

13. Assume the variable `dct` references a dictionary. Write code that pickles the dictionary and saves it to a file named `mydata.dat`.
14. Write code that retrieves and unpickles the dictionary that you pickled in Algorithm Workbench 11.

Programming Exercises

1. Course information

Write a program that creates a dictionary containing course numbers and the room numbers of the rooms where the courses meet. The dictionary should have the following key-value pairs:

| Course Number (key) | Room Number (value) |
|---------------------|---------------------|
| CS101 | 3004 |
| CS102 | 4501 |
| CS103 | 6755 |
| NT110 | 1244 |
| CM241 | 1411 |

The program should also create a dictionary containing course numbers and the names of the instructors that teach each course. The dictionary should have the following key-value pairs:

| Course Number (key) | Instructor (value) |
|---------------------|--------------------|
| CS101 | Haynes |
| CS102 | Alvarado |
| CS103 | Rich |
| NT110 | Burke |
| CM241 | Lee |

The program should also create a dictionary containing course numbers and the meeting times of each course. The dictionary should have the following key-value pairs:

| Course Number (key) | Meeting Time (value) |
|---------------------|----------------------|
| CS101 | 8:00 a.m. |
| CS102 | 9:00 a.m. |
| CS103 | 10:00 a.m. |
| NT110 | 11:00 a.m. |
| CM241 | 1:00 p.m. |

The program should let the user enter a course number, then it should display the course's room number, instructor, and meeting time.



VideoNote
The Capital Quiz
Problem

2. Capital Quiz

Write a program that creates a dictionary containing the U.S. states as keys, and their capitals as values. (Use the Internet to get a list of the states and their capitals.) The program should then randomly quiz the user by displaying the name of a state and asking the user to enter that state's capital. The program should keep a count of the number of correct and incorrect responses. (As an alternative to the U.S. states, the program can use the names of countries and their capitals.)

3. File Encryption and Decryption

Write a program that uses a dictionary to assign “codes” to each letter of the alphabet. For example:

```
codes = { 'A' : '%', 'a' : '9', 'B' : '@', 'b' : '#', etc... }
```

Using this example, the letter A would be assigned the symbol %, the letter a would be assigned the number 9, the letter B would be assigned the symbol @, and so forth.

The program should open a specified text file, read its contents, then use the dictionary to write an encrypted version of the file's contents to a second file. Each character in the second file should contain the code for the corresponding character in the first file.

Write a second program that opens an encrypted file and displays its decrypted contents on the screen.

4. Unique Words

Write a program that opens a specified text file then displays a list of all the unique words found in the file.

Hint: Store each word as an element of a set.

5. Word Frequency

Write a program that reads the contents of a text file. The program should create a dictionary in which the keys are the individual words found in the file and the values are the number of times each word appears. For example, if the word “the” appears 128 times, the dictionary would contain an element with 'the' as the key and 128 as the value. The program should either display the frequency of each word or create a second file containing a list of each word and its frequency.

6. File Analysis

Write a program that reads the contents of two text files and compares them in the following ways:

- It should display a list of all the unique words contained in both files.
- It should display a list of the words that appear in both files.
- It should display a list of the words that appear in the first file but not the second.
- It should display a list of the words that appear in the second file but not the first.
- It should display a list of the words that appear in either the first or second file, but not both.

Hint: Use set operations to perform these analyses.

7. World Series Winners

In this chapter's source code folder (available on the Computer Science Portal at www.pearsonhighered.com/gaddis), you will find a text file named `WorldSeriesWinners.txt`. 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. There are entries in the file indicating this.)

Write a program that reads this file and creates a dictionary in which the keys are the names of the teams, and each key's associated value is the number of times the team has won the World Series. The program should also create a dictionary in which the keys are the years, and each key's associated value is the name of the team that won that year.

The program should prompt the user for a year in the range of 1903 through 2009. It should then display the name of the team that won the World Series that year, and the number of times that team has won the World Series.

8. Name and Email Addresses

Write a program that keeps names and email addresses in a dictionary as key-value pairs. The program should display a menu that lets the user look up a person's email address, add a new name and email address, change an existing email address, and delete an existing name and email address. The program should pickle the dictionary and save it to a file when the user exits the program. Each time the program starts, it should retrieve the dictionary from the file and unpickle it.

9. Blackjack Simulation

Previously in this chapter you saw the `card_dealer.py` program that simulates cards being dealt from a deck. Enhance the program so it simulates a simplified version of the game of Blackjack between two virtual players. The cards have the following values:

- Numeric cards are assigned the value they have printed on them. For example, the value of the 2 of spades is 2, and the value of the 5 of diamonds is 5.
- Jacks, queens, and kings are valued at 10.
- Aces are valued at 1 or 11, depending on the player's choice.

The program should deal cards to each player until one player's hand is worth more than 21 points. When that happens, the other player is the winner. (It is possible that both players' hands will simultaneously exceed 21 points, in which case neither player wins.) The program should repeat until all the cards have been dealt from the deck.

If a player is dealt an ace, the program should decide the value of the card according to the following rule: The ace will be worth 11 points, unless that makes the player's hand exceed 21 points. In that case, the ace will be worth 1 point.

10. Word Index

Write a program that reads the contents of a text file. The program should create a dictionary in which the key-value pairs are described as follows:

- Key. The keys are the individual words found in the file.
- Values. Each value is a list that contains the line numbers in the file where the word (the key) is found.

For example, suppose the word “robot” is found in lines 7, 18, 94, and 138. The dictionary would contain an element in which the key was the string “robot”, and the value was a list containing the numbers 7, 18, 94, and 138.

Once the dictionary is built, the program should create another text file, known as a word index, listing the contents of the dictionary. The word index file should contain an alphabetical listing of the words that are stored as keys in the dictionary, along with the line numbers where the words appear in the original file. Figure 9-2 shows an example of an original text file (Kennedy.txt) and its index file (index.txt).

Figure 9-2 Example of original file and index file

