# Writing CSV files in Python

We are going to exclusively use the `csv` module built into Python for this task. But first, we will have to import the module as :

```
import csv
```

We have already covered the basics of how to use the `csv` module to read and write into CSV files. If you don't have any idea on using the `csv` module, check out our tutorial on Python CSV: Read and Write CSV files

---

## Basic Usage of csv.writer()

Let's look at a basic example of using `csv.writer()` to refresh your existing knowledge.

### Example 1: Write into CSV files with csv.writer()

Suppose we want to write a CSV file with the following entries:

```
SN,Name,Contribution
1,Linus Torvalds,Linux Kernel
2,Tim Berners-Lee,World Wide Web
3,Guido van Rossum,Python Programming
```

Here's how we do it.

```
import csv
with open('innovators.csv', 'w', newline='') as file:
    writer = csv.writer(file)
    writer.writerow(["SN", "Name", "Contribution"])
    writer.writerow([1, "Linus Torvalds", "Linux Kernel"])
    writer.writerow([2, "Tim Berners-Lee", "World Wide Web"])
    writer.writerow([3, "Guido van Rossum", "Python Programming"])
```

When we run the above program, an **innovators.csv** file is created in the current working directory with the given entries.

Here, we have opened the **innovators.csv** file in writing mode using `open()` function.

To learn more about opening files in Python, visit: Python File Input/Output

Next, the `csv.writer()` function is used to create a `writer` object. The `writer.writerow()` function is then used to write single rows to the CSV file.

---

### Example 2: Writing Multiple Rows with writerows()

If we need to write the contents of the 2-dimensional list to a CSV file, here's how we can do it.

```
import csv
row_list = [["SN", "Name", "Contribution"],
            [1, "Linus Torvalds", "Linux Kernel"],
            [2, "Tim Berners-Lee", "World Wide Web"],
            [3, "Guido van Rossum", "Python Programming"]]
with open('protagonist.csv', 'w', newline='') as file:
    writer = csv.writer(file)
    writer.writerows(row_list)
```

The output of the program is the same as in **Example 1**.

Here, our 2-dimensional list is passed to the `writer.writerows()` function to write the content of the list to the CSV file.

---

Now let's see how we can write CSV files in different formats. We will then learn how to customize the `csv.writer()` function to write them.

---

## CSV Files with Custom Delimiters

By default, a comma is used as a delimiter in a CSV file. However, some CSV files can use delimiters other than a comma. Few popular ones are `|` and `\t`.

Suppose we want to use `|` as a delimiter in the **innovators.csv** file of **Example 1**. To write this file, we can pass an additional `delimiter` parameter to the `csv.writer()` function.

Let's take an example.

### Example 3: Write CSV File Having Pipe Delimiter

```
import csv
data_list = [["SN", "Name", "Contribution"],
             [1, "Linus Torvalds", "Linux Kernel"],
             [2, "Tim Berners-Lee", "World Wide Web"],
             [3, "Guido van Rossum", "Python Programming"]]
with open('innovators.csv', 'w', newline='') as file:
    writer = csv.writer(file, delimiter='|')
    writer.writerows(data_list)
```

**Output**

```
SN|Name|Contribution
1|Linus Torvalds|Linux Kernel
2|Tim Berners-Lee|World Wide Web
3|Guido van Rossum|Python Programming
```

As we can see, the optional parameter `delimiter = '|'` helps specify the `writer` object that the CSV file should have `|` as a delimiter.

---

## CSV files with Quotes

Some CSV files have quotes around each or some of the entries.

Let's take **quotes.csv** as an example, with the following entries:

```
"SN";"Name";"Quotes"
1;"Buddha";"What we think we become"
2;"Mark Twain";"Never regret anything that made you smile"
3;"Oscar Wilde";"Be yourself everyone else is already taken"
```

Using `csv.writer()` by default will not add these quotes to the entries.

In order to add them, we will have to use another optional parameter called `quoting`.

Let's take an example of how quoting can be used around the non-numeric values and `;` as delimiters.

### Example 4: Write CSV files with quotes

```
import csv
row_list = [
    ["SN", "Name", "Quotes"],
    [1, "Buddha", "What we think we become"],
    [2, "Mark Twain", "Never regret anything that made you smile"],
    [3, "Oscar Wilde", "Be yourself everyone else is already taken"]
]
with open('quotes.csv', 'w', newline='') as file:
    writer = csv.writer(file, quoting=csv.QUOTE_NONNUMERIC, delimiter=';')
    writer.writerows(row_list)
```

**Output**

```
"SN";"Name";"Quotes"
1;"Buddha";"What we think we become"
2;"Mark Twain";"Never regret anything that made you smile"
3;"Oscar Wilde";"Be yourself everyone else is already taken"
```

Here, the **quotes.csv** file is created in the working directory with the above entries.

As you can see, we have passed `csv.QUOTE_NONNUMERIC` to the `quoting` parameter. It is a constant defined by the `csv` module.

`csv.QUOTE_NONNUMERIC` specifies the `writer` object that quotes should be added around the non-numeric entries.

There are 3 other predefined constants you can pass to the `quoting` parameter:

- `csv.QUOTE_ALL` - Specifies the `writer` object to write CSV file with quotes around all the entries.
- `csv.QUOTE_MINIMAL` - Specifies the `writer` object to only quote those fields which contain special characters (**delimiter**, **quotechar** or any characters in **lineterminator**)
- `csv.QUOTE_NONE` - Specifies the `writer` object that none of the entries should be quoted. It is the default value.

---

## CSV files with custom quoting character

We can also write CSV files with custom quoting characters. For that, we will have to use an optional parameter called `quotechar`.

Let's take an example of writing **quotes.csv** file in **Example 4**, but with `*` as the quoting character.

### Example 5: Writing CSV files with custom quoting character

```
import csv
row_list = [
    ["SN", "Name", "Quotes"],
    [1, "Buddha", "What we think we become"],
    [2, "Mark Twain", "Never regret anything that made you smile"],
    [3, "Oscar Wilde", "Be yourself everyone else is already taken"]
]
with open('quotes.csv', 'w', newline='') as file:
    writer = csv.writer(file, quoting=csv.QUOTE_NONNUMERIC,
                        delimiter=';', quotechar='*')
    writer.writerows(row_list)
```

**Output**

```
*SN*;*Name*;*Quotes*
1;*Buddha*;*What we think we become*
2;*Mark Twain*;*Never regret anything that made you smile*
3;*Oscar Wilde*;*Be yourself everyone else is already taken*
```

Here, we can see that `quotechar='*'` parameter instructs the `writer` object to use `*` as quote for all non-numeric values.

---

## Dialects in CSV module

Notice in **Example 5** that we have passed multiple parameters (`quoting`, `delimiter` and `quotechar`) to the `csv.writer()` function.

This practice is acceptable when dealing with one or two files. But it will make the code more redundant and ugly once we start working with multiple CSV files with similar formats.

As a solution to this, the `csv` module offers `dialect` as an optional parameter.

---

Dialect helps in grouping together many specific formatting patterns like `delimiter`, `skipinitialspace`, `quoting`, `escapechar` into a single dialect name.

It can then be passed as a parameter to multiple `writer` or `reader` instances.

---

### Example 6: Write CSV file using dialect

Suppose we want to write a CSV file (**office.csv**) with the following content:

```
"ID"|"Name"|"Email"
"A878"|"Alfonso K. Hamby"|"[emailÂ protected]"
"F854"|"Susanne Briard"|"[emailÂ protected]"
"E833"|"Katja Mauer"|"[emailÂ protected]"
```

The CSV file has quotes around each entry and uses `|` as a delimiter.

Instead of passing two individual formatting patterns, let's look at how to use dialects to write this file.

```
import csv
row_list = [
    ["ID", "Name", "Email"],
    ["A878", "Alfonso K. Hamby", "[emailÂ protected]"],
    ["F854", "Susanne Briard", "[emailÂ protected]"],
    ["E833", "Katja Mauer", "[emailÂ protected]"]
]
csv.register_dialect('myDialect',
                     delimiter='|',
                     quoting=csv.QUOTE_ALL)
with open('office.csv', 'w', newline='') as file:
    writer = csv.writer(file, dialect='myDialect')
    writer.writerows(row_list)
```

**Output**

```
"ID"|"Name"|"Email"
"A878"|"Alfonso K. Hamby"|"[emailÂ protected]"
"F854"|"Susanne Briard"|"[emailÂ protected]"
"E833"|"Katja Mauer"|"[emailÂ protected]"
```

Here, **office.csv** is created in the working directory with the above contents.

From this example, we can see that the `csv.register_dialect()` function is used to define a custom dialect. Its syntax is:

```
csv.register_dialect(name[, dialect[, **fmtparams]])
```

The custom dialect requires a name in the form of a string. Other specifications can be done either by passing a sub-class of the `Dialect` class, or by individual formatting patterns as shown in the example.

---

While creating the `writer` object, we pass `dialect='myDialect'` to specify that the writer instance must use that particular dialect.

The advantage of using `dialect` is that it makes the program more modular. Notice that we can reuse **myDialect** to write other CSV files without having to re-specify the CSV format.

---

# Write CSV files with csv.DictWriter()

The objects of `csv.DictWriter()` class can be used to write to a CSV file from a Python dictionary.

The minimal syntax of the `csv.DictWriter()` class is:

```
csv.DictWriter(file, fieldnames)
```

Here,

- `file` - CSV file where we want to write to
- `fieldnames` - a `list` object which should contain the column headers specifying the order in which data should be written in the CSV file

## Example 7: Python csv.DictWriter()

```
import csv

with open('players.csv', 'w', newline='') as file:
    fieldnames = ['player_name', 'fide_rating']
    writer = csv.DictWriter(file, fieldnames=fieldnames)

    writer.writeheader()
    writer.writerow({'player_name': 'Magnus Carlsen', 'fide_rating': 2870})
    writer.writerow({'player_name': 'Fabiano Caruana', 'fide_rating': 2822})
    writer.writerow({'player_name': 'Ding Liren', 'fide_rating': 2801})
```

**Output**

The program creates a **players.csv** file with the following entries:

```
player_name,fide_rating
Magnus Carlsen,2870
Fabiano Caruana,2822
Ding Liren,2801
```

---

The full syntax of the `csv.DictWriter()` class is:

```
csv.DictWriter(f, fieldnames, restval='', extrasaction='raise', dialect='excel', *args, **kwds)
```

To learn more about it in detail, visit: Python csv.DictWriter() class

---

# CSV files with lineterminator

A `lineterminator` is a string used to terminate lines produced by `writer` objects. The default value is `\r\n`. You can change its value by passing any string as a `lineterminator` parameter.

However, the `reader` object only recognizes `\n` or `\r` as `lineterminator` values. So using other characters as line terminators is highly discouraged.

---

# doublequote & escapechar in CSV module

In order to separate delimiter characters in the entries, the `csv` module by default quotes the entries using quotation marks.

So, if you had an entry: `He is a strong, healthy man`, it will be written as: `"He is a strong, healthy man"`.

Similarly, the `csv` module uses double quotes in order to escape the quote character present in the entries by default.

If you had an entry: `Go to "programiz.com"`, it would be written as: `"Go to ""programiz.com"""`.

Here, we can see that each `"` is followed by a `"` to escape the previous one.

---

## doublequote

It handles how `quotechar` present in the entry themselves are quoted. When `True`, the quoting character is doubled and when `False`, the `escapechar` is used as a prefix to the `quotechar`. By default its value is `True`.

## escapechar

`escapechar` parameter is a string to escape the delimiter if quoting is set to `csv.QUOTE_NONE` and quotechar if doublequote is `False`. Its default value is None.

## Example 8: Using escapechar in csv writer

```
import csv
row_list = [
    ['Book', 'Quote'],
    ['Lord of the Rings',
        '"All we have to decide is what to do with the time that is given us."'],
    ['Harry Potter', '"It matters not what someone is born, but what they grow to be."']
]
with open('book.csv', 'w', newline='') as file:
    writer = csv.writer(file, escapechar='/', quoting=csv.QUOTE_NONE)
    writer.writerows(row_list)
```

### Output

```
Book,Quote
Lord of the Rings,/"All we have to decide is what to do with the time that is given us./"
Harry Potter,/"It matters not what someone is born/, but what they grow to be./"
```

Here, we can see that `/` is prefix to all the `"` and `,` because we specified `quoting=csv.QUOTE_NONE`.

If it wasn't defined, then, the output would be:

```
Book,Quote
Lord of the Rings,"""All we have to decide is what to do with the time that is given us."""
Harry Potter,"""It matters not what someone is born, but what they grow to be."""
```

Since we allow quoting, the entries with special characters(`"` in this case) are double-quoted. The entries with `delimiter` are also enclosed within quote characters.(Starting and closing quote characters)

The remaining quote characters are to escape the actual `"` present as part of the string, so that they are not interpreted as quotechar.

---

**Note:** The csv module can also be used for other file extensions (like: **.txt**) as long as their contents are in proper structure.

**Recommended Reading:** Read CSV Files in Python

## Table of Contents