LOOPS

http://www.cs.cornell.edu/courses/cs11102013fa/labs/lab06.pdf

First Name: _____ **Last Name**: _____ **NetID**: _____

Just when you had become an expert at string slicing, you discovered another sliceable data type: lists. However, lists are different from strings in that they are *mutable*. Not only can we slice a list, but we can also change its contents. The purpose of the lab is to introduce you to these new features, and demonstrate just how powerful the list type can be.

0.1. **Getting Credit for the Lab.** There are no files to download for this lab. For the the first part of the lab you will be playing with the Python interactive prompt (again). We do ask you to implement several functions below. However, we just want you to write the implementation on a piece of paper (which you will show to the instructor). You do not need to submit any modules, and you do not need to write any unit tests.

When you are done, show all of this handout to your instructor. You instructor will then swipe your ID card to record your success. You do not need to submit the paper with your answers, and you do not need to submit the module.

As with previous labs, you do not need to finish during your section. However, next week is Fall Break and **there are no labs next week**. Therefore, you have until your lab the following week (October 21) to finish. However, we strongly recommend that you finish this lab before then, as all of this material is on the exam.

1. List Expressions and Commands

This part of the lab will take place in the Python interactive prompt, much like the first two labs. You do not need to create a module. First, execute the following assignment statement:

```
lablist = ['H','e','l','l','o',' ','W','o','r','l','d','!']
```

Like a string, this is a list of individual characters. Unlike a string, however, the contents of this list can be changed.

Enter the following statements **in the order they are presented**. Many of the commands below are always type in expressions, Python will immediately display the value; the commands below are all followed by a print statement showing the new contents of the list. Each case, describe what you see and *explain the result.*

Course authors: D. Gries, L. Lee, S. Marschner, W. White

Commands	Result/Explanation
`lablist.remove('o')` `print lablist`	
`lablist.remove('x')`	
`pos = lablist.index('o')` `print pos`	
`pos = lablist.index('B')`	
`lablist[0] = 'J'` `print lablist`	
`lablist.insert(5,'o')` `print lablist`	
`s = lablist[:]` `print s`	

`s[0] = 'C'`
`print s`
`print lablist`	

2. List Functions

On the next page are several function specifications; implement them. You will probably want to test them out on the computer. However, to turn them in you just need to write the final version on a piece of paper and show it. You may find the following list methods useful.

Method	Result When Called
l.index(c)	**Returns**: the first position of c in list l; error if not there
l.count(c)	**Returns**: the number of times that c appears in the list l.
l.append(c)	Add the value c to the end of the list. This method alters the list; it does not make a new list.

Lists do *not* have a `find()` method like strings do. They only have `index()`. To check if an element is in a list, use the `in` operator (e.g. `x in thelist`).

Function `lesser_than(thelist,value)`.

The function below **should not alter** `thelist`. If you need to call a method that might alter the contents of thelist, you should make a copy of it first.

```
def lesser_than(thelist,value)
    """Returns:  number of elements in thelist strictly less than value,
    without altering thelist.

    Example:  lesser_than([5, 9, 1, 7], 6) evaluates to 2

    Precondition:  thelist is a list of ints; value is an int"""
```

Function `uniques(thelist)`.

```
def unique(thelist)
    """Returns:  The number of unique elements in the list.

    Example:  is_uniform([5, 9, 5, 7]) evaluates to 3
    Example:  is_uniform([5, 5, 1, 'a', 5, 'a']) evaluates to 3

    Precondition:  thelist is a list."""
```

Function `isuniform(thelist)`.

```
def isuniform(thelist)
    """Returns:  True if the elements of thelist are all the same type.

    Example:  isuniform([5, 9, 1, 7]) evaluates to True
    Example:  isuniform([5, 9.5, 'a']) evaluates to False

    Example:  isuniform([]) evaluates to True
    Precondition:  thelist is a list."""
```