CS 1110, LAB 6: LISTS AND ASSERTS

http://www.cs.cornell.edu/courses/cs1110/2014fa/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.

The other major topic this week was asserts and the try-except statement. We give you some practice with this at the end of the lab. You will get a lot of experience with asserts on Assignment 3, due next week.

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. If you do not finish during section, you have **until the beginning of lab next week to finish it**. You should always do your best to finish during lab hours; remember that labs are graded on effort, not correctness.

1. Asserts and Error Handling

Strings have the following useful methods:

Method	Description
s.isalpha()	**Returns**: True if `s` is nonempty and has only letters; False otherwise.
s.isdigit()	**Returns**: True if `s` is nonempty and has only numbers; False otherwise.
s.islower()	**Returns**: True if all letters in `s` are lower case; False otherwise.
s.isupper()	**Returns**: True if all letters in `s` are upper case; False otherwise.

Recall the function `pigify` from the previous lab. It had the following specification:

```
def pigify(w):
    """Returns: copy of w converted to Pig Latin.
    Precondition: w is a nonempty string with only lowercase letters"""
```

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

You do not need your implementation (in `lab05.py` for this lab. However, you should reread the precondition. In the box below, write assert statement(s) that enforce this precondition. Remember to check the type as well!

Now suppose you had the following function to be used together with `pigify` (you can add this to `lab05.py` if you wish).

```
def try_pigify(w)
    """Returns: copy of w converted to Pig Latin.
    Precondition: there is no precondition"""
    result = ''
    try:
        result = w.lower()
        return pigify(result)

    except:
        return result
```

Suppose you executed the assignment statement `s = try_pigify('Piggy')`. Once the call is completed, what is in the variable `s`? **Explain your answer**.

On the other hand, suppose you executed the statement `s = try_pigify('Piggy1')`. Now what is in the variable `s` and why?

2. 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.*

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`	

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

3. Implementing a Function on Lists

Listed below is a single function specification; implement this function. You will probably want to test it out on the computer. However, to turn it in, you just need to write the final version on this piece of paper and show it. You might 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 c to the end of the list. This method alters the list; it does not make a copy.
l.sort()	Rearrange the elements of the list l in ascending order. This method alters the list; it does not make a new list.

Remember, 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 lesser than value,
    without altering thelist.

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

    Precondition:  thelist is a list of ints; value is an int, and value is
    contained in thelist"""
```