BYU IDAHO CSE 110 | Programming Building Blocks

# 10 Prepare: Preparation Material

## Overview

This lesson dives deeper into working with lists.

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

## Preparation Material

### I. 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 typed
```

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.

### II. 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)
```

### III. 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.

### IV. 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.

### V. 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.

## VI. 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.