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.

In addition to working with lists, this lab will also require you to use the new conditionals that we have seen in class: conditionals and for loops.

0.1. **Getting Credit for the Lab.** There are no files to download for this lab. For 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, who will record that you did it. As with previous labs, if you do not finish during your 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. **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
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.

<table>
<thead>
<tr>
<th>Method</th>
<th>Result When Called</th>
</tr>
</thead>
<tbody>
<tr>
<td>l.index(c)</td>
<td><strong>Returns</strong>: the first position of c in list l; error if not there</td>
</tr>
<tr>
<td>l.count(c)</td>
<td><strong>Returns</strong>: the number of times that c appears in the list l.</td>
</tr>
<tr>
<td>l.append(c)</td>
<td>Add the value c to the end of the list. This method alters the list; it does not make a new list.</td>
</tr>
</tbody>
</table>

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 swap(thelist,a,b).

```python
def swap(thelist,a,b):
    """Swap the elements at positions a and b in thelist.

    Precondition: thelist is a list; a, b are valid positions 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.

```python
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 `is_uniform(thelist)`.

```python
def is_uniform(thelist):
    """Returns: True if the elements of thelist are all the same type.
    Example: is_uniform([5, 9, 1, 7]) evaluates to True
    Example: is_uniform([5, 9.5, 'a']) evaluates to False
    Example: is_uniform([]) evaluates to True
    Precondition: thelist is a list.""
```

Function `uniques(thelist)`.

```python
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.""
```