[Using the CSV module in Python](http://www.pythonforbeginners.com/systems-programming/using-the-csv-module-in-python/)***3***

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

Tags: [CSV](http://www.pythonforbeginners.com/tag/csv/) · [Modules](http://www.pythonforbeginners.com/tag/modules/)

What is CSV?

CSV (Comma Separated Values) format is the most common import and export format

for spreadsheets and databases.

A csv file contains a number of rows, each containing a number of columns,

usually separated by commas.

CSV Module

To read (cvs) files you can simply loop over the lines and use split method to

get individual columns.

The csv module's purpose is to make it easier to deal with csv formatted file,

especially when working with data exported from spreadsheets and databases into

text files.

There is no well-defined standard, so the CSV module uses "dialects" to support

parsing using different parameters. Along with a generic reader and writer,

the module includes a dialect for working with Microsoft Excel.

CSV Functions

The CSV module contains the following functions:

**csv.reader**

**csv.writer**

**csv.register\_dialect**

**csv.unregister\_dialect**

**csv.get\_dialect**

**csv.list\_dialects**

**csv.field\_size\_limit**

In this article we will only be focusing on the **reader** and **writer** functions.

Reading CSV Files

To read data from a csv file, use the reader function to create a reader object.

The reader function will take each line of the file and make a list containing

all that line's columns.

Then we can just pick the columns we are interested in.

Example 1 – Reading CSV files

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | #To be able to read csv formated files, we will first have to import the  #csv module.  import csv  with open('some.csv', 'rb') as f:      reader = csv.reader(f)      for row in reader:          print row |

Example 2 – Reading CSV files

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | import csv     # imports the csv module  import sys      # imports the sys module    f = open(sys.argv[1], 'rb') # opens the csv file  try:      reader = csv.reader(f)  # creates the reader object      for row in reader:   # iterates the rows of the file in orders          print row    # prints each row  finally:      f.close()      # closing |

CSV Example file

As I wrote above, a csv file contains a number of rows, each containing a number

of columns, usually separated by commas, such this:

*Title,Release Date,Director*

*And Now For Something Completely Different,1971,Ian MacNaughton*

*Monty Python And The Holy Grail,1975,Terry Gilliam and Terry Jones*

*Monty Python's Life Of Brian,1979,Terry Jones*

*Monty Python Live At The Hollywood Bowl,1982,Terry Hughes*

*Monty Python's The Meaning Of Life,1983,Terry Jones*

The following example is copied from this [post](http://www.linuxjournal.com/content/handling-csv-files-python)

It starts with that you have a test csv file which contains 3 columns "A", "B",

and "C D":

**$ cat test.csv**

A,B,"C D"

1,2,"3 4"

5,6,7

Example 3 – - Reading CSV files

The following python program will then read it and displays its contents.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19 | import csv    ifile  = open('test.csv', "rb")  reader = csv.reader(ifile)    rownum = 0  for row in reader:      # Save header row.      if rownum == 0:          header = row      else:          colnum = 0          for col in row:              print '%-8s: %s' % (header[colnum], col)              colnum += 1        rownum += 1    ifile.close() |

The output will look like this when it runs:

**$ python csv1.py**

A : 1

B : 2

C D : 3 4

A : 5

B : 6

C D : 7

Writing CSV Files

When you have data to be imported into some other application, writing csv files

is just as easy as reading them.

Use writer() to create an object for writing, then iterate over the rows,

using writerow() to print them.

Example 1 – Writing CSV Files

The following Python program converts the test.csv file to a csv file that uses

tabs as a value separator and that has all values quoted.

The delimiter character and the quote character, as well as how/when to quote,

are specified when the writer is created.

These same options are available when creating reader objects.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | import csv    ifile  = open('test.csv', "rb")  reader = csv.reader(ifile)  ofile  = open('ttest.csv', "wb")  writer = csv.writer(ofile, delimiter='\t', quotechar='"', quoting=csv.QUOTE\_ALL)    for row in reader:      writer.writerow(row)    ifile.close()  ofile.close() |

The output should look like this:

**$ python csv2.py**

$ cat ttest.csv

"A" "B" "C D"

"1" "2" "3 4"

"5" "6" "7"

Quoting

The csv module contains a the following quoting options.

**csv.QUOTE\_ALL**

Quote everything, regardless of type.

**csv.QUOTE\_MINIMAL**

Quote fields with special characters

**csv.QUOTE\_NONNUMERIC**

Quote all fields that are not integers or floats

**csv.QUOTE\_NONE**

Do not quote anything on output

More reading and sources

<http://docs.python.org/2/library/csv.html>

<http://www.doughellmann.com/PyMOTW/csv/>

<http://effbot.org/librarybook/csv.htm>

<http://www.linuxjournal.com/content/handling-csv-files-python>

[http://programming-crash-course.codepoint.net/there\_are\_columns](http://programming-crash-course.codepoint.net/)