Introduction to File Input and Output

When a program needs to save data for later use, it writes the data in a file. The data can be read from the file at a later time.

The programs you have written so far require the user to reenter data each time the program runs, because data that is stored in RAM (referenced by variables) disappears once the program stops running. If a program is to retain data between the times it runs, it must have a way of saving it. Data is saved in a file, which is usually stored on a computer's connected disk drives. Once the data is saved in a file, it will remain there after the program stops running or the computer is shut off. Data that is stored in a file can be retrieved and used at a later time.

Commercial software packages store data in files.

Here are a few examples.

Word processors. Word processing programs are used to write letters, memos, reports, and other types of documents. The documents are then saved in files so they can be edited and printed.

Image/ Graphic editors. Image editing programs are used to draw graphics and edit images such as the ones that you take with a digital camera. The images that you create or edit with an image editor are saved in files.

• **Spreadsheets**. Spreadsheet programs are used to work with numerical data. Numbers and mathematical formulas can be inserted into the rows and columns of the spreadsheet. The spreadsheet can then be saved in a file for use later.

• **Games**. Many computer games keep data stored in files. For example, some games keep a list of player names with their scores stored in a file. These games typically display the players' names in order of their scores, from highest to lowest. Some games also allow you to save your current game status in a file so you can quit the game and then resume playing it later without having to start from the beginning.

• **Web browers**. Sometimes when you visit a Web page, the browser stores a small file known as a cookie on your computer. Cookies typically contain information about the browsing session, such as the contents of a shopping cart.

Programs that are used in daily business operations rely extensively on files. Payroll programs keep employee data in files, inventory programs keep data about a company’s products in files, accounting systems keep data about a company’s financial operations in files, and so on. Programmers usually refer to the process of saving data in a file as “writing data to” the file. When a piece of data is written to a file, it is copied from a variable in RAM to the file.

The process of retrieving data from a file is known as “reading data from” the file. When a piece of data is read from a file, it is copied from the file into RAM and referenced by a variable.

The term input file is used to describe a file that data is read from. It is called an input file because the program gets data from the file.

There are always three steps that must be taken when a file is used by a program.

1. Open the file—Opening a file creates a connection between the file and the program. Opening an output file usually creates the file on the disk and allows the program to write data to it. Opening an input file allows the program to read data from the file.

2. Process the file—In this step data is either written to the file (if it is an output file) or read from the file (if it is an input file).

3. Close the file—When the program is finished using the file, the file must be closed. Closing a file disconnects the file from the program.

Types of Files

In general, there are two types of files: text and binary. A text file contains data that has been encoded as text, using a scheme such as ASCII or Unicode. Even if the file contains numbers, those numbers are stored in the file as a series of characters. As a result, the file may be opened and viewed in a text editor such as Notepad. A binary file contains data that has not been converted to text. The data that is stored in a binary file is intended only for a program to read. As a consequence, you cannot view the contents of a binary file with a text editor.

Python allows you to work both text files and binary files. You will be able to use a text editor to

examine the files that your programs create.

File Access Methods

Most programming languages provide two different ways to access data stored in a file: sequential access and direct access. When you work with a sequential access file, you access data from the beginning of the file to the end of the file. If you want to read a piece of data that is stored at the very end of the file, you have to read all of the data that comes before it. In a Sequential File, you cannot jump directly to the desired data. This is similar to the way cassette tape or old VHS tape players work. If you want to listen to the last song on a cassette tape, you have to either fast-forward over all of the songs that come before it or listen to them. There is no way to jump directly to a specific song.

When you work with a direct access file (which is also known as a random access file), you can jump directly to any piece of data in the file without reading the data that comes before it. This is similar to the way a CD player or an MP3 player works. You can jump directly to any song that you want to listen to.

Sequential access files are easy to work with, and you can use them to gain an understanding of basic file operations.

Filenames and File Objects

Files are identified by a filename. When you create a document with a word processor and then save the document in a file, you have to specify a filename. When you use File Management Software to examine the contents of your disk, you see a list of filenames

Each operating system has its own rules for naming files. Many systems support the use of filename extensions, which are short sequences of characters that appear at the end of a filename preceded by a period (which is known as a “dot”). The extension usually indicates the type of data stored in the file. For example, the .jpg extension usually indicates that the file contains a graphic image that is compressed according to the JPEG image standard. The .txt extension usually indicates that the file contains text. The .doc extension (as well as the .docx extension) usually indicates that the file contains a Microsoft Word document.

In order for a program to work with a file on the computer’s disk, the program must create a file object in memory. A file object is an object that is associated with a specific file and provides a way for the program to work with that file. In the program, a variable references the file object. This variable is used to carry out any operations that are performed on the file

Opening a File

You use the open function in Python to open a file. The open function creates a file object and associates it with a file on the disk. Here is the general format of how the open function is used:

```
file_variable = open(filename, mode)
```

General Format:

file_variable is the name of the variable that will reference the file object.

filename is a string specifying the name of the file.

mode is a string specifying the mode (reading, writing, etc.) in which the file will be opened.

Some of the Python file modes

Mode Types and their Description

'r'	Open a file for reading only. The file cannot be changed or written to.
'w'	Open a file for writing. If the file already exists, erase its contents. If it does not exist, create it.
'a'	Open a file to be written to. All data written to the file will be appended to it end. If the file does not exist, create it.

The file customers.txt contains customer data, and we want to open it for reading.

Here is an example of the opening a file for input

customer_file = open('customers.txt', 'r')

After this statement executes, the file named customers.txt will be opened, and the variable customer_file will reference a file object that we can use to read data from the file. Suppose we want to create a file named sales.txt and write data to it.

Here is an example of the opening a file for input

sales_file = open('sales.txt', 'w')

After this statement executes, the file named sales.txt will be created, and the variable sales_file will reference a file object that we can use to write data to the file.

Remember, when you use the 'w' mode you are creating the file on the disk. If a file with the specified name already exists when the file is opened, the contents of the existing file will be erased.

Specifying the Location of a File

When you pass a file name that does not contain a path as an argument to the open function, the Python interpreter assumes that the file’s location is the same as that of the program. For example, suppose a program is located in the following folder on a Windows computer:

C:\Users\Blake\Documents\Python

If the program is running and it executes the following statement, the file test.txt is created

in the same folder:

test_file = open('test.txt', 'w')

If you want to open a file in a different location, you can specify a path as well as a filename in the argument that you pass to the open function. If you specify a path in a string literal (particularly on a Windows computer), be sure to prefix the string with the letter r.

Here is an example:

test_file = open(r'C:\Users\Blake\temp\test.txt', 'w')

This statement creates the file test.txt in the folder C:\Users\Blake\temp. The r prefix specifies that the string is a raw string. This causes the Python interpreter to read the backslash characters as literal backslashes. Without the r prefix, the interpreter would assume that the backslash characters were part of escape sequences, and an error would occur.

Writing Data to a File

You have worked with several of Python’s library functions

A method is a function that belongs to an object and performs some operation using that object. Once you have opened a file, you use the file object’s methods to perform operations on the file. File objects have a method named write that can be used to write data to a file.

Here is the general format of how you call the write method:

```
file_variable.write(string)
```

In the format, file_variable is a variable that references a file object, and string is a string that will be written to the file. The file must be opened for writing (using the 'w' or 'a' mode) or an error will occur. In our current example, customer_file references a file object, and the file was opened for writing with the 'w' mode. Here is an example of how we would write the string ‘Charles Pace’ to the file:

```
customer_file.write('Charles Pace')
```

Here is another example:

```
name = 'Charles Pace'

customer_file.write(name)
```

The second statement writes the value referenced by the name variable to the file associated with customer_file. In this case, it would write the string ‘Charles Pace’ to the file. (These examples show a string being written to a file, but you can also write numeric values.) Once a program is finished working with a file, it should close the file. Closing a file disconnects the program from the file. In some systems, failure to close an output file can cause a loss of data. This happens because the data that is written to a file is first written to a buffer, which is a small “holding section” in memory. When the buffer is full, the system writes the buffer’s contents to the file. This technique increases the system’s performance, because writing data to memory is faster than writing it to a disk. The process of closing an output file forces any unsaved data that remains in the buffer to be written to the file.

In Python you use the file object’s close method to close a file. For example, the following statement closes the file that is associated with customer_file:

```
customer_file.close()
```

Reading Data From a File

If a file has been opened for reading (using the 'r' mode) you can use the file object’s read method to read its entire contents into memory. When you call the read method, it returns the file’s contents as a string. The file’s contents are read into memory as a string and assigned to the reference variable.

Although the read method allows you to easily read the entire contents of a file with one statement, many programs need to read and process the items that are stored in a file one at a time. For example, suppose a file contains a series of sales amounts, and you need to write a program that calculates the total of the amounts in the file. The program would read each sale amount from the file and add it to an accumulator.

In Python you can use the readline method to read a line from a file. (A line is simply a string of characters that are terminated with a \n.) The method returns the line as a string, including the \n. a blank line is displayed after each line in the output. This is because each item that is read from the file ends with a newline character (\n). Later you will learn how to remove the newline character.

When a file is opened for reading, a special value known as a read position is internally maintained for that file. A file’s read position marks the location of the next item that will be read from the file. Initially, the read position is set to the beginning of the file.

After the read statement executes, the read position for the file will be positioned If there is a \n in the file the file’s read position will be advanced to the next line in the file. If the last line in a file is not terminated with a \n, the readline method will return the line without a \n.

Concatenating a Newline to a String

In most cases, the data items that are written to a file are not string literals, but values in memory that are referenced by variables. This would be the case in a program that prompts the user to enter data and then writes that data to a file. When a program writes data that has been entered by the user to a file, it is usually necessary to concatenate a \n escape sequence to the data before writing it. This ensures that each piece of data is written to a separate line in the file. each name should have the \n escape sequence added to it when written to the file.

Reading a String and Stripping the Newline from It

Sometimes complications are caused by the \n that appears at the end of the strings that are returned from the readline method. in many cases you want to remove the \n from a string after it is read from a file. Each string in Python has a method named rstrip that removes, or “strips,” specific characters from the end of a string. (It is named rstrip because it strips characters from the right side of a string.) The following code shows an example of how the rstrip method can be used.

```
name = 'Joanne Manchester\n'

name = name.rstrip('\n')
```

The first statement assigns the string 'Joanne Manchester\n' to the name variable. Notice that the string ends with the \n escape sequence.) The second statement calls the name.rstrip('\n') method. The method returns a copy of the name string without the trailing \n. This string is assigned back to the name variable. The result is that the trailing \n is stripped away from the name string.

Appending Data to an Existing File

When you use the 'w' mode to open an output file and a file with the specified filename already exists on the disk, the existing file will be erased and a new empty file with the same name will be created. Sometimes you want to preserve an existing file and append new data to its current contents. Appending data to a file means writing new data to the end of the data that already exists in the file.

In Python you can use the 'a' mode to open an output file in append mode, which means the following.

- If the file already exists, it will not be erased. If the file does not exist, it will be created.
- When data is written to the file, it will be written at the end of the file's current contents.

For example, assume the file friends.txt contains the following names, each in a separate

line:

Joe

Rose

Geri

The following code opens the file and appends additional data to its existing contents.

```
myfile = open('friends.txt', 'a')

myfile.write('Matt\n')

myfile.write('Chris\n')

myfile.write('Suze\n')

myfile.close()
```

After this program runs, the file friends.txt will contain the following data:

Joe

Rose

Geri

Matt

Chris

Suze

Writing and Reading Numeric Data

Strings can be written directly to a file with the write method, but numbers must be converted to strings before they can be written. Python has a built-in function named str that converts a value to a string. For example, assuming the variable num is assigned the value 24, the expression str(num) will return the string '24'.

If we wanter to contatinate a line end to it, the expression str(num) + '\n' converts the value referenced by num1 to a string and concatenates the \n escape sequence to the string. In the program's sample run, the user entered 24 as the first number, so this expression produces the string '24\n'. As a result, the string '24\n' is written to the file.

When you read numbers from a text file, they are always read as strings. If you want to perform math operations on an inputted string, you must convert the string to a numeric data type. Python provides the built-in function int to convert a string to \ an integer and the built-in function float to convert a string to a floating-point number.

```
infile = open('numbers.txt', 'r')

string_input = infile.readline()

value = int(string_input)

infile.close()
```

The statement in line 2 reads a line from the file and assigns it to the string_input variable. As a result, string_input will reference the first line of data in the file. Then the statement in line 3 uses the int function to convert the string_input variable to an integer, and assigns the result to value. After this statement executes, the value variable will reference the integer value in the data stream. (Both the int and float functions ignore any \n at the end of the string that is passed as an argument.)

The previous code demonstrates the the basic framework involved in reading a string from a file with the readline method and then converting that string to an integer with the int function. In many situations, however, the code can be simplified. A better way is to read the string from the file and convert it in one statement, as shown here:

```
infile = open('numbers.txt', 'r')

value = int(infile.readline())

infile.close()
```

Notice in line 2 that a call to the readline method is used as the argument to the int function. Here’s how the code works: the readline method is called, and it returns a string. That string is passed to the int function, which converts it to an integer. The result is assigned to the value variable.

Using Loops to Process Files

Files usually hold large amounts of data, and programs typically use a loop to process the data in a file. Although some programs use files to store only small amounts of data, files are typically used to hold large collections of data. When a program uses a file to write or read a large amount of data, a loop is typically involved.

Reading a File with a Loop and Detecting the End of the File

Quite often a program must read the contents of a file without knowing the number of items that are stored in the file.

If you want to write a program that processes all of the items in the file. You can use a loop to read the items in the file, but you need a way of knowing when the end of the file has been reached.

In Python, the readline method returns an empty string ('') when it has attempted to read beyond the end of a file. This makes it possible to write a while loop that determines when the end of a file has been reached.

Here is the general algorithm, in pseudocode:

1. Open the file
2. Use readline to read the first line from the file
3. While the value returned from readline is not an empty string:
4. Process the item that was just read from the file
5. Use readline to read the next line from the file.
6. Close the file

In this algorithm we call the readline method just before entering the while loop. The purpose of this method call is to get the first line in the file, so it can be tested by the loop. This initial read operation is called a priming read.

Using Python's for Loop to Read Lines

In the previous example you saw how the readline method returns an empty string when the end of the file has been reached. Most programming languages provide a similar technique for detecting the end of a file. If you plan to learn programming languages other than Python, it is important for you to know how to construct this type of logic.

The Python language also allows you to write a for loop that automatically reads line in a file without testing for any special condition that signals the end of the file. The loop does not require a priming read operation, and it automatically stops when the end of the file has been reached. When you simply want to read the lines in a file, one after the other, this technique is simpler and more elegant than writing a while loop that explicitly tests for an end of the file condition.

Here is the general format of the loop:

```
for variable in file_object:
        statement
        statement
        etc.
```

In the general format, variable is the name of a variable and file_object is a variable that references a file object. The loop will iterate once for each line in the file. The first time the loop iterates, variable will reference the first line in the file (as a string), the second time the loop iterates, variable will reference the second line, etc.

Processing Records

The data that is stored in a file is frequently organized in records. A record is a complete set of data about an item, and a field is an individual piece of data within a record.

When data is written to a file, it is often organized into records and fields. A record is a complete set of data that describes one item, and a field is a single piece of data within a record. For example, suppose we want to store data about employees in a file. The file will contain a record for each employee. Each record will be a collection of fields, such as name, ID number, and department. Each time you write a record to a sequential access file, you write the fields that make up the record, one after the other When we read a record from a sequential access file, we read the data for each field, one after the other, until we have read the complete record. The loop starts over and this process continues until there are no more records to read.

Programs that store records in a file typically require more capabilities than simply writing and reading records. In the following. We will examine algorithms for adding records to a file, searching a file for specific records, modifying a record, and deleting a record.

When working with a sequential access file, it is necessary to copy the entire file each time one item in the file is modified. As you can imagine, this approach is inefficient, especially if the file is large. Other, more advanced techniques are available, especially when working with direct access files, that are much more efficient. When working with a sequential access file, it is necessary to copy the entire file each time one item in the file is deleted. As was previously mentioned, this approach is inefficient, especially if the file is large. Other, more advanced techniques are available, especially when working with direct access files, that are much more efficient.

Review Questions

1. What is an output file?
2. What is an input file?
3. What three steps must be taken by a program when it uses a file?
4. In general, what are the two types of files? What is the difference between these two types of files?
5. What are the two types of file access? What is the difference between these two?
6. When writing a program that performs an operation on a file, what two file associated names do you have to work with in your code?
7. If a file already exists what happens to it if you try to open it as an output file (using the 'w' mode)?
8. What is the purpose of opening a file?
9. What is the purpose of closing a file?
10. What is a file's read position? Initially, where is the read position when an input file is opened?
11. In what mode do you open a file if you want to write data to it, but you do not want to erase the file's existing contents? When you write data to such a file, to what part of the file is the data written?
12. Write a short program that uses a for loop to write the numbers 1 through 10 to a file.
13. What does it mean when the readline method returns an empty string?
14. Assume that the file data.txt exists and contains several lines of text. Write a short program using the while loop that displays each line in the file.
15. Revise the program that you wrote using a while loop to use the for loop instead of the while loop.

16. What is a record?

17. What is a field?

18. Describe the way that you use a temporary file in a program that modifies a record in a sequential access file.

Describe the way that you use a temporary file in a program that deletes a record from a sequential file.