

# :mod:`csv` --- CSV File Reading and Writing

**System Message: ERROR/3** (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]csv.rst, line 1); [backlink](#)

Unknown interpreted text role "mod".

**System Message: ERROR/3** (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]csv.rst, line 4)

Unknown directive type "module".

```
.. module:: csv
   :synopsis: Write and read tabular data to and from delimited files.
```

**System Message: ERROR/3** (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]csv.rst, line 7)

Unknown directive type "sectionauthor".

```
.. sectionauthor:: Skip Montanaro <skip@pobox.com>
```

Source code: :source:`Lib/csv.py`

**System Message: ERROR/3** (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]csv.rst, line 9); [backlink](#)

Unknown interpreted text role "source".

**System Message: ERROR/3** (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]csv.rst, line 11)

Unknown directive type "index".

```
.. index::
   single: csv
   pair: data; tabular
```

The so-called CSV (Comma Separated Values) format is the most common import and export format for spreadsheets and databases. CSV format was used for many years prior to attempts to describe the format in a standardized way in [RFC 4180](#). The lack of a well-defined standard means that subtle differences often exist in the data produced and consumed by different applications. These differences can make it annoying to process CSV files from multiple sources. Still, while the delimiters and quoting characters vary, the overall format is similar enough that it is possible to write a single module which can efficiently manipulate such data, hiding the details of reading and writing the data from the programmer.

The `:mod:`csv`` module implements classes to read and write tabular data in CSV format. It allows programmers to say, "write this data in the format preferred by Excel," or "read data from this file which was generated by Excel," without knowing the precise details of the CSV format used by Excel. Programmers can also describe the CSV formats understood by other applications or define their own special-purpose CSV formats.

**System Message: ERROR/3** (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]csv.rst, line 28); [backlink](#)

Unknown interpreted text role "mod".

The `:mod:`csv`` module's `:class:`reader`` and `:class:`writer`` objects read and write sequences. Programmers can also read and write data in dictionary form using the `:class:`DictReader`` and `:class:`DictWriter`` classes.

**System Message: ERROR/3** (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]csv.rst, line 35); [backlink](#)

Unknown interpreted text role "mod".

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]csv.rst, line 35); [backlink](#)**

Unknown interpreted text role "class".

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]csv.rst, line 35); [backlink](#)**

Unknown interpreted text role "class".

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]csv.rst, line 35); [backlink](#)**

Unknown interpreted text role "class".

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]csv.rst, line 35); [backlink](#)**

Unknown interpreted text role "class".

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]csv.rst, line 39)**

Unknown directive type "seealso".

```
.. seealso::
```

```
:pep:`305` - CSV File API
    The Python Enhancement Proposal which proposed this addition to Python.
```

## Module Contents

The `mod:csv` module defines the following functions:

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]csv.rst, line 50); [backlink](#)**

Unknown interpreted text role "mod".

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]csv.rst, line 53)**

Unknown directive type "index".

```
.. index::
    single: universal newlines; csv.reader function
```

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]csv.rst, line 56)**

Unknown directive type "function".

```
.. function:: reader(csvfile, dialect='excel', **fmtparams)
```

Return a reader object which will iterate over lines in the given `*csvfile*`. `*csvfile*` can be any object which supports the `:term:`iterator`` protocol and returns a string each time its `:meth:`!__next__`` method is called --- `:term:`file objects`` `<file object>` and list objects are both suitable. If `*csvfile*` is a file object, it should be opened with ```newline=''```. [1]\_ An optional `*dialect*` parameter can be given which is used to define a set of parameters specific to a particular CSV dialect. It may be an instance of a subclass of the `:class:`Dialect`` class or one of the strings returned by the `:func:`list_dialects`` function. The other optional `*fmtparams*` keyword arguments can be given to override individual formatting parameters in the current dialect. For full details about the dialect and formatting parameters, see section `:ref:`csv-fmt-params``.

Each row read from the csv file is returned as a list of strings. No

automatic data type conversion is performed unless the ``QUOTE\_NONNUMERIC`` format option is specified (in which case unquoted fields are transformed into floats).

A short usage example::

```
>>> import csv
>>> with open('eggs.csv', newline='') as csvfile:
...     spamreader = csv.reader(csvfile, delimiter=' ', quotechar='|')
...     for row in spamreader:
...         print(', '.join(row))
Spam, Spam, Spam, Spam, Spam, Baked Beans
Spam, Lovely Spam, Wonderful Spam
```

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]csv.rst, line 86)**

Unknown directive type "function".

```
.. function:: writer(csvfile, dialect='excel', **fmtparams)
```

Return a writer object responsible for converting the user's data into delimited strings on the given file-like object. \*csvfile\* can be any object with a :func:`write` method. If \*csvfile\* is a file object, it should be opened with ``newline=''`` [1]\_. An optional \*dialect\* parameter can be given which is used to define a set of parameters specific to a particular CSV dialect. It may be an instance of a subclass of the :class:`Dialect` class or one of the strings returned by the :func:`list\_dialects` function. The other optional \*fmtparams\* keyword arguments can be given to override individual formatting parameters in the current dialect. For full details about dialects and formatting parameters, see the :ref:`csv-fmt-params` section. To make it as easy as possible to interface with modules which implement the DB API, the value :const:`None` is written as the empty string. While this isn't a reversible transformation, it makes it easier to dump SQL NULL data values to CSV files without preprocessing the data returned from a ``cursor.fetch\*`` call. All other non-string data are stringified with :func:`str` before being written.

A short usage example::

```
import csv
with open('eggs.csv', 'w', newline='') as csvfile:
    spamwriter = csv.writer(csvfile, delimiter=' ',
                           quotechar='|', quoting=csv.QUOTE_MINIMAL)
    spamwriter.writerow(['Spam'] * 5 + ['Baked Beans'])
    spamwriter.writerow(['Spam', 'Lovely Spam', 'Wonderful Spam'])
```

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]csv.rst, line 115)**

Unknown directive type "function".

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

Associate \*dialect\* with \*name\*. \*name\* must be a string. The dialect can be specified either by passing a sub-class of :class:`Dialect`, or by \*fmtparams\* keyword arguments, or both, with keyword arguments overriding parameters of the dialect. For full details about dialects and formatting parameters, see section :ref:`csv-fmt-params`.

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]csv.rst, line 124)**

Unknown directive type "function".

```
.. function:: unregister_dialect(name)
```

Delete the dialect associated with \*name\* from the dialect registry. An :exc:`Error` is raised if \*name\* is not a registered dialect name.

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]csv.rst, line 130)**

Unknown directive type "function".

```
.. function:: get_dialect(name)
```

Return the dialect associated with *\*name\**. An `:exc:`Error`` is raised if *\*name\** is not a registered dialect name. This function returns an immutable `:class:`Dialect``.

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]csv.rst, line 136)**

Unknown directive type "function".

```
.. function:: list_dialects()
```

Return the names of all registered dialects.

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]csv.rst, line 141)**

Unknown directive type "function".

```
.. function:: field_size_limit([new_limit])
```

Returns the current maximum field size allowed by the parser. If *\*new\_limit\** is given, this becomes the new limit.

The `:mod:`csv`` module defines the following classes:

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]csv.rst, line 147); [backlink](#)**

Unknown interpreted text role "mod".

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]csv.rst, line 149)**

Invalid class attribute value for "class" directive: "DictReader(f, fieldnames=None, restkey=None, restval=None, \ndialect='excel', \*args, \*\*kwargs)".

```
.. class:: DictReader(f, fieldnames=None, restkey=None, restval=None, \
                    dialect='excel', *args, **kwargs)
```

Create an object that operates like a regular reader but maps the information in each row to a `:class:`dict`` whose keys are given by the optional *\*fieldnames\** parameter.

The *\*fieldnames\** parameter is a `:term:`sequence``. If *\*fieldnames\** is omitted, the values in the first row of file *\*f\** will be used as the fieldnames. Regardless of how the fieldnames are determined, the dictionary preserves their original ordering.

If a row has more fields than fieldnames, the remaining data is put in a list and stored with the fieldname specified by *\*restkey\** (which defaults to ``None``). If a non-blank row has fewer fields than fieldnames, the missing values are filled-in with the value of *\*restval\** (which defaults to ``None``).

All other optional or keyword arguments are passed to the underlying `:class:`reader`` instance.

```
.. versionchanged:: 3.6
   Returned rows are now of type :class:`OrderedDict`.
```

```
.. versionchanged:: 3.8
   Returned rows are now of type :class:`dict`.
```

A short usage example::

```
>>> import csv
>>> with open('names.csv', newline='') as csvfile:
...     reader = csv.DictReader(csvfile)
...     for row in reader:
...         print(row['first_name'], row['last_name'])
...
Eric Idle
John Cleese

>>> print(row)
{'first_name': 'John', 'last_name': 'Cleese'}
```

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]csv.rst, line 191)**

Invalid class attribute value for "class" directive: "DictWriter(f, fieldnames, restval=", extrasaction='raise', \ndialect='excel', \*args, \*\*kwds)".

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

Create an object which operates like a regular writer but maps dictionaries onto output rows. The `*fieldnames*` parameter is a `:mod:`sequence <collections.abc>`` of keys that identify the order in which values in the dictionary passed to the `:meth:`writerow`` method are written to file `*f*`. The optional `*restval*` parameter specifies the value to be written if the dictionary is missing a key in `*fieldnames*`. If the dictionary passed to the `:meth:`writerow`` method contains a key not found in `*fieldnames*`, the optional `*extrasaction*` parameter indicates what action to take.

If it is set to ```'raise'```, the default value, a `:exc:`ValueError`` is raised.

If it is set to ```'ignore'```, extra values in the dictionary are ignored. Any other optional or keyword arguments are passed to the underlying `:class:`writer`` instance.

Note that unlike the `:class:`DictReader`` class, the `*fieldnames*` parameter of the `:class:`DictWriter`` class is not optional.

A short usage example::

```
import csv

with open('names.csv', 'w', newline='') as csvfile:
    fieldnames = ['first_name', 'last_name']
    writer = csv.DictWriter(csvfile, fieldnames=fieldnames)

    writer.writeheader()
    writer.writerow({'first_name': 'Baked', 'last_name': 'Beans'})
    writer.writerow({'first_name': 'Lovely', 'last_name': 'Spam'})
    writer.writerow({'first_name': 'Wonderful', 'last_name': 'Spam'})
```

The `:class:`Dialect`` class is a container class whose attributes contain information for how to handle doublequotes, whitespace, delimiters, etc. Due to the lack of a strict CSV specification, different applications produce subtly different CSV data. `:class:`Dialect`` instances define how `:class:`reader`` and `:class:`writer`` instances behave.

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]csv.rst, line 228); [backlink](#)**

Unknown interpreted text role "class".

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]csv.rst, line 228); [backlink](#)**

Unknown interpreted text role "class".

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]csv.rst, line 228); [backlink](#)**

Unknown interpreted text role "class".

**System Message: ERROR/3** (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library] csv.rst, line 228); [backlink](#)

Unknown interpreted text role "class".

All available `:class:`Dialect`` names are returned by `:func:`list_dialects``, and they can be registered with specific `:class:`reader`` and `:class:`writer`` classes through their initializer (`__init__`) functions like this:

**System Message: ERROR/3** (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library] csv.rst, line 234); [backlink](#)

Unknown interpreted text role "class".

**System Message: ERROR/3** (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library] csv.rst, line 234); [backlink](#)

Unknown interpreted text role "func".

**System Message: ERROR/3** (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library] csv.rst, line 234); [backlink](#)

Unknown interpreted text role "class".

**System Message: ERROR/3** (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library] csv.rst, line 234); [backlink](#)

Unknown interpreted text role "class".

```
import csv

with open('students.csv', 'w', newline='') as csvfile:
    writer = csv.writer(csvfile, dialect='unix')
    ^^^^^^^^^^^^^^^^^
```

The `:class:`excel`` class defines the usual properties of an Excel-generated CSV file. It is registered with the dialect name `'excel'`.

**System Message: ERROR/3** (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library] csv.rst, line 247); [backlink](#)

Unknown interpreted text role "class".

The `:class:`excel_tab`` class defines the usual properties of an Excel-generated TAB-delimited file. It is registered with the dialect name `'excel-tab'`.

**System Message: ERROR/3** (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library] csv.rst, line 253); [backlink](#)

Unknown interpreted text role "class".

The `:class:`unix_dialect`` class defines the usual properties of a CSV file generated on UNIX systems, i.e. using `'\n'` as line terminator and quoting all fields. It is registered with the dialect name `'unix'`.

**System Message: ERROR/3** (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library] csv.rst, line 259); [backlink](#)

Unknown interpreted text role "class".

**System Message: ERROR/3** (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library] csv.rst, line 263)

Unknown directive type "versionadded".

```
.. versionadded:: 3.2
```

The `:class:`Sniffer`` class is used to deduce the format of a CSV file.

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]csv.rst, line 268); [backlink](#)**

Unknown interpreted text role "class".

The `:class:`Sniffer`` class provides two methods:

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]csv.rst, line 270); [backlink](#)**

Unknown interpreted text role "class".

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]csv.rst, line 272)**

Unknown directive type "method".

```
.. method:: sniff(sample, delimiters=None)
```

Analyze the given *\*sample\** and return a `:class:`Dialect`` subclass reflecting the parameters found. If the optional *\*delimiters\** parameter is given, it is interpreted as a string containing possible valid delimiter characters.

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]csv.rst, line 280)**

Unknown directive type "method".

```
.. method:: has_header(sample)
```

Analyze the sample text (presumed to be in CSV format) and return `:const:`True`` if the first row appears to be a series of column headers. Inspecting each column, one of two key criteria will be considered to estimate if the sample contains a header:

- the second through n-th rows contain numeric values
- the second through n-th rows contain strings where at least one value's length differs from that of the putative header of that column.

Twenty rows after the first row are sampled; if more than half of columns + rows meet the criteria, `:const:`True`` is returned.

#### Note

This method is a rough heuristic and may produce both false positives and negatives.

An example for `:class:`Sniffer`` use:

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]csv.rst, line 299); [backlink](#)**

Unknown interpreted text role "class".

```
with open('example.csv', newline='') as csvfile:
    dialect = csv.Sniffer().sniff(csvfile.read(1024))
    csvfile.seek(0)
    reader = csv.reader(csvfile, dialect)
    # ... process CSV file contents here ...
```

The `mod:`csv`` module defines the following constants:

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]csv.rst, line 308); [backlink](#)**

Unknown interpreted text role "mod".

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-**

main\Doc\library\[cpython-main] [Doc] [library]csv.rst, line 310)

Unknown directive type "data".

```
.. data:: QUOTE_ALL
```

Instructs :class:`writer` objects to quote all fields.

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]csv.rst, line 315)**

Unknown directive type "data".

```
.. data:: QUOTE_MINIMAL
```

Instructs :class:`writer` objects to only quote those fields which contain special characters such as *\*delimiter\**, *\*quotechar\** or any of the characters in *\*lineterminator\**.

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]csv.rst, line 322)**

Unknown directive type "data".

```
.. data:: QUOTE_NONNUMERIC
```

Instructs :class:`writer` objects to quote all non-numeric fields.

Instructs the reader to convert all non-quoted fields to type *\*float\**.

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]csv.rst, line 329)**

Unknown directive type "data".

```
.. data:: QUOTE_NONE
```

Instructs :class:`writer` objects to never quote fields. When the current *\*delimiter\** occurs in output data it is preceded by the current *\*escapechar\** character. If *\*escapechar\** is not set, the writer will raise :exc:`Error` if any characters that require escaping are encountered.

Instructs :class:`reader` to perform no special processing of quote characters.

The `mod:`csv`` module defines the following exception:

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]csv.rst, line 338); [backlink](#)**

Unknown interpreted text role "mod".

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]csv.rst, line 341)**

Unknown directive type "exception".

```
.. exception:: Error
```

Raised by any of the functions when an error is detected.

## Dialects and Formatting Parameters

To make it easier to specify the format of input and output records, specific formatting parameters are grouped together into dialects. A dialect is a subclass of the `:class:`Dialect`` class having a set of specific methods and a single `:meth:`validate`` method. When creating `:class:`reader`` or `:class:`writer`` objects, the programmer can specify a string or a subclass of the `:class:`Dialect`` class as the dialect parameter. In addition to, or instead of, the *dialect* parameter, the programmer can also specify individual formatting



parameters, which have the same names as the attributes defined below for the `:class:'Dialect'` class.

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]csv.rst, line 350); [backlink](#)**

Unknown interpreted text role "class".

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]csv.rst, line 350); [backlink](#)**

Unknown interpreted text role "meth".

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]csv.rst, line 350); [backlink](#)**

Unknown interpreted text role "class".

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]csv.rst, line 350); [backlink](#)**

Unknown interpreted text role "class".

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]csv.rst, line 350); [backlink](#)**

Unknown interpreted text role "class".

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]csv.rst, line 350); [backlink](#)**

Unknown interpreted text role "class".

Dialects support the following attributes:

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]csv.rst, line 363)**

Unknown directive type "attribute".

```
.. attribute:: Dialect.delimiter
```

A one-character string used to separate fields. It defaults to ``,``.

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]csv.rst, line 368)**

Unknown directive type "attribute".

```
.. attribute:: Dialect.doublequote
```

Controls how instances of `*quotechar*` appearing inside a field should themselves be quoted. When `:const:`True``, the character is doubled. When `:const:`False``, the `*escapechar*` is used as a prefix to the `*quotechar*`. It defaults to `:const:`True``.

On output, if `*doublequote*` is `:const:`False`` and no `*escapechar*` is set, `:exc:`Error`` is raised if a `*quotechar*` is found in a field.

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]csv.rst, line 379)**

Unknown directive type "attribute".

```
.. attribute:: Dialect.escapechar
```

A one-character string used by the writer to escape the `*delimiter*` if `*quoting*`

is set to :const:`QUOTE\_NONE` and the \*quotechar\* if \*doublequote\* is :const:`False`. On reading, the \*escapechar\* removes any special meaning from the following character. It defaults to :const:`None`, which disables escaping.

.. versionchanged:: 3.11  
An empty \*escapechar\* is not allowed.

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]csv.rst, line 389)**

Unknown directive type "attribute".

.. attribute:: Dialect.lineterminator

The string used to terminate lines produced by the :class:`writer`. It defaults to ``'\r\n'``.

.. note::

The :class:`reader` is hard-coded to recognise either ``'\r'`` or ``'\n'`` as end-of-line, and ignores \*lineterminator\*. This behavior may change in the future.

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]csv.rst, line 401)**

Unknown directive type "attribute".

.. attribute:: Dialect.quotechar

A one-character string used to quote fields containing special characters, such as the \*delimiter\* or \*quotechar\*, or which contain new-line characters. It defaults to `'"'`.

.. versionchanged:: 3.11  
An empty \*quotechar\* is not allowed.

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]csv.rst, line 410)**

Unknown directive type "attribute".

.. attribute:: Dialect.quoting

Controls when quotes should be generated by the writer and recognised by the reader. It can take on any of the :const:`QUOTE\_\*` constants (see section :ref:`csv-contents`) and defaults to :const:`QUOTE\_MINIMAL`.

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]csv.rst, line 417)**

Unknown directive type "attribute".

.. attribute:: Dialect.skipinitialspace

When :const:`True`, whitespace immediately following the \*delimiter\* is ignored. The default is :const:`False`.

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]csv.rst, line 423)**

Unknown directive type "attribute".

.. attribute:: Dialect.strict

When ``True``, raise exception :exc:`Error` on bad CSV input. The default is ``False``.

## Reader Objects

Reader objects (:class:`DictReader` instances and objects returned by the :func:`reader` function) have the following public methods:

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library] csv.rst, line 431); [backlink](#)**

Unknown interpreted text role "class".

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library] csv.rst, line 431); [backlink](#)**

Unknown interpreted text role "func".

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library] csv.rst, line 434)**

Unknown directive type "method".

```
.. method:: csvreader.__next__()
```

Return the next row of the reader's iterable object as a list (if the object was returned from :func:`reader`) or a dict (if it is a :class:`DictReader` instance), parsed according to the current :class:`Dialect`. Usually you should call this as ``next(reader)``.

Reader objects have the following public attributes:

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library] csv.rst, line 444)**

Unknown directive type "attribute".

```
.. attribute:: csvreader.dialect
```

A read-only description of the dialect in use by the parser.

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library] csv.rst, line 449)**

Unknown directive type "attribute".

```
.. attribute:: csvreader.line_num
```

The number of lines read from the source iterator. This is not the same as the number of records returned, as records can span multiple lines.

DictReader objects have the following public attribute:

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library] csv.rst, line 457)**

Unknown directive type "attribute".

```
.. attribute:: csvreader.fieldnames
```

If not passed as a parameter when creating the object, this attribute is initialized upon first access or when the first record is read from the file.

## Writer Objects

:class:`Writer` objects (:class:`DictWriter` instances and objects returned by the :func:`writer` function) have the following public

methods. A *row* must be an iterable of strings or numbers for `:class: 'Writer'` objects and a dictionary mapping fieldnames to strings or numbers (by passing them through `:func: 'str'` first) for `:class: 'DictWriter'` objects. Note that complex numbers are written out surrounded by parens. This may cause some problems for other programs which read CSV files (assuming they support complex numbers at all).

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]csv.rst, line 468); [backlink](#)**

Unknown interpreted text role "class".

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]csv.rst, line 468); [backlink](#)**

Unknown interpreted text role "class".

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]csv.rst, line 468); [backlink](#)**

Unknown interpreted text role "func".

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]csv.rst, line 468); [backlink](#)**

Unknown interpreted text role "class".

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]csv.rst, line 468); [backlink](#)**

Unknown interpreted text role "func".

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]csv.rst, line 468); [backlink](#)**

Unknown interpreted text role "class".

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]csv.rst, line 477)**

Unknown directive type "method".

```
.. method:: csvwriter.writerow(row)
```

Write the *\*row\** parameter to the writer's file object, formatted according to the current `:class: 'Dialect'`. Return the return value of the call to the *\*write\** method of the underlying file object.

```
.. versionchanged:: 3.5
   Added support of arbitrary iterables.
```

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]csv.rst, line 486)**

Unknown directive type "method".

```
.. method:: csvwriter.writerows(rows)
```

Write all elements in *\*rows\** (an iterable of *\*row\** objects as described above) to the writer's file object, formatted according to the current dialect.

Writer objects have the following public attribute:

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]csv.rst, line 495)**

Unknown directive type "attribute".

```
.. attribute:: csvwriter.dialect
```

A read-only description of the dialect in use by the writer.

DictWriter objects have the following public method:

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]csv.rst, line 503)**

Unknown directive type "method".

```
.. method:: DictWriter.writeheader()
```

Write a row with the field names (as specified in the constructor) to the writer's file object, formatted according to the current dialect. Return the return value of the :meth:`csvwriter.writerow` call used internally.

```
.. versionadded:: 3.2
```

```
.. versionchanged:: 3.8
```

:meth:`writeheader` now also returns the value returned by the :meth:`csvwriter.writerow` method it uses internally.

## Examples

The simplest example of reading a CSV file:

```
import csv
with open('some.csv', newline='') as f:
    reader = csv.reader(f)
    for row in reader:
        print(row)
```

Reading a file with an alternate format:

```
import csv
with open('passwd', newline='') as f:
    reader = csv.reader(f, delimiter=':', quoting=csv.QUOTE_NONE)
    for row in reader:
        print(row)
```

The corresponding simplest possible writing example is:

```
import csv
with open('some.csv', 'w', newline='') as f:
    writer = csv.writer(f)
    writer.writerows(someiterable)
```

Since :func:`open` is used to open a CSV file for reading, the file will by default be decoded into unicode using the system default encoding (see :func:`locale.getpreferredencoding`). To decode a file using a different encoding, use the `encoding` argument of `open`:

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]csv.rst, line 543); [backlink](#)**

Unknown interpreted text role "func".

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]csv.rst, line 543); [backlink](#)**

Unknown interpreted text role "func".

```
import csv
with open('some.csv', newline='', encoding='utf-8') as f:
    reader = csv.reader(f)
    for row in reader:
        print(row)
```

The same applies to writing in something other than the system default encoding: specify the encoding argument when opening the output file.

Registering a new dialect:

```
import csv
```

```
csv.register_dialect('unixpwd', delimiter=':', quoting=csv.QUOTE_NONE)
with open('passwd', newline='') as f:
    reader = csv.reader(f, 'unixpwd')
```

A slightly more advanced use of the reader --- catching and reporting errors:

```
import csv, sys
filename = 'some.csv'
with open(filename, newline='') as f:
    reader = csv.reader(f)
    try:
        for row in reader:
            print(row)
    except csv.Error as e:
        sys.exit('file {}, line {}: {}'.format(filename, reader.line_num, e))
```

And while the module doesn't directly support parsing strings, it can easily be done:

```
import csv
for row in csv.reader(['one,two,three']):
    print(row)
```

## Footnotes

- [1] If `newline=''` is not specified, newlines embedded inside quoted fields will not be interpreted correctly, and on platforms that use `\r\n` lineendings on write an extra `\r` will be added. It should always be safe to specify `newline=''`, since the csv module does its own (term universal <universal newlines>) newline handling.

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library] csv.rst, line 586);**  
*backlink*

Unknown interpreted text role "term".