[Reading and Writing Files in Python](http://www.pythonforbeginners.com/systems-programming/reading-and-writing-files-in-python/)***5***

8 Jul 2013   | [System & OS](http://www.pythonforbeginners.com/category/systems-programming/)

Tags: [Files](http://www.pythonforbeginners.com/tag/files/) · [IO](http://www.pythonforbeginners.com/tag/io/)

Overview

In Python, you don't need to import any library to read and write files.

The first step is to get a file object.

The way to do this is to use the open function.

File Types

A file is usually categorized as either text or binary.

A text file is often structured as a sequence of lines and a line is a sequence

of characters.

The line is terminated by a EOL (End Of Line) character.

The most common line terminator is the \n , or the newline character.

The backslash character indicates that the next character will be treated as a

newline.

A binary file is basically any file that is not a text file. Binary files can

only be processed by application that know about the file's structure.

Open ( )

To open a file for writing use the built-i open() function. open() returns a

file object, and is most commonly used with two arguments.

The syntax is:

file\_object = open(filename, mode) where file\_object is the variable to put the

file object.

The second argument describes the way in which the file will be used.

Mode

The mode argument is optional; 'r' will be assumed if it’s omitted.

**The modes can be:**

'r' when the file will only be read

'w' for only writing (an existing file with the same name will be erased)

'a' opens the file for appending; any data written to the file is automatically

added to the end.

'r+' opens the file for both reading and writing.

|  |  |
| --- | --- |
| 1  2 | >>> f = open('workfile', 'w')  >>> print f |

Next the file objects functions can be called. The two most common functions are

read and write.

Create a text file

Let's first create a new text file. You can name it anything you like,

in this example we will name it "newfile.txt".

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | file = open("newfile.txt", "w")    file.write("hello world in the new file\n")    file.write("and another line\n")    file.close() |

If we now look in the newfile.txt, we can see the text that we wrote:

|  |  |
| --- | --- |
| 1  2  3 | $ cat newfile.txt  hello world in the new file  and another line |

How to read a text file

To read a file, we can use different methods.

file.read( )

If you want to return a string containing all characters in the file, you can

use file.read().

|  |  |
| --- | --- |
| 1  2  3 | file = open('newfile.txt', 'r')    print file.read() |

**Output:**

hello world in the new file

and another line

We can also specify how many characters the string should return, by using

file.read(n), where "n" determines number of characters.

This reads the first 5 characters of data and returns it as a string.

|  |  |
| --- | --- |
| 1  2 | file = open('newfile.txt', 'r')  print file.read(5) |

**Output:**

hello

file.readline( )

The readline() function will read from a file line by line (rather than pulling

the entire file in at once).

Use readline() when you want to get the first line of the file, subsequent calls

to readline() will return successive lines.

Basically, it will read a single line from the file and return a string

containing characters up to \n.

|  |  |
| --- | --- |
| 1  2  3 | file = open('newfile.txt', 'r')    print file.readline(): |

**Output:**

hello world in the new file

file.readlines( )

readlines() returns the complete ﬁle as a list of strings each separated by \n

|  |  |
| --- | --- |
| 1  2  3 | file = open('newfile.txt', 'r')    print file.readlines() |

**Output:**

['hello world in the new file\n', 'and another line\n']

Looping over a file object

For reading lines from a file, you can loop over the file object.

This is memory efficient, fast, and leads to simple code.

|  |  |
| --- | --- |
| 1  2  3  4 | file = open('newfile.txt', 'r')    for line in file:      print line, |

Output:

hello world in the new file

and another line

file.write( )

The write method takes one parameter, which is the string to be written.

To start a new line after writing the data, add a \n character to the end.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | file = open("newfile.txt", "w")    file.write("This is a test\n")    file.write("And here is another line\n")    file.close() |

Close ( )

When you’re done with a file, call f.close() to close it and free up any system

resources taken up by the open file.

After calling f.close(), attempts to use the file object will automatically fail.

File Handling Usages

Let's show some example on how to use the different file methods

To open a text file, use:

|  |  |
| --- | --- |
| 1 | fh = open("hello.txt", "r") |

To read a text file, use:

|  |  |
| --- | --- |
| 1  2 | fh = open("hello.txt","r")  print fh.read() |

To read one line at a time, use:

|  |  |
| --- | --- |
| 1  2 | fh = open("hello".txt", "r")  print fh.readline() |

To read a list of lines use:

|  |  |
| --- | --- |
| 1  2 | fh = open("hello.txt.", "r")  print fh.readlines() |

To write to a file, use:

|  |  |
| --- | --- |
| 1  2  3 | fh = open("hello.txt","w")  fh.write("Hello World")  fh.close() |

To write to a file, use:

|  |  |
| --- | --- |
| 1  2  3  4 | fh = open("hello.txt", "w")  lines\_of\_text = ["a line of text", "another line of text", "a third line"]  fh.writelines(lines\_of\_text)  fh.close() |

To append to file, use:

|  |  |
| --- | --- |
| 1  2  3 | fh = open("Hello.txt", "a")  fh.write("Hello World again")  fh.close |

To close a file, use

|  |  |
| --- | --- |
| 1  2  3 | fh = open("hello.txt", "r")  print fh.read()  fh.close() |

With Statement

Another way of working with file objects is the With statement.

It is good practice to use this statement.

With the "With" statement, you get better syntax and exceptions handling.

In addition, it will automatically close the file. The with statement provides a

way for ensuring that a clean-up is always used.

Opening a file using with is as simple as:

|  |  |
| --- | --- |
| 1 | with open(filename) as file: |

Let's take a look at some examples

|  |  |
| --- | --- |
| 1  2  3 | with open("newtext.txt") as file:   # Use file to refer to the file object      data = file.read()      do something with data |

You can of course also loop over the file object:

|  |  |
| --- | --- |
| 1  2  3 | with open("newfile.txt") as f:      for line in f:          print line, |

Notice, that we didn't have to write "file.close()". That will automatically be

called.

With Examples

Let's show some examples on how we can use this in our every day programming.

Write to a file using With

Write to a file using the With statement

|  |  |
| --- | --- |
| 1  2 | with open("hello.txt", "w") as f:      f.write("Hello World") |

Read a file line by line into an list

This will read the file hello.txt and save the content into "data".

|  |  |
| --- | --- |
| 1  2 | with open(hello.txt) as f:      data = f.readlines() |

Splitting Lines

As a last example, we will show how to split lines from a text file.

The split function in our example, splits the string contained in the variable

data, whenever it sees a space character.

You can split by whatever you wish, line.split(":") would split the line using

colons.

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | with open('data.txt', 'r') as f:      data = f.readlines()        for line in data:          words = line.split()          print words |

**Output:**

Because multiple values are returned by split, they are returned as an array.

['hello', 'world,', 'how', 'are', 'you', 'today?']

['today', 'is', 'saturday']

More Reading

<http://docs.python.org/2/tutorial/inputoutput.html#reading-and-writing-files>

<http://www.pythonforbeginners.com/cheatsheet/python-file-handling/>

[http://en.wikibooks.org/wiki/Non-Programmer's\_Tutorial\_for\_Python\_3/](http://en.wikibooks.org/wiki/Non-Programmer)

<http://chryswoods.com/beginning_python/>