Session 6: Programming Exercises

1. **Total Sales**

   Design a program that asks the user to enter a store's sales for each day of the week. The amounts should be stored in a list. Use a loop to calculate the total sales for the week and display the result.

2. **Lottery Number Generator**

   Design a program that generates a seven-digit lottery number. The program should generate seven random numbers, each in the range of 0 through 9, and assign each number to a list element. (Random numbers were discussed in **Chapter 5**.) Then write another loop that displays the contents of the list.

3. **Rainfall Statistics**

   Design a program that lets the user enter the total rainfall for each of 12 months into a list. The program should calculate and display the total rainfall for the year, the average monthly rainfall, the months with the highest and lowest amounts.

4. **Number Analysis Program**

   Design a program that asks the user to enter a series of 20 numbers. The program should store the numbers in a list then display the following data:

   - The lowest number in the list
   - The highest number in the list
   - The total of the numbers in the list
   - The average of the numbers in the list

5. **Charge Account Validation**

   If you have downloaded the source code from the Computer Science Portal you will find a file named charge_accounts.txt in the ***Chapter 07*** folder. This file has a list of a company's valid charge account numbers. Each account number is a seven-digit number, such as 5658845.

   Write a program that reads the contents of the file into a list. The program should then ask the user to enter a charge account number. The program should determine whether the number is valid by searching for it in the list. If the number is in the list, the program should display a message indicating the number is valid. If the number is not in the list, the program should display a message indicating the number is invalid.

(You can access the Computer Science Portal at **www.pearsonhighered.com/gaddis**.)

6. Larger Than *n*

In a program, write a function that accepts two arguments: a list, and a number *n*. Assume that the list contains numbers. The function should display all of the numbers in the list that are greater than the number *n*.

7. Driver's License Exam

The local driver's license office has asked you to create an application that grades the written portion of the driver's license exam. The exam has 20 multiple-choice questions. Here are the correct answers:

a. A
b. C
c. A
d. A
e. D
f. B
g. C
h. A
i. C
j. B
k. A
l. D
m. C
n. A
o. D
p. C
q. B
r. B
s. D
t. A

Your program should store these correct answers in a list. The program should read the student's answers for each of the 20 questions from a text file and store the answers in another list. (Create your own text file to test the application.) After the student's answers have been read from the file, the program should display a message indicating whether the student passed or failed the exam. (A student must correctly answer 15 of the 20 questions to pass the exam.) It should then display the total number of correctly answered questions, the total number of incorrectly answered questions, and a list showing the question numbers of the incorrectly answered questions.

8. **Name Search**

   If you have downloaded the source code you will find the following files in the ***Chapter 07*** folder:

   - GirlNames.txt This file contains a list of the 200 most popular names given to girls born in the United States from the year 2000 through 2009.
   - BoyNames.txt This file contains a list of the 200 most popular names given to boys born in the United States from the year 2000 through 2009.

   - Write a program that reads the contents of the two files into two separate lists. The user should be able to enter a boy's name, a girl's name, or both, and the application will display messages indicating whether the names were among the most popular.

   - (You can access the Computer Science Portal at **www.pearsonhighered.com/gaddis**.)

9. **Population Data**

   If you have downloaded the source code you will find a file named USPopulation.txt in the ***Chapter 07*** folder. The file contains the midyear population of the United States, in thousands, during the years 1950 through 1990. The first line in the file contains the population for 1950, the second line contains the population for 1951, and so forth.

   Write a program that reads the file's contents into a list. The program should display the following data:

   - The average annual change in population during the time period
   - The year with the greatest increase in population during the time period
   - The year with the smallest increase in population during the time period

10. **World Series Champions**

   If you have downloaded the source code you will find a file named WorldSeriesWinners.txt in the ***Chapter 07*** folder. This file contains a chronological list of the World Series winning teams from 1903 through 2009. (The first line in the file is the name of the team that won in 1903, and the last line is the name of the team that won in 2009. Note the World Series was not played in 1904 or 1994.)

   Write a program that lets the user enter the name of a team, then displays the number of times that team has won the World Series in the time period from 1903 through 2009.

Read the contents of the WorldSeriesWinners.txt file into a list. When the user enters the name of a team, the program should step through the list, counting the number of times the selected team appears.

11. Lo Shu Magic Square

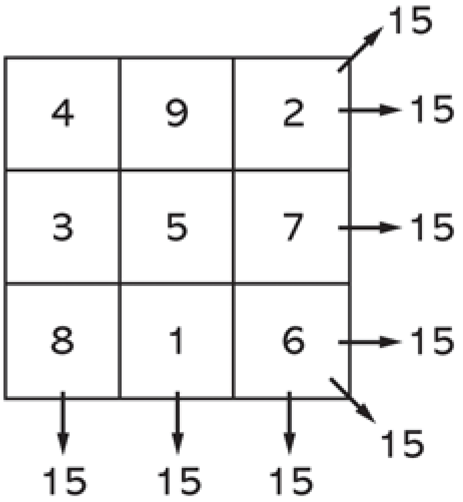
The Lo Shu Magic Square is a grid with 3 rows and 3 columns, shown in **Figure 7-18**. The Lo Shu Magic Square has the following properties:

Figure 7-18: The Lo Shu Magic Square

4	9	2
3	5	7
8	1	6

- The grid contains the numbers 1 through 9 exactly.
- The sum of each row, each column, and each diagonal all add up to the same number. This is shown in **Figure 7-19**.

Figure 7-19: The sum of the rows, columns, and diagonals



In a program you can simulate a magic square using a two-dimensional list. Write a function that accepts a two-dimensional list as an argument and determines whether the list is a Lo Shu Magic Square. Test the function in a program.

12. Prime Number Generation

A positive integer greater than 1 is said to be *prime* if it has no divisors other than 1 and itself. A positive integer greater than 1 is *composite* if it is not prime. Write a program that asks the user to enter an integer greater than 1, then displays all of the prime numbers that are less than or equal to the number entered. The program should work as follows:

- Once the user has entered a number, the program should populate a list with all of the integers from 2 up through the value entered.
- The program should then use a loop to step through the list. The loop should pass each element to a function that displays the element whether it is a prime number.

13. Magic 8 Ball

Write a program that simulates a Magic 8 Ball, which is a fortune-telling toy that displays a random response to a yes or no question. In the student sample programs for this book, you will find a text file named 8_ball_responses.txt. The file contains 12 responses, such as “I don’t think so”, “Yes, of course!”, “I’m not sure”, and so forth. The program should read the responses from the file into a list. It should prompt the user to ask a question, then display one of the responses, randomly selected from the list. The program should repeat until the user is ready to quit.

Contents of 8_ball_responses.txt:

Yes, of course!Without a doubt, yes.You can count on it.For sure!Ask me later.I’m not sure.I can’t tell you right now.I’ll tell you after my nap.No way!I don’t think so.Without a doubt, no.The answer is clearly NO.

14. Expense Pie Chart

Create a text file that contains your expenses for last month in the following categories:

- Rent
- Gas
- Food
- Clothing
- Car payment
- Misc

Write a Python program that reads the data from the file and uses matplotlib to plot a pie chart showing how you spend your money.

15. 1994 Weekly Gas Graph

In the student sample programs for this book, you will find a text file named 1994_Weekly_Gas_Averages.txt. The file contains the average gas price for each week in the year 1994. (There are 52 lines in the file.) Using matplotlib, write a Python program that reads the contents of the file then plots the data as either a line graph or a bar chart. Be sure to display meaningful labels along the *X* and *Y* axes, as well as the tick marks.