FILES AND THE OPEN() AND FILE() BUILT IN FUNCTION

File access is one of the more important aspects of a language once you are comfortable with the syntax; there is nothing like the power of persistent storage to get some real work done.

How to Open a File

```
handle = open(file_name, access_mode = 'r')
```

The file_name variable contains the string name of the file we wish to open, and access_mode is either 'r' for read, 'w' for write, or 'a' for append. Other flags that can be used in the access_mode string include the '+' for dual read-write access and the 'b' for binary access. If the mode is not provided, a default of read-only ('r') is used to open the file.

If open() is successful, a file object will be returned as the handle (handle). All succeeding access to this file must go through its file handle. Once a file object is returned, we then have access to the other functionality through its methods such as readlines() and close(). Methods are attributes of file objects and must be accessed via the dotted attribute notation (see the following Core Note).

Here is some code that prompts the user for the name of a text file, then opens the file and displays its contents to the screen:

```
filename = raw_input('Enter file name: ')
fobj = open(filename, 'r')
for eachLine in fobj:
print eachLine,
fobj.close()
```

- ✓ Rather than looping to read and display one line at a time, our code does something a little different. We read all lines in one fell swoop, close the file, and then iterate through the lines of the file. One advantage to coding this way is that it permits the file access to complete more quickly. The output and file access do not have to alternate back and forth between reading a line and printing a line.

- ✓ It is cleaner and separates two somewhat unrelated tasks. The caveat here is the file size. The code above is reasonable for files with reasonable sizes. Very large data files may take up too much memory, in which case you would have to revert back to reading one line at a time.
- ✓ The other interesting statement in our code is that we are again using the comma at the end of the print statement to suppress the printing of the NEWLINE character. Why? Because each text line of the file already contains NEWLINEs at the end of every line.

- ✓ If we did not suppress the NEWLINE from being added by print, our display would be double-spaced. The file() built-in function was recently added to Python. It is identical to open(), but is named in such a way to indicate that is a factory function (producing file objects), similar to how int() produces integers and dict() results in dictionary objects.