Chapter 10 — Files

Introduction

While working with computers, we all have worked with files and directories. A file is "a collection of data or information that has a name."[8] We can use files to store programs, text, and other structured information.

Throughout this chapter you will see files referred to as streams. You can think of a sequence of data arriving on a conveyor belt or like water in a stream. The file input/output methods we will be describing will work for many types of stream data.

> NOTE: If you are using the "grader" online system, you MUST include the line "`from BrowserFile import open`" at the top of your program. The BrowserFile module uses the browser's persistent storage to store the data read from and written to a file. This is required because a browser app written in JavaScript can't directly access files on the computer.

Objectives

Upon completion of this chapter's exercises, you should be able to:

- Employ files to retrieve and store plain text.
- Demonstrate different methods for reading sequential data.
- Modify an existing file to add new data to the end of it.
- Use Python's context manager to manage a finite resource.
- Utilize data in the CSV format in a program.

Prerequisites

This Chapter requires...

Save Text to a File

The first operation we will do with a file, is to create one. There are three basic operations required to

8 http://www.webopedia.com/TERM/F/file.html retrieved 2017-04-13

do this: 1) open the file, 2) write the data to the file, and 3) close the file.

When opening a file that is stored in the same folder as our Python program, we need just its name. If the file is stored in another location on the computer we need to specify the location and name of the file, this is called its path. The examples in this chapter will not add the complexity of storing files in other locations.

The first step in creating a file is opening it. To open a file we use the `open` function and pass it the file name or path and how we want to use the file. The most common file modes are 'r' for reading, 'w' for writing, and 'a' for appending new data.

`open(`*`file_path, mode`***`)`**	Function
Open a file at the specified path and return a file object. The mode is a string that specifies how the file is to be handled. Modes include: **Mode** — **Operation** 'r' — Open a text file for reading. The file pointer will be placed at the beginning of the file. 'w' — Create a new empty file for writing. If the file exists, clear the file. 'a' — Open a file for writing but preserve the existing data. The file pointer will be at the end of the file so that we can add new data. https://docs.python.org/3/library/functions.html#open	

The second thing we need to do is to write the data out to the stream. This is done with the write method on the file object. When writing lines of text, the system does not automatically add an end of line (EOL) character. In the example below you can see the strings have '\n' in them, which is the EOL character.

*`file`***`.write(`***`string`***`)`**	Method of file
Write a string to the file object. If a new line is needed be sure to append the "\n" character, or do a second write with it. https://docs.python.org/3/library/io.html#io.TextIOBase.write	

All file operations should be closed, when we are finished with the file. On most systems, input/output (I/O) operations are stored on the computer's memory and may not be written immediately to the hard disk. This is known as caching and speeds up disk access. In Python, we use the close method on the file object to finalize the disk operation and to release the system resources.

file.**close()**	Method of file
Commit the changes to the file, if there are any, and release the operating resources. https://docs.python.org/3/library/io.html#io.IOBase.close	

```
1| ##### output a text file
2| ##### Text of Moby Dick, by Herman Melville,
3| ##### downloaded from Project Guttenberg
4| path = "mobydick.txt"
5| f = open(path,"w")
6| f.write("Call me Ishmael.\n")
7| f.write("Some years ago—never mind how long precisely—having
     little or no money in my purse, and nothing particular to
     interest me on shore,")
8| f.write(" I thought I would sail about a little and see the
     watery part of the world.\n")
9| f.write("It is a way I have of driving off the spleen and
     regulating the circulation.\n")
10| f.close()
11| print(path, "written")
```

Text of Moby Dick, by Herman Melville, downloaded from Project Guttenberg.[9]

Using a Context Manger

Python allows for objects to create a special context at run-time. When objects have been created to take advantage of this feature, you can use the `with` statement to create and automatically finalize or close them. All the input/output examples in this chapter and the remainder of this book will perform these operations in a `with` context.

with *object(...)* **as** *variable*: *suite*	Statement
The with statement wraps a suite of code into a managed context. Exception handling (using try/except) may be kept local to errors within the block. By using a context manager, it insures that the object is closed once the suite is complete. https://docs.python.org/3/reference/compound_stmts.html#the-with-statement	

9 http://www.gutenberg.org/files/2701/2701-0.txt

The following program re-writes the program that creates the Moby Dick text file, using a context manager. It is the preferred way to do input/output.

```
1| ##### output a text file - THE BETTER WAY
2| ##### Text of Moby Dick, by Herman Melville,
3| ##### downloaded from Project Guttenberg
4| fn = "mobydick.txt"
5| with open(fn,"w") as f:
6|     f.write("Call me Ishmael.\n")
7|     f.write("Some years ago—never mind how long precisely—having
   little or no money in my purse, and nothing particular to
   interest me on shore,")
8|     f.write(" I thought I would sail about a little and see the
   watery part of the world.\n")
9|     f.write("It is a way I have of driving off the spleen and
   regulating the circulation.\n")
10| print(fn, "written")
```

Read Text from a File

Using for

The easiest way to read the lines from a text file is to use the for statement. If we open a file for reading "r", we can simply treat the file object like a list and get each string (line) directly from the file. In the example above, the variable line will be set to each line. Remember a line may be written out by multiple calls to write and ends when we find a newline "\n".

```
1| with open("mobydick.txt") as file:
2|     for line in file:
3|         print("line=", line)
```

```
line= Call me Ishmael.

line= Some years ago—never mind how long precisely—having little
or no money in my purse, and nothing particular to interest me
on shore, I thought I would sail about a little and see the
watery part of the world.

line= It is a way I have of driving off the spleen and
regulating the circulation.
```

When a line is read in it will contain the "\n" new line character on the end. You may want to use the `.rstrip()` string method to remove it.

```
with open("mobydick.txt") as file:
    for line in file:
        line = line.rstrip()
        print("line=", line)
```

Using readline

You may also use the `readline` method of the file to read one line at a time. The newline character will be included on the end of all lines that are successfully read, even blank ones. When `newline` reaches the end of the file an empty string will be returned, signaling the end.

file.**readline()**	Method of file
Read the file up-to and including the new line character and return it as a string. If the stream is at its end, return an empty string. https://docs.python.org/3/library/io.html?highlight=readline#io.IOBase.readline	

```
with open("mobydick.txt") as file:
    line = file.readline()
    while line:
        line = line.rstrip()
        print("line=", line)
        line = file.readline()
```

Appending Text to a File

In addition to writing to a new file and reading from a file, it is often necessary to add additional data to the end of a file. We call that process appending. The open function has a mode "a" that opens a file for writing, leaves the existing data, and moves the write pointer to the end.

```
###### append to a text file
##### Text of Moby Dick, by Herman Melville,
##### downloaded from Project Guttenberg
fn = "mobydick.txt"
with open(fn,"a") as stream:
```

```
 6|     stream.write("Whenever I find myself growing grim about the"
 7|         " mouth; whenever it is a damp, drizzly November in my"
 8|         " soul; whenever I find myself involuntarily pausing"
 9|         " before coffin warehouses, and bringing up the rear of"
10|         " every funeral I meet; and especially whenever my"
11|         " hypos get such an upper hand of me, that it requires"
12|         " a strong moral principle to prevent me from"
13|         " deliberately stepping into the street, and"
14|         " methodically knocking people's hats off—then, I"
15|         " account it high time to get to sea as soon as I can.\
    n")
16| print(fn, "line appended")
```

Writing Comma Separated Values (CSV)

Comma Separated Values (CSV) …..
Discuss dialects, but the default 'execll'....

csv	Module
The csv module will read and write stream data and will convert between comma delimited text and lists or dictionaries. The module can be configured to read and write virtually any comma delimited text, by default it understands the "Excell" variant. https://docs.python.org/3/library/csv.html#module-csv	

csv.**writer(***file_object***)**	Class in csv
The `.writer` is a class in the csv module that will be used to write data to a stream. The .writer object will not be passed data directly but a method of the csv.writer will. Be sure when you open a file for writing CSV data, to include the "newline="" argument of the open function. The CSV module handles end of line and the automatic end of line handling needs to be turned off. https://docs.python.org/3/library/csv.html#csv.writer	

csv_writer.**writerow(***sequence***)**	Method of csv.writer
The .writerow method of the csv.writer will accept a sequence of data (a list, tuple, set...) and will format it as csv text and then write it put to the stream. https://docs.python.org/3/library/csv.html#csv.csvwriter.writerow	

```
1| import csv
2| with open("test.csv","w", newline='') as stream:
3|     writer = csv.writer(stream)
4|     writer.writerow([1, 2, 3])
5|     writer.writerow([-234.78, 0.0, 345.7])
6|     writer.writerow(['a string', 'another string', '$%^*#@!'])
7|     writer.writerow(["str with, comma", 'str with "quote"', ""])
8| print("done")
```

The file "test.csv" should contain:

```
1,2,3\n
-234.78,0.0,345.7\n
a string,another string,$%^*#@!\n
"str with, comma","str with ""quote""",\n
```

Reading CSV

Need description

csv.**reader(***file_object***)**	Class in csv
The .reader is another class within the csv module. Like the writer, an instance of this class is used to read csv formatted data from a stream. By default, the reader will return a list of strings even if the data consisted of numbers and quotes were not used in the actual input stream. Be sure when you open a file for writing CSV data, to include the "newline="" argument of the open function. The CSV module handles end of line and the automatic end of line handling needs to be turned off.	

https://docs.python.org/3/library/csv.html#csv.reader

The following program reads a file in CSV format, that you specify, and displays the data in the file. Pay special attention that the original data used above include integer, float and string values. The default result of a `csv.reader` object is a collection of strings.

```
1| import csv
2| file = input("Enter file name>>")
3| with open(file, newline='') as csvfile:
4|     rdr = csv.reader(csvfile)
5|     for data in rdr:
6|         print(data)
7| print("end", file)
```

```
8| Enter file name>>test.csv
9| ['1', '2', '3']
10| ['-234.78', '0.0', '345.7']
11| ['a string', 'another string', '$%^*#@!']
12| ['a string with, a comma', 'a string with a "quote"', '']
13| end test.csv
```


Sample Program — Mega Millions Lottery

This sample uses file I/O and the csv module to create a file of lottery numbers and to read through them counting the number of times a ball is drawn.

The first program will create the file for later reading.

```
1| # mega millions lottery numbers
2| # original data from New York Lottery Commission
3| # data in format of  ['Draw Date','Ball One','Ball Two',
4| # 'Ball Three','Ball Four','Ball Five','Mega Ball','Multiplier']
5| mega = [
6| ['08/31/2018',7,18,29,32,45,17,3],
7| ['09/04/2018',2,7,25,35,44,3,3],
8| ['09/07/2018',8,10,41,54,68,10,2],
9| ['09/11/2018',15,30,51,62,67,19,2],
10| ['09/14/2018',23,30,40,43,66,13,4],
```

```
['09/18/2018',31,32,43,63,68,17,2],
['09/21/2018',1,2,11,52,64,9,4],
['09/25/2018',8,16,32,48,61,12,2],
['09/28/2018',39,45,52,56,59,15,3],
['10/02/2018',2,22,29,31,34,1,3],
['10/05/2018',27,28,32,41,69,12,2],
['10/09/2018',20,22,39,54,60,18,3],
['10/12/2018',4,24,46,61,70,7,3],
['10/16/2018',3,45,49,61,69,9,5],
['10/19/2018',15,23,53,65,70,7,2],
['10/23/2018',5,28,62,65,70,5,3],
['10/26/2018',1,28,61,62,63,5,4],
['10/30/2018',20,31,39,46,49,23,2],
['11/02/2018',3,23,28,46,62,16,2],
['11/06/2018',28,34,37,56,69,12,2],
['11/09/2018',8,14,27,57,67,5,4],
['11/13/2018',34,46,57,65,69,11,3],
['11/16/2018',33,36,63,68,69,16,3],
['11/20/2018',10,16,31,42,66,10,3]
]
import csv
with open("mega.csv","w", newline='') as f:
    fw = csv.writer(f)
    for m in mega:
        fw.writerow(m)
print("done")
```

The file mega.csv should contain:

```
08/31/2018,7,18,29,32,45,17,3\n
09/04/2018,2,7,25,35,44,3,3\n
09/07/2018,8,10,41,54,68,10,2\n
09/11/2018,15,30,51,62,67,19,2\n
...
11/20/2018,10,16,31,42,66,10,3\n
```

Now let's write a program to open the mega.csv file and count the number of times that a white ball has been drawn. Looking at the data the 5 white balls are at index 1-5. Remember that the `csv.reader` returns a list of strings, and we need to convert them to integer ball numbers.

```
import csv
try:
    ball = int(input("enter ball number to count>> "))
```

```
4|  except:
5|      print("please enter an integer ball number")
6|
7|  with open("mega.csv","r",newline='') as f:
8|      fr = csv.reader(f)
9|      kount = 0
10|     for drawing in fr:
11|         # loop through the 5 columns with white balls
12|         for index in range(1,6):
13|             if(int(drawing[index])==ball):
14|                 kount = kount + 1
15|
16| print("the white", ball, "ball was drawn", kount, "times")
```

```
enter ball number to count>> 22
the white 22 ball was drawn 2 times
```


Summary

Goes here

Important Terms

- CSV
- append
- cache
- close
- context
- file
- open
- reader
- readline
- stream
- text
- with
- write
- writer

Exercises

Here

Word Search

```
· i c h t s q j v x c j
n l a z w t o e m a o w
j q c t r r p g v f n r
c m h f i e e b w q t y
s r e a t a n n i d e f
v t j z e m c i d w x v
l w r i t e r t e x t k
y o i r e a d l i n e i
o h k r e a d e r d h n
h d m j d a p p e n d a
· c l o s e p u o i k ·
w i t h j x f i l e h u
```

CSV, append, cache, close, context, file, open, reader, readline, stream, text, with, write, writer

References

https://en.wikipedia.org/wiki/Stream_(computing)

https://catalog.data.gov/dataset/lottery-mega-millions-winning-numbers-beginning-2002/resource/46ff28c6-040c-48c9-bbb1-b7e90dfcd97e source New York Lottery Commission