CS 1110, LAB 7: LISTS AND FOR-LOOPS

http://www.cs.cornell.edu/courses/cs1110/2016fa/labs/lab07/

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.

This lab will also give you some experience writing functions with for-loops. While lists will be on the first exam, for-loops are on the second exam, after you have had more practice with them.

Lab Materials. We have created several Python files for this lab. You can download all of the from the Labs section of the course web page.

http://www.cs.cornell.edu/courses/cs1110/2016fa/labs

For today's lab you will notice two files.

- `lab07.py` (a module functions for you to implement)
- `test07.py` (a unit test for your module)

Once again you should create a *new* directory on your hard drive and download all of the files into that directory. Alternatively, you can get all of the files bundled in a single ZIP file called `lab07.zip` from the Labs section of the course web page.

Getting Credit for the Lab. Once again, you have a choice between getting credit through the online system or your instructor. The online lab is available at the web page

http://www.cs.cornell.edu/courses/cs1110/2016fa/labs/lab07/

By now you should have a good idea of which version you are more comfortable with. This lab involves quite a bit of coding, so you may have trouble finishing it during lab time. Fortunately, **you have two weeks this time to finish the lab**.

If you are using the worksheet, show both this handout and the file `lab07.py` to your lab instructor. Your instructor will record that you did it. Remember, despite the demands of the online system, labs are graded on effort, not correctness.

1. List Expressions and Commands

The first 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. This makes lists a very important data type.

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

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, write down what you see in the second column. Unlike previous labs, we are not asking you to guess the output beforehand.

Commands	Output
`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(4,'o')` `print lablist`	
`s = lablist[:]` `print s`	

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

`a = '-'.join(s)` `print a`	
`a = ''.join(s)` `print a`	
`t = list(a)` `print t`	

This next table is similar to those from previous labs. The commands and expressions in the first column are missing something, represented by a box. That something may be a literal, a variable, or a method name. The second column displays the output. You need to fill in the box to give the desired output. If you are confused, go back and look at your answers on the previous page.

Commands	Output	Contents in the Box

`lablist = list('CS1110')`
`lablist.remove( )`
`print lablist`
	`['C','S','1','1','0']`	

`pos = lablist.index( )` `print pos`	`2`	
`pos = lablist.index('1', )` `print pos`	`3`	
`lablist.insert( ,'I')` `print lablist`	`['C','I','S','1','1','0']`	
`a = .join(lablist)` `print a`	`'C.I.S.1.1.0'`	

2. List Functions

On the next two pages are several function specifications; implement them. The stubs for these functions are in the file `lab07.py`. You will need to use for-loops to implement them. In addition, we have already provided you with test cases in `test_lab07.py`. So all you need to do is implement the functions.

In addition to using a for-loop, 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)	Adds 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

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

    Parameter thelist:  the list to check (WHICH SHOULD NOT BE MODIFIED)
    Precondition: thelist is a list of ints

    Parameter value: the value to compare to the list
    Precondition: value is an int"""
```

Function `uniques(thelist).`

Once again, 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 uniques(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

    Parameter thelist: the list to check (WHICH SHOULD NOT BE MODIFIED)
    Precondition: thelist is a list."""
```

Function `clamp(thelist,min,max).`

Unlike the previous two functions, this function *does* alter `thelist`. This function is a procedure with no return value. You might want to look at `test_lab07.py` to see how we would test a procedure like this.

```
def clamp(thelist,min,max):
    """Modifies the list so that every element is between min and max.

    Values in the list less than min are replaced with min.  Values greater than
    max are replaced with max.  Values between min and max are left unchanged.

    This is a PROCEDURE. It modifies thelist, but does not return a new list.

    Example: if thelist is [-1, 1, 3, 5], then clamp(thelist,0,4) changes thelist
    to have [0,1,3,4] as its contents.

    Parameter thelist: the list to modify
    Precondition: thelist is a list of numbers (float or int)

    Parameter min: the minimum value for the list
    Precondition: min <= max is a number

    Parameter max: the maximum value for the list
    Precondition: max >= min is a number"""
```