

[🏠](#) > [API reference](#) > [Input/output](#) > [pandas.DataF...](#)

pandas.DataFrame.to_csv

```
DataFrame.to_csv(path_or_buf=None, *, sep=',', na_rep='',  
float_format=None, columns=None, header=True, index=True,  
index_label=None, mode='w', encoding=None, compression='infer',  
quoting=None, quotechar='\"', lineterminator=None, chunksize=None,  
date_format=None, doublequote=True, escapechar=None, decimal='.',  
errors='strict', storage_options=None)
```

[\[source\]](#)

Write object to a comma-separated values (csv) file.

Parameters:

path_or_buf : *str, path object, file-like object, or None, default None*

String, path object (implementing `os.PathLike[str]`), or file-like object implementing a `write()` function. If `None`, the result is returned as a string. If a non-binary file object is passed, it should be opened with `newline=""`, disabling universal newlines. If a binary file object is passed, `mode` might need to contain a `'b'`.

sep : *str, default ','*

String of length 1. Field delimiter for the output file.

na_rep : *str, default ''*

Missing data representation.

float_format : *str, Callable, default None*

Format string for floating point numbers. If a `Callable` is given, it takes precedence over other numeric formatting parameters, like `decimal`.

columns : *sequence, optional*

Columns to write.

header : *bool or list of str, default True*

Write out the column names. If a list of strings is given it is assumed to be aliases for the column names.

index : *bool, default True*

[Skip to main content](#)

index_label : str or sequence, or False, default None

Column label for index column(s) if desired. If None is given, and *header* and *index* are True, then the index names are used. A sequence should be given if the object uses MultiIndex. If False do not print fields for index names. Use `index_label=False` for easier importing in R.

mode : {'w', 'x', 'a'}, default 'w'

Forwarded to either `open(mode=)` or `fsspec.open(mode=)` to control the file opening. Typical values include:

- 'w', truncate the file first.
- 'x', exclusive creation, failing if the file already exists.
- 'a', append to the end of file if it exists.

encoding : str, optional

A string representing the encoding to use in the output file, defaults to 'utf-8'. *encoding* is not supported if *path_or_buf* is a non-binary file object.

compression : str or dict, default 'infer'

For on-the-fly compression of the output data. If 'infer' and 'path_or_buf' is path-like, then detect compression from the following extensions: '.gz', '.bz2', '.zip', '.xz', '.zst', '.tar', '.tar.gz', '.tar.xz' or '.tar.bz2' (otherwise no compression). Set to `None` for no compression. Can also be a dict with key `'method'` set to one of `{ 'zip', 'gzip', 'bz2', 'zstd', 'xz', 'tar' }` and other key-value pairs are forwarded to `zipfile.ZipFile`, `gzip.GzipFile`, `bz2.BZ2File`, `zstandard.ZstdCompressor`, `lzma.LZMAFile` or `tarfile.TarFile`, respectively. As an example, the following could be passed for faster compression and to create a reproducible gzip archive:

```
compression={'method': 'gzip', 'compresslevel': 1, 'mtime': 1}.
```

 **New in version 1.5.0:** Added support for .tar files.

May be a dict with key 'method' as compression mode and other entries as additional compression options if compression mode is 'zip'.

Passing compression options as keys in dict is supported for compression modes 'gzip', 'bz2', 'zstd', and 'zip'.

quoting : optional constant from csv module

Defaults to `csv.QUOTE_MINIMAL`. If you have set a *float_format* then floats are converted to strings and thus `csv.QUOTE_NONNUMERIC` will treat them as non-


[Skip to main content](#)

quotechar : *str, default `"`*

String of length 1. Character used to quote fields.

lineterminator : *str, optional*

The newline character or character sequence to use in the output file. Defaults to `os.linesep`, which depends on the OS in which this method is called (`'\n'` for linux, `'\r\n'` for Windows, i.e.).

 **Changed in version 1.5.0:** Previously was `line_terminator`, changed for consistency with `read_csv` and the standard library `'csv'` module.

chunksize : *int or None*

Rows to write at a time.

date_format : *str, default None*

Format string for datetime objects.

doublequote : *bool, default True*

Control quoting of *quotechar* inside a field.

escapechar : *str, default None*

String of length 1. Character used to escape *sep* and *quotechar* when appropriate.

decimal : *str, default `'`*

Character recognized as decimal separator. E.g. use `'` for European data.

errors : *str, default `'strict'`*

Specifies how encoding and decoding errors are to be handled. See the `errors` argument for `open\(\)` for a full list of options.

storage_options : *dict, optional*

Extra options that make sense for a particular storage connection, e.g. host, port, username, password, etc. For HTTP(S) URLs the key-value pairs are forwarded to `urllib.request.Request` as header options. For other URLs (e.g. starting with `"s3://"`, and `"gcs://"`) the key-value pairs are forwarded to `fsspec.open`. Please see `fsspec` and `urllib` for more details, and for more examples on storage options refer [here](#).

Returns:

None or str

[Skip to main content](#)

If `path_or_buf` is `None`, returns the resulting csv format as a string. Otherwise returns `None`.

➡ See also

[read_csv](#)

Load a CSV file into a DataFrame.

[to_excel](#)

Write DataFrame to an Excel file.

Examples

Create 'out.csv' containing 'df' without indices

```
>>> df = pd.DataFrame({'name': ['Raphael', 'Donatello'],
...                    'mask': ['red', 'purple'],
...                    'weapon': ['sai', 'bo staff']})
>>> df.to_csv('out.csv', index=False)
```

Create 'out.zip' containing 'out.csv'

```
>>> df.to_csv(index=False)
'name,mask,weapon\nRaphael,red,sai\nDonatello,purple,bo staff\n'
>>> compression_opts = dict(method='zip',
...                           archive_name='out.csv')
>>> df.to_csv('out.zip', index=False,
...           compression=compression_opts)
```

To write a csv file to a new folder or nested folder you will first need to create it using either Pathlib or os:

```
>>> from pathlib import Path
>>> filepath = Path('folder/subfolder/out.csv')
>>> filepath.parent.mkdir(parents=True, exist_ok=True)
>>> df.to_csv(filepath)
```

```
>>> import os
>>> os.makedirs('folder/subfolder', exist_ok=True)
>>> df.to_csv('folder/subfolder/out.csv')
```

[Skip to main content](#)

© 2024, pandas via [NumFOCUS, Inc.](#) Hosted by [OVHcloud](#).

Built with the [PyData Sphinx Theme](#)

0.14.4.

Created using [Sphinx](#) 7.2.6.