Chapter 7

Files

Files provide permanent storage of data. Any program that retains data from one execution of the program to another stores that data in files, whether the user can see those files or not. For example, any program that has a "preferences" tab that allows you to change some features of the program must store those preferences in a file. In this chapter we will look at ways to incorporate files into our programs, using files both for input (reading data from a file) and output (storing data in a file).

7.1 Concepts

There are two basic types of files: *text* files, which consist of a sequence of characters, and *binary* files, which contain binary encodings of information. Microsoft Word, and many other commercial programs that format information, create binary files because data can usually be packed much more efficiently into binary files than into text files. A Python program can open up binary files, but in order to obtain useful information from a binary file you need to know the way it is encoded; this information may or may not be available to you. In this chapter we will only look at text files. You can create a text file with the Python editor. Most word processing programs, such as Microsoft Word, also enable you to save files in text format.

Since files exist apart from your programs, an extra step is needed to make a file available to your program. This is called *opening* the file. You must always open a file before using it. This provides a link between the physical file and your program. When you open a file, you must say how you are going to use it: whether you want to read data out of it or write data into it. You may want to do both, but only one of these can be done at any given time.

The **open** function is

open(<**file** name>, <mode>)

where both the <**file** name> and <mode> are strings. This returns a *file object* that you need to reference each time you use the file, so you should save it in a variable, as in

F = **open**(<**file** name>, <mode>)

In this open statement the mode can be one of three strings:

- "r" means you want to open an existing file to read information from it. Your program will crash if the file is not found.
- "w" means you want to create a new file so you can write information into it. Any previous file with this file name will be destroyed.
- "a" means you want to append information at the end of a new or existing file.

For example, we could open for reading a file whose name is "data.txt" with the statement

F = **open**("data.txt", "r")

Note that to open a file you need to know the full name, including any file extension such as ".txt" Many modern software packages hide the file extension; you need to know it in order to write a program that opens a file with this extenstion. If you create a text file to use in a Python program, use a standard, explicit file extension, such as ".txt".

It is possible to specify a full path to a file, such as

```
F=open("C:\Documents and Settings\Bob\My Documents\data.txt","r")
```

If the file is in the same folder as the Python program that uses it, you only need to give the file name rather than its full path. For example, we might use

```
F=open("data.txt","r")
```

if the "data.txt" file is in the same folder as the program. We will later see a graphical way to obtain the path to a file; for now we will just work with files in the same folder as our programs.

With other data structures there are simple ways to move around in the structures. We call these "random access" structures because we can move directly to arbitrary positions in the structure. With files random access is much more difficult. We generally read files from beginning to end. When writing files we also write from beginning to end, rather than inserting characters into the middle of a file. This makes some file manipulations awkward. It is usually easier to work with lists and other structures internal to Python than with files. A simple paradigm for handling files is:

- Read the data from your file into a list or other structure at the start of your program.
- Manipulate the list or other structures.
- At the end of the program write the data from your structures back into the file.