# Working with CSV files

CSV files are the most common format of exchanging files in Data Science. Most online repositories would rather store files for download in CSV format than an xlsx format. Although they are both files in spreadsheet format, a CSV file is a simplified and lightweight spreadsheet that stores data as texts.

As a data scientist, it is extremely vital to learn how to work with CSV files. That's the aim of this tutorial. By the end of this article, you will learn

* What CSV files are
* What a CSV file looks like
* How to read and write data to a CSV file using the csv library
* How to read and write data to a CSV file using the pandas library

Let’s get started.

## What is a CSV file?

A CSV file simply stands for comma-separated values. It is a file format for storing and extracting texts in a spreadsheet. Each cell in the spreadsheet is separated by commas, hence the name.

The texts are separated by commas which form the individual column. Conversely, each line of text is seen as a row and a new line indicates a new row. This is how a CSV file structures texts into a tabular format.

When working with CSV files, you do not need to reinvent the wheel by building systems to parse and write texts in CSV files. There are modules that do these tasks. We will use the CSV module as well as the pandas module in this tutorial. Before we jump into it, let’s understand a CSV file really looks like.

## What a CSV file looks like.

As mentioned above, texts separated in columns are converted into a tabular format. Let’s see an example.

|  |  |  |  |
| --- | --- | --- | --- |
| Login email | Identifier | First name | Last name |
| laura@example.com | 2070 | Laura | Grey |
| craig@example.com | 4081 | Craig | Johnson |
| mary@example.com | 9346 | Mary | Jenkins |
| jamie@example.com | 5079 | Jamie | Smith |

The CSV file for the table above is shown below.

Login email,Identifier,First name,Last name

laura@example.com,2070,Laura,Grey

craig@example.com,4081,Craig,Johnson

mary@example.com,9346,Mary,Jenkins

jamie@example.com,5079,Jamie,Smith

As seen, the columns are separated by commas while a new line is depicted as a new row.

## Working with CSV files.

In Python, there exists an in-built module for working with CSV files, the csv module. The module allows you to pull data, write, modify and store data to a CSV file. Let’s list the functions available with the csv module in python

* **csv.writer**: This is used to write data to a CSV file
* **csv.reader**: This function is used to read a CSV file
* **csv.register\_dialect**: This function is used to map string names to a dialect class
* **csv.unregister\_dialect**: This function is used to delete the string name connected to a dialect class
* **csv.list\_dialects**: This function is used to return the list of dialect names
* **csv.get\_dialect**: This function is used to return instances of dialect connected to name
* **csv.field\_size\_limit**: The function is used to set an upper limit on the parser

Here, we will focus our attention on the csv.writer and csv.reader function as those are the most functions used to do the most common tasks, reading and writing to a CSV file

## Reading a CSV file

This function is used to read the data in a CSV file. The csv.reader method parses the data in the CSV file and returns the rows. A for loop can then be used to parse and print all the rows.

#open the file in text mode

with open('email.csv', 'rt') as file:

#store it I n a variable

data = csv.reader(file)

#parse each row and print the result

for each\_row in data:

print(each\_row)

Output:

['Login email', 'Identifier', 'First name', 'Last name']

['laura@example.com', '2070', 'Laura', 'Grey']

['craig@example.com', '4081', 'Craig', 'Johnson']

['mary@example.com', '9346', 'Mary', 'Jenkins']

['jamie@example.com', '5079', 'Jamie', 'Smith']

Let's unpack what we did in the code above. We began by opening the file with the ‘with open’ statement so that we don't have to close it manually. Thereafter, we read the file using the reader() method. We then parse each row of the file using a for loop.

Notice that the column names were the first lists printed. This does not look too great. You can read the list in a nicer way by reading the CSV file as a dictionary. Let’s see how to do it.

## How to Read a CSV File as a Dictionary.

To read a CSV file as a dictionary, you use the DictReader() method rather than the reader() method. Let’s see an example.

#open the file in text mode

with open('email.csv', 'rt') as file:

#store it in a variable

data = csv.DictReader(file)

#parse each row and print the result

for each\_row in data:

print(each\_row)

Output:

OrderedDict([('Login email', 'laura@example.com'), ('Identifier', '2070'), ('First name', 'Laura'), ('Last name', 'Grey')])

OrderedDict([('Login email', 'craig@example.com'), ('Identifier', '4081'), ('First name', 'Craig'), ('Last name', 'Johnson')])

OrderedDict([('Login email', 'mary@example.com'), ('Identifier', '9346'), ('First name', 'Mary'), ('Last name', 'Jenkins')])

OrderedDict([('Login email', 'jamie@example.com'), ('Identifier', '5079'), ('First name', 'Jamie'), ('Last name', 'Smith')])

As seen, this time, the result is a list of tuples containing the column name and the values for each entry.

## Writing Data to a CSV file

Just as it is possible to read data in a CSV, you can convert write data to a CSV file. To do this, you use the writer() method. The writerow() method is then used to add data to a row.

To see how this is done, see the example below.

#import necessary libraries

import csv

#open the CSV file

with open('writen\_data.csv', mode='w') as file:

data = csv.writer(file)

#write data to the file, row by row

data.writerow(['Login email', 'Identifier', 'First name', 'Last name'])

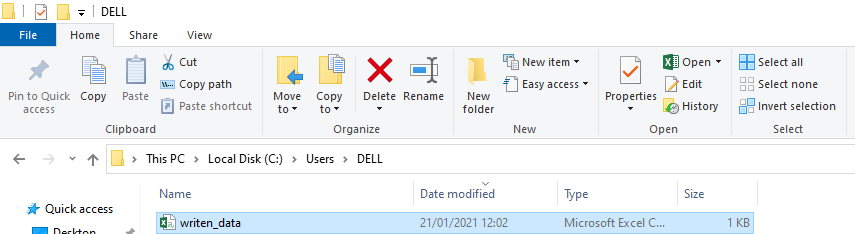
data.writerow(['laura@example.com', '2070', 'Laura', 'Grey'])

data.writerow(['craig@example.com', '4081', 'Craig', 'Johnson'])

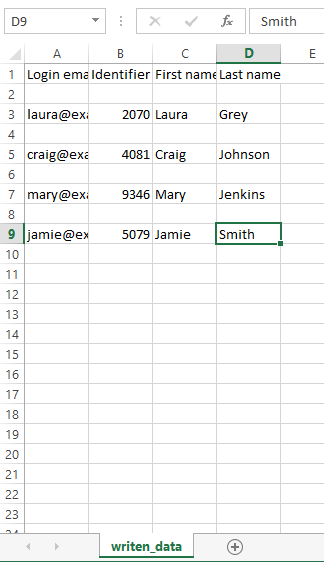
data.writerow(['mary@example.com', '9346', 'Mary', 'Jenkins'])

data.writerow(['jamie@example.com', '5079', 'Jamie', 'Smith'])

Output:



Here’s what the file looks like



## Working with CSV files using Pandas

Most often than not, you’d be using the Pandas library to work with CSV files. This is because it is easier to read CSV files on the terminal and do other operations using pandas. In this section, we will discuss how to work with CSV files using pandas.

To read CSV with pandas, we use the read\_csv() method while to write to a CSV in pandas, we use the to\_csv() method. Let’s illustrate each of these tasks using examples.

## Reading a CSV file using Pandas

As mentioned earlier, the read\_csv() method is used to read a CSV file using pandas

#import the necessary libraries

import pandas as pd

#read the CSV file

data = pd.read\_csv('email.csv')

#print the result

print(data)

Output:

Login email Identifier First name Last name

0 laura@example.com 2070 Laura Grey

1 craig@example.com 4081 Craig Johnson

2 mary@example.com 9346 Mary Jenkins

3 jamie@example.com 5079 Jamie Smith

Easy as that!

## Writing to a CSV file using Pandas

To write data into a CSV file, the to\_csv() method is used. However, it is important to point out that the data is preferably stored as a dictionary. The dictionary is converted into a dataframe. Thereafter, the dataframe is converted to a CSV file. The example below illustrates this process.

#import the necessary libraries

import pandas as pd

#create a data in dictionary form

data = {'Login email': ['laura@example.com', '2070', 'Laura', 'Grey'],

'Identifier': ['craig@example.com', '4081', 'Craig', 'Johnson'],

'First name': ['mary@example.com', '9346', 'Mary', 'Jenkins'],

'Last name' : ['jamie@example.com', '5079', 'Jamie', 'Smith'] }

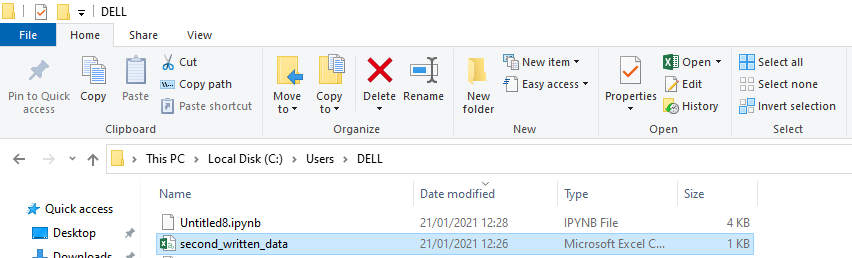
#convert the dictionary to a pandas dataframe

df = pd.DataFrame(data)

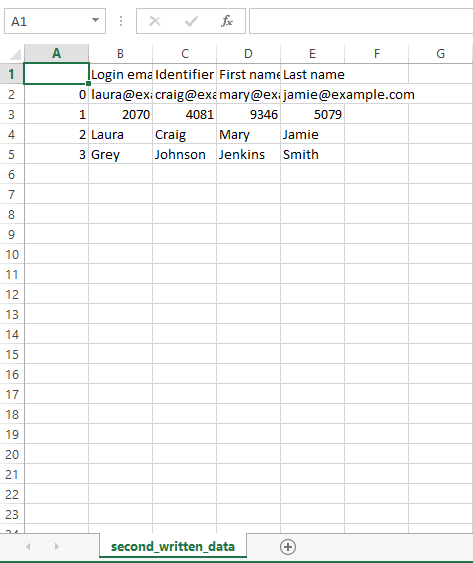
#save the dataframe as a CSV file

df.to\_csv('second\_written\_data.csv')

Output:



The CSV file looks like this:



That is all there is to it.

In this tutorial, you have learned how to write and read CSV files using the csv module. In addition, you learned how to do the same using the pandas library. It is worthy of mention to recall these methods.

1. csv.reader()
2. csv.DictReader()
3. csv.writer()
4. pandas.read\_csv()
5. pandas.to\_csv()

If you’ve got any questions, feel free to leave them in the comment section and I’d do my best to answer them.