Assignment 3: Lists

Lists

Lists are exactly as they sound, they are just lists of data. The data that is contained within a list can be of any type. You can have a string, float, integer, and Boolean all in the same list.

In Python, lists are exactly as they sound, an ordered sequence of elements that can be changed or modified. Each element or value contained within a list is referred to as an item.

The data “items” contained within a list can be of any type. You can have a string, float, integer, and Boolean all in the same list.

Lists are defined by having values inside square brackets [], just as strings are defined by characters inside quotations.

Lists are useful when you need to work with a large number of related values. They allow you to keep data that belongs together, simplify your code, and apply the same procedures and operations on several items at the same time.

How To Create A List

```
empty = []
presidents = ["Washington", "Adams", "Jefferson"]
stuff = ["Purdue", 3, 4.2, True]
```

Lists are created like any other variable, with the name of the list being followed by an equal sign. Lists are denoted by a pair of square brackets ([]) at the beginning and end. Each of the items in the list is separated by commas. The presidents list only contains strings, however you can mix variables in lists, as is seen with the stuff list. The empty variable is still a list, however, it is simply empty. Since there are no values inside the square brackets.

Accessing Lists

```
presidents = ["Washington", "Adams", "Jefferson"]
first = presidents[0] # Washington
```

In order to access one of the items in a list, you type the name of the list, followed by the respective index number surrounded by square brackets.

Python uses *zero-based numbering*. This means that the first element in a list is accessed by using the index 0. I know this is weird, but you'll get the hang of it

With the presidents list example above, in order to access each of the strings, you would use the following indices.

```
presidents[0] # Washington
presidents[1] # Adams
presidents[2] # Jefferson
```

Accessing Range of Items:

```
presidents = ["Washington", "Adams", "Jefferson"]
twoThree = presidents[1:3] # Adams and Jefferson
```

The process for accessing multiple items in a list is similar to accessing a single item. Instead of only specifying a single index, you provide a range, divided by a colon ([start:end]). Keep in mind that the range starts with the first index and ends with the second index (non-inclusive). In the example above, we have to do a range 1:3, because we want to start with "Adams" (index 1) and end with "Jefferson" (index 2). Remember, we start counting with 0 in Python. We put 3 since we want to stop right before index 3 (which doesn't exist in this case, since the list ends).

Add Items to a List:

The list.append(item) function will add the specified value to the end of the list. You type the name of your respective list (like presidents), then .append(), with the item you would like to add to the list in the parentheses.

```
presidents = ["Washington", "Adams", "Jefferson"]
presidents.append("Monroe")
 # ["Washington", "Adams", "Jefferson", "Monroe "]
```

The list.insert(index, item) will add the passed-in item to the specified index. The preexisting items in the list following the insertion location will be shifted over by a single index (i.e. "Monroe" goes from index 3 to index 4).

```
presidents.insert(3, "Madison")
# ["Washington", "Adams", "Jefferson", "Madison", "Monroe"]
```

Keep in mind that these methods can only add one item at a time. There are ways to add several at the same time, but those are not necessary for you to learn.

Remove Items from a List:

Removing an item from a list is as simple as using the list.remove(item) notation, with the value of the item between the parentheses.

```
states = ["Alaska", "California", "Indiana", "Washington, D.C.",
"Wyoming"]
states.remove("Washington, D.C.")
```

len() Function:

A very useful function for lists is the len() function. It returns the length of various data structures, like strings, lists, and dictionaries. For instance, it returns the number of characters in a string. With lists, it returns the number of items in a list.

This function is extremely useful and will come in handy when we learn about loops.

```
word = "amazing!"

print(len(word)) # prints 8

court = ["Thomas", "Ginsburg", "Breyer", "Roberts", "Alito",
"Sotomayor", "Kagen", "Gorsuch", "Kavanaugh"]

print(len(court)) # prints 9
```

Okay, that was a lot about lists. Let's get some practice in.

List_Exercises:

You have 3 exercises covering lists. Most of the code has been given to you. Your goal is to fill in the gaps.

The starter code has been given to you in the lists.ipynb file in Brightspace.

Submit the lists.py file to Gradescope when done.