# Reading from a Text File

Unlike other data, files have a life outside of your program. To access a file from a program you need to *open* it. There are several ways to open a file depending on how you plan to use it. Right now we only want to read data from a file, so we open it with the line

```
        F = open( <file name>, "r")
```

Here the file name should be a string. For example, to open the Python file foo.py you should say

```
        F = open( "foo.py", "r")
```

You can then use variable F in a for-loop as a sequence of lines.  We iterate through F with the loop

```
    for line in F:
        <do something with the line>
```

*Text files* are files that consist of a sequence of characters. Most text files are divided into lines that are terminated by the '\n' character. For example, a file that contains 3 names might have the characters

```
wally\nalice\ndilbert\n
```

If you open this file up in a text editor or word processor, most software packages are smart enough to recognize the line breaks and print it as

```
wally
alice
dilbert
```

so you don't actually see the linebreaks.

When python reads a text file with a for-loop, it includes the '\n' character at the end of each line. In other words, if you read the previous file with the code

```
for line in F:
    <do something with line>
```

then the first line will be the string "wally\n", the second "alice\n" and the third "dilbert\n".

You can see this if you print the file in the for-loop:

```
for line in F:
    print(line)
```

The newline character '\n' is printed as part of the variable line, and the print( ) function prints another newline, so the file appears to be double-spaced.

If we are getting numeric data from the file we can usually ignore the '\n' characters. If we are reading text, those characters get in the way. An easy way to eliminate them from variable **line** is to use the strip( ) method for strings. This removes characters from the front and back of a string. For example

```
        strippedLine = line.strip( "\n" )
```

will remove the newline markers and nothing else.

If you aren't worried about indentation on the line,

```
strippedLine = line.strip( )
```

will remove all white space, including the newline markers, from both the beginning and the end of the line.