

CS 1110, LAB 7: LISTS AND FOR-LOOPS

<http://www.cs.cornell.edu/courses/cs1110/2015fa/labs/lab07.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.

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/cs1110/2015fa/labs>

For today's lab you will notice two files.

- `lab07.py` (a module functions for you to implement)
- `test_lab07.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 `lab04.zip` from the Labs section of the course web page.

Getting Credit for the Lab. 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**. However, you must be finished **before** the start of your next lab. Too many students are spending most of the lab time working on the lab from the previous week and we can no longer allow this.

When you are done, show all of this handout to your instructor. Your 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.

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.

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*.

Commands	Result/Explanation
<pre>lablist.remove('o') print lablist</pre>	
<pre>lablist.remove('x')</pre>	
<pre>pos = lablist.index('o') print pos</pre>	
<pre>pos = lablist.index('B')</pre>	
<pre>lablist[0] = 'J' print lablist</pre>	
<pre>lablist.insert(5,'o') print lablist</pre>	
<pre>s = lablist[:] print s</pre>	
<pre>s[0] = 'C' print s print lablist</pre>	
<pre>a = '-'.join(s) print a</pre>	
<pre>a = ''.join(s) print a</pre>	
<pre>t = list(a) print t</pre>	

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
<code>l.index(c)</code>	Returns: the first position of <code>c</code> in list <code>l</code> ; error if not there
<code>l.count(c)</code>	Returns: the number of times that <code>c</code> appears in the list <code>l</code> .
<code>l.append(c)</code>	Add the value <code>c</code> 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.  
    Any number in the list less than min is replaced with min. Any number  
    in the list greater than max is replaced with max. Any number between  
    min and max is left unchanged.  
    This is a PROCEDURE. It modified 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"""
```