C3

Introducing Lists

CHAPTER ASSIGNMENT

Name _____ Class _____ No. _____ Date _____

Try these short programs to get some firsthand experience with Python's lists. You might want to create a new folder for each chapter's exercises to keep them organized. The following exercises are a bit more complex than those in Chapter 2, but they give you an opportunity to use lists in all of the ways described.

Questions from *Python Crash Course*, 2nd edition, Eric Matthes (No Starch Press), for classroom use.

Try It Yourself

3-1 **Names:** Store the names of a few of your friends in a list called `names`. Print each person's name by accessing each element in the list, one at a time.

3-2 **Greetings:** Start with the list you used in Exercise 3-1, but instead of just printing each person's name, print a message to them. The text of each message should be the same, but each message should be personalized with the person's name.

3-3 **Your Own List:** Think of your favorite mode of transportation, such as a motorcycle or a car, and make a list that stores several examples. Use your list to print a series of statements about these items, such as “I would like to own a Honda motorcycle.”

3-4 **Guest List:** If you could invite anyone, living or deceased, to dinner, who would you invite? Make a list that includes at least three people you’d like to invite to dinner. Then use your list to print a message to each person, inviting them to dinner.

3-5 **Changing Guest List:** You just heard that one of your guests can't make the dinner, so you need to send out a new set of invitations. You'll have to think of someone else to invite.

• Start with your program from Exercise 3-4. Add a `print()` call at the end of your program stating the name of the guest who can't make it.

• Modify your list, replacing the name of the guest who can't make it with the name of the new person you are inviting.

• Print a second set of invitation messages, one for each person who is still in your list.

3-6 **More Guests:** You just found a bigger dinner table, so now more space is available. Think of three more guests to invite to dinner.

• Start with your program from Exercise 3-4 or Exercise 3-5. Add a `print()` call to the end of your program informing people that you found a bigger dinner table.

• Use `insert()` to add one new guest to the beginning of your list.

• Use `insert()` to add one new guest to the middle of your list.

• Use `append()` to add one new guest to the end of your list.

• Print a new set of invitation messages, one for each person in your list.

3-7 **Shrinking Guest List:** You just found out that your new dinner table won't arrive in time for the dinner, and you have space for only two guests.

• Start with your program from Exercise 3-6. Add a new line that prints a message saying that you can invite only two people for dinner.

• Use `pop()` to remove guests from your list one at a time until only two names remain in your list. Each time you pop a name from your list, print a message to that person letting them know you're sorry you can't invite them to dinner.

• Print a message to each of the two people still on your list, letting them know they're still invited.

• Use `del` to remove the last two names from your list, so you have an empty list. Print your list to make sure you actually have an empty list at the end of your program.

3-8 **Seeing the World:** Think of at least five places in the world you'd like to visit.

- Store the locations in a list. Make sure the list is not in alphabetical order.
- Print your list in its original order. Don't worry about printing the list neatly, just print it as a raw Python list.
- Use `sorted()` to print your list in alphabetical order without modifying the actual list.
- Show that your list is still in its original order by printing it.
- Use `sorted()` to print your list in reverse alphabetical order without changing the order of the original list.
- Show that your list is still in its original order by printing it again.
- Use `reverse()` to change the order of your list. Print the list to show that its order has changed.
- Use `reverse()` to change the order of your list again. Print the list to show it's back to its original order.
- Use `sort()` to change your list so it's stored in alphabetical order. Print the list to show that its order has been changed.
- Use `sort()` to change your list so it's stored in reverse alphabetical order. Print the list to show that its order has changed.

3-9 **Dinner Guests:** Working with one of the programs from Exercises 3-4 through 3-7 (page 42), use `len()` to print a message indicating the number of people you are inviting to dinner.

3-10 **Every Function:** Think of something you could store in a list. For example, you could make a list of mountains, rivers, countries, cities, languages, or anything else you'd like. Write a program that creates a list containing these items and then uses each function introduced in this chapter at least once.

3-11 **Intentional Error:** If you haven't received an index error in one of your programs yet, try to make one happen. Change an index in one of your programs to produce an index error. Make sure you correct the error before closing the program.