CSE 110

Learning Activity (2 of 2): List Indexes

Prepare

Overview

This activity dives deeper into working with lists.

In the previous activity, you learned how to append items to a list and how to iterate through the items one by one. In this activity, you'll learn how to access items in a list at any location, and also how to remove items.

Accessing items in a list via their index

You can access an item in a list at a given index via the square bracket `[]` notation:

```
first = the_list[0] # gets the first item
second = the_list[1] # gets the second item

index = int(input("Which index would you like to get? "))
user_choice = the_list[index] # gets the item at the index the user typ
```

Don't forget that Python, like most programming languages, starts counting at 0 for its indexes. So the first item is at index 0, and the last item in a list of 10 elements would be at index 9.

Finding the size of the list

You can find out how many elements are in a list, by using the **`len`** function (which is short for *length*) as follows:

```
number_of_books = len(books)
```

Iterating through a list using an index

In previous lessons, you learned how to iterate through each item in a list using a standard for loop. Another way to do this is to have the loop iterate through the indexes 0 through the size of the list, and then access each item using the `[]` notation:

```
for i in range(len(books)):
    book = books[i]
    print(book) # print each book to the screen.
```

Looking a little closer at that for loop you can see that **`len(books)`** gets the number of items in the list, and then the **`for i in range(...):`** part iterates through the numbers 0, 1, 2, 3, ..., up until (but not including) the size of the list. So if the list has 10 elements, this will loop through from 0–9, which are the exact indexes you want.

Hint from Instructor:

A note about variable names:

Typically, we avoid single letter variable names like **`i`** or **`j`**, but when it comes to using loops to iterate through numbers like this, it is very common to use "i" as the variable name. Then if you happen to have a second loop inside of it, it is common to use "j" for that variable. A third loop uses **`k`**.

If you have more than 3 loops inside one another, typically you need to examine your code a little closer to see if there is a way to simplify your problem.

Printing indexes

If you want to print the index numbers next to the elements of the list as you iterate through, you can print the value of the **`i`** variable:

```
for i in range(len(books)):
    book = books[i]
    print(f"{i}. {book}")
```

But be careful. Don't forget that the indexes will start at 0. If you want to display these numbers in a more user-friendly manner, you may need to add 1 to the variable **`i`** when you display it.

Working with parallel lists

Sometimes, you have a situation where you have two lists that you want to work with in parallel. For example, the first list may have the names of your friends, and the second may have their phone numbers. If you want to iterate through and display the name and the phone number, you would not be able to use a standard loop such as **`for name in names:`** because you wouldn't be able to get the corresponding number.

The most common solution to this problem is to loop through the indexes that correspond to one of the lists and then you can access the item from each list at that index:

```
names = []
numbers = []

# ...
# Code here that populates the two lists
# ...

for i in range(len(names)):
    name = names[i]
    number = numbers[i]

    print(f"{name} - {number}")
```

Hint from Instructor:

Keep in mind that you have to be very careful in cases like this that the two lists do not get out of sync. In future courses, you'll learn other techniques to combine two elements into a single type of data, so that you can always keep them together.

Removing items from a list

There are a few different ways to remove items from a list in Python, but the easiest is to use the "pop" function to *pop* the item out of the list. You tell the pop function the

index of the item you want to remove. If you don't give it an index, it will remove the last one:

```
the_list.pop(3) # Removes the item at index 3

the_list.pop() # Removes the last item
```

When you remove an item from a certain index, all of the elements in the list that follow it will move up. In other words, if you remove the item at index 3, the item that was at index 4 will move to 3, the one at 5 will move to 4, and so on.

Activity Instructions

Overview

Practice using lists and list indexes.

Instructions

Ask the user for a list of list of items in a shopping list, stop asking for items when the user types "quit."

Then complete the following:

1. Loop through the items in the regular way (for example, **`for thing in the_list`**) and display each one to make sure you have the initial list built correctly.
2. Add another loop to go through and print the items in the list, but this time, instead of using the **`for ... in`** syntax, use an index (for example, **`for ... in range`**) and then access the item using the index and the square brackets. Print the index in front of the items like so: **`0. Milk`**
3. Ask the user for an index and a new shopping list item. Replace the item at that index with the new item. Then, display the whole list again.

The following examples show the expected output:

```
Please enter the items of the shopping list (type: quit to finish):
Item: Milk
Item: Bread
Item: Candy
Item: Carrots
Item: quit
```

```
The shopping list is:
Milk
Bread
Candy
Carrots

The shopping list with indexes is:
0. Milk
1. Bread
2. Candy
3. Carrots

Which item would you like to change? 2
What is the new item? Apples

The shopping list with indexes is:
0. Milk
1. Bread
2. Apples
3. Carrots
```

Sample Solution

When your program is finished, please view a sample solution of this program to compare your approach to that one.

You should work to complete this checkpoint program first, without looking at the sample solution. However, if you have worked on it for at least an hour and are still having problems, you may feel free to use the sample solution to help you finish your program.

- Sample Solution

Testing Procedure

Verify that your program works correctly by following each step in this testing procedure:

1. Verify that you can add all the items to the list and display them. (Verify similar to the check point from the previous lesson.)
2. Iterate through the list using an index. Verify that your program displays the index before the item and that the index starts at 0 for the first item.
3. Verify that you can ask the user to type an index and a new item.

4. Set the value at that index to be the new item the user typed. Verify that this does not cause errors.
5. Print the items out again, and verify that your new item is in the list at the correct spot.

Submission

You have now completed all of the learning activities for the week!

Make sure to:

- Return to Canvas and submit the associated quiz.

Up Next

- Check Your Understanding

Other Links:

- Return to: Week Overview | Course Home