Reading From Text Files

It is often impractical, and prone to error, to process large amounts of data by entering each value one-by-one via the keyboard. In this case, it is much easier to read data that is stored in a file. Python provides built-in methods for reading from files.

To open a file for reading, use the `open()` function, assigning the file object to a variable. Two arguments are required: the name of the file, and a mode for input/output (in this case, “`r`” for reading). Note that if the file does not exist, or cannot be opened, then the program will crash. A typical call might look like this.

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

Python provides several functions for reading data: `readline()`, which reads a single line of data from the file; `readlines()`, which reads the entire file into a list; and `read()`, which reads the entire file into a string. The end of a file is signified with an empty string, `""`. Like keyboard input, all data read from a text file are strings. To read numeric values, it is necessary to convert them to integers or floats using the `int` or `float` functions. If it is certain that all data are integers, a list comprehension provides a fast way to do this.

```
L = f.readlines()
L = [int(L) for val in L]
```

If data are not guaranteed to be integers, however, this will crash the program when it encounters non-numeric data. In this case, additional code will be necessary to process each value one-by-one.

It is important to close a file using `close()` once all data has been read, to free up memory. Alternatively, using the `with` keyword will automatically close the file, even if the program crashes.

Common Commands For Reading From Text Files

Description	Command
Open a text file for reading and assign to a file object, f.	`f = open("FILENAME", "r")`

Alternate method, using `with` (closes file automatically).	`with open("FILENAME", "r") as f:`
	`   # CODE GOES HERE`

Read a single line from a text file into a string, s.	`s = f.readline()`
Read the entire file into a single string, s.	`s = f.read()`
Read the entire file into a list, L, each line is an element.	`L = f.readlines()`

Loop over every line in a text file sequentially.	`for line in f:`
	`   # CODE GOES HERE`

Close a file when finished.	`f.close()`

Reading From Text Files

Answer the following questions.

1. What are the three main steps involved in working with files?

2. A file contains a sequence of integer values. Explain why the following code would cause the program to crash.

```
input_file = open("values.txt", "r")
total = 0
for value in input_file:
      total += value
print(total)
```

Write programs that accomplish each task. Use proper conventions for variable names, input prompts, output statements, and program structure. You may assume that all files contain data in the formats specified. You may download the file `read_files.zip` for testing.

3. Read a text file containing information about a character in a role-playing game, and print the information in a nicely-formatted manner. The first line contains a string representing the player's name. The next three lines are integers representing the player's Strength, Agility and Luck. The last line contains a comma-separated sequence of strings representing items in the player’s inventory. Store these in a list. The "split" command might be useful here.

4. Read integers from a file (one per line), and determine the number of integers that are multiples of three.

5. Read integers between 1-10 from a file (one per line), and generate a tally of the frequencies of each value. Display these frequencies in a table.

6. Read names (strings) from a file, and generate a list with all duplicates omitted. Count the number of vowels in the unique names.

7. A file is used to keep track of a grocery store’s inventory of items. Each item has two entries in the file. The first is a string with its description (e.g. “Garvin’s Baked Beans, 14 oz”) and the second is an integer representing the quantity in stock. Read the file and print the names and quantities of all items that are out of stock (quantity is zero), low stock (less than 10) or overstock (greater than 50), in a nicely-formatted manner.