csvio

Release 0.2.0

Salman Raza

Oct 03, 2021

CONTENTS

csvio is a Python library that provides a wrapper around Python's built in `csv.DictReader` and `csv.DictWriter`, for ease of reading and writing CSV files.

Rows in a CSV are represented and processed as a list of dictionaries. Each item in this list is a dictionary that represents a row. The key, value pairs in each dictionary is a mapping between the column and its associated row value from the CSV.

CHAPTER

ONE

INSTALLATION

```
pip install csvio
```

CHAPTER

TWO

DOCUMENTATION

Readthedocs

CHAPTER

THREE

READING CSVS

```
>>> from csvio import CSVReader
>>> reader = CSVReader("fruit_stock.csv")
>>> reader.fieldnames
['Supplier', 'Fruit', 'Quantity']
>>> len(reader.rows)
4

>>> import json
>>> print(json.dumps(reader.rows, indent=4))
[
   {
         "Supplier": "Big Apple",
         "Fruit": "Apple",
         "Quantity": "1"
   },
   {
         "Supplier": "Big Melons",
         "Fruit": "Melons",
         "Quantity": "2"
   },
   {
         "Supplier": "Long Mangoes",
         "Fruit": "Mango",
         "Quantity": "3"
   },
   {
         "Supplier": "Small Strawberries",
         "Fruit": "Strawberry",
         "Quantity": "4"
   }
]
```

CSV file contents:

```
Supplier,Fruit,Quantity
Big Apple,Apple,1
Big Melons,Melons,2
Long Mangoes,Mango,3
Small Strawberries,Strawberry,4
```

CHAPTER

FOUR

WRITING CSVS

```
>>> from csvio import CSVWriter
>>> writer = CSVWriter("fruit_stock.csv", fieldnames=["Supplier", "Fruit", "Quantity"])
>>> row1 = {"Supplier": "Big Apple", "Fruit": "Apple", "Quantity": 1}
>>> writer.add_rows(row1)
>>> rows2_3_4 = [
...     {"Supplier": "Big Melons", "Fruit": "Melons", "Quantity": 2},
...     {"Supplier": "Long Mangoes", "Fruit": "Mango", "Quantity": 3},
...     {"Supplier": "Small Strawberries", "Fruit": "Strawberry", "Quantity": 4}
... ]
>>> writer.add_rows(rows2_3_4)
>>> len(writer.pending_rows)
4

>>> len(writer.rows)
0

>>> writer.flush()
>>> len(writer.pending_rows)
0

>>> len(writer.rows)
4
```

Once flush is called a CSV file with the name *fruit_stock.csv* will be written with the following contents.

```
Supplier,Fruit,Quantity
Big Apple,Apple,1
Big Melons,Melons,2
Long Mangoes,Mango,3
Small Strawberries,Strawberry,4
```

CHAPTER

FIVE

CONTENTS

5.1 API Specifications

5.1.1 FileBase

class csvio.filebase.**FileBase**(*filename: str*)

Bases: object

This is a base class representing a basic file.

Parameters **filename** (*required*) – Full path to a file.

delete(*missing_ok: bool = False*) → bool

Delete the file at the path provided in the *filename* parameter

Parameters **missing_ok** (*optional*) – Parameter to pass to the pathlib.Path.unlink() method.

Returns

True If file is deleted successfully.

False On failure.

property file_ext: str

Returns Extension suffix of the file without parent directory and file name.

property filedir: str

Returns Parent directory path of the file (excluding the name of the file)

property filename: str

Returns File name without the parent directory path.

property filename_no_ext: str

Returns File name without parent directory and file extension.

property filepath: str

Returns Complete file path including the parent directory, file name and extension

property path_obj: pathlib.Path

Returns pathlib.Path object representing *filename*.

touch(*exist_ok: bool = False*) → bool

Create a blank file at the path provided in the *filename* parameter.

Parameters **exist_ok** (*optional*) – Parameter to pass to the `pathlib.Path.touch()` method.

Returns

`True` If blank file is created successfully.

`False` On failure.

5.1.2 CSVBase

class csvio.csvbase.**CSVBase**(*filename: str*, *open_kwargs: Dict[str, Any] = {}*, *csv_kwargs: Dict[str, Any] = {}*)

Bases: *csvio.filebase.FileBase*

This is a base class representing a basic CSV file for reading/writing.

Parameters

- **filename** (*required*) – Full path to the CSV file for reading/writing.
- **open_kwargs** (*optional*) – A dictionary of key, value pairs that should be passed to the open method within this class.
- **csv_kwargs** (*optional*) – A dictionary of key, value pairs that should be passed to the DictReader constructor within this class.

property csv_kwargs: Dict[str, Any]

Returns A dictionary of key, value pairs that should be passed to the DictReader constructor within this class.

delete(*missing_ok: bool = False*) → bool

Delete the file at the path provided in the *filename* parameter

Parameters **missing_ok** (*optional*) – Parameter to pass to the `pathlib.Path.unlink()` method.

Returns

`True` If file is deleted successfully.

`False` On failure.

property fieldnames: List[str]

Returns List of column headings

property file_ext: str

Returns Extension suffix of the file without parent directory and file name.

property filedir: str

Returns Parent directory path of the file (excluding the name of the file)

property filename: str

Returns File name without the parent directory path.

property filename_no_ext: str

Returns File name without parent directory and file extension.

property filepath: str

Returns Complete file path including the parent directory, file name and extension

property num_rows: int

Returns The total number of rows in the CSV (excluding column headings)

property open_kwargs: Dict[str, Any]

Returns A dictionary of key, value pairs that should be passed to the open method within this class.

property path_obj: pathlib.Path

Returns `pathlib.Path` object representing *filename*.

property rows: List[Dict[str, Any]]

Returns A list of dictionaries where each item in it represents a row in the CSV file. Each dictionary in the list maps the column heading (fieldname) to the corresponding value for it from the CSV.

rows_from_column_key(*column_name: str*, *rows: Optional[List[Dict[str, Any]]] = None*) → Dict[str, List[Dict[str, Any]]]

Collect all the rows in the `rows` parameter that have the same values for the column defined in the `column_name` parameter, and construct a dictionary with the `column_name` value as the key and the corresponding rows as a list of dictionaries, as the value of this key.

Parameters

- **column_name** (*required*) – Name of the column that is to be used as the key under which all the rows having the samee value of this column will be collected.
- **rows** (optional. If not provided `self.rows` will be used.) – List of dictionaries representing the rows that will be separated and collected under a the common value of the column name provided in `column_name` parameter.

Returns A dictionary constructed using the logic as explained above.

rows_to_nested_dicts(*column_order: List[str]*, *rows: Optional[List[Dict[str, Any]]] = None*) → Dict[str, Any]

Collect all values of columns that are the same and construct a nested dictionary that has the common values as the keys, in the same order of hierarchy as provided in the *column_order* parameter.

The value of the last column name in the *column_order* list

Parameters

- **column_order** (*required*) – An ordered list of column names, to be used for constructing the dictionary
- **rows** (optional. If not provided `self.rows` will be used.) – List of dictionaries representing the rows that will be transformed to the output Dictionary.

Returns A dictionary with same column values collected under a common key in a hierarchical order.

Example:

CSV Contents: *fruit_stock.csv*

```
Supplier,Fruit,Origin,Quantity
Big Apples,Apple,Spain,1
Big Melons,Melons,Italy,2
Long Mangoes,Mango,India,3
Small Strawberries,Strawberry,France,4
```

(continues on next page)

(continued from previous page)

```
Short Mangoes,Mango,France,5
Sweet Strawberries,Strawberry,Spain,6
Square Apples,Apple,Italy,7
Small Melons,Melons,Italy,8
Dark Berries,Strawberry,Australia,9
Sweet Berries,Blackcurrant,Australia,10
```

Create dictionary with hierarchy `{"Fruit": [rows]}`

```
from csvio.csvreader import CSVReader
from json import dumps

reader = CSVReader("fruit_stock.csv")

col_order = ["Fruit"]

dict_tree= reader.rows_to_nested_dicts(col_order)

print(dumps(dict_tree, indent=4))
```

Output:

```
{
    "Apple": [
        {
            "Supplier": "Big Apples",
            "Fruit": "Apple",
            "Origin": "Spain",
            "Quantity": "1"
        },
        {
            "Supplier": "Square Apples",
            "Fruit": "Apple",
            "Origin": "Italy",
            "Quantity": "7"
        }
    ],
    "Melons": [
        {
            "Supplier": "Big Melons",
            "Fruit": "Melons",
            "Origin": "Italy",
            "Quantity": "2"
        },
        {
            "Supplier": "Small Melons",
            "Fruit": "Melons",
            "Origin": "Italy",
            "Quantity": "8"
        }
    ],
    "Mango": [
```

(continues on next page)

(continued from previous page)

```
        {
            "Supplier": "Long Mangoes",
            "Fruit": "Mango",
            "Origin": "India",
            "Quantity": "3"
        },
        {
            "Supplier": "Short Mangoes",
            "Fruit": "Mango",
            "Origin": "France",
            "Quantity": "5"
        }
    ],
    "Strawberry": [
        {
            "Supplier": "Small Strawberries",
            "Fruit": "Strawberry",
            "Origin": "France",
            "Quantity": "4"
        },
        {
            "Supplier": "Sweet Strawberries",
            "Fruit": "Strawberry",
            "Origin": "Spain",
            "Quantity": "6"
        },
        {
            "Supplier": "Dark Berries",
            "Fruit": "Strawberry",
            "Origin": "Australia",
            "Quantity": "9"
        }
    ],
    "Blackcurrant": [
        {
            "Supplier": "Sweet Berries",
            "Fruit": "Blackcurrant",
            "Origin": "Australia",
            "Quantity": "10"
        }
    ]
}
```

Create dictionary with hierarchy `{"Fruit": "Origin" : [rows]}`

```
from csvio.csvreader import CSVReader
from json import dumps

reader = CSVReader("fruit_stock.csv")

col_order = ["Fruit", "Origin"]
```

(continues on next page)

(continued from previous page)

```
dict_tree= reader.rows_to_nested_dicts(col_order)

print(dumps(dict_tree, indent=4))
```

Output:

```
{
    "Apple": {
        "Spain": [
            {
                "Supplier": "Big Apples",
                "Fruit": "Apple",
                "Origin": "Spain",
                "Quantity": "1"
            }
        ],
        "Italy": [
            {
                "Supplier": "Square Apples",
                "Fruit": "Apple",
                "Origin": "Italy",
                "Quantity": "7"
            }
        ]
    },
    "Melons": {
        "Italy": [
            {
                "Supplier": "Big Melons",
                "Fruit": "Melons",
                "Origin": "Italy",
                "Quantity": "2"
            },
            {
                "Supplier": "Small Melons",
                "Fruit": "Melons",
                "Origin": "Italy",
                "Quantity": "8"
            }
        ]
    },
    "Mango": {
        "India": [
            {
                "Supplier": "Long Mangoes",
                "Fruit": "Mango",
                "Origin": "India",
                "Quantity": "3"
            }
        ],
        "France": [
            {
```

(continues on next page)

(continued from previous page)

```
                "Supplier": "Short Mangoes",
                "Fruit": "Mango",
                "Origin": "France",
                "Quantity": "5"
            }
        ]
    },
    "Strawberry": {
        "France": [
            {
                "Supplier": "Small Strawberries",
                "Fruit": "Strawberry",
                "Origin": "France",
                "Quantity": "4"
            }
        ],
        "Spain": [
            {
                "Supplier": "Sweet Strawberries",
                "Fruit": "Strawberry",
                "Origin": "Spain",
                "Quantity": "6"
            }
        ],
        "Australia": [
            {
                "Supplier": "Dark Berries",
                "Fruit": "Strawberry",
                "Origin": "Australia",
                "Quantity": "9"
            }
        ]
    },
    "Blackcurrant": {
        "Australia": [
            {
                "Supplier": "Sweet Berries",
                "Fruit": "Blackcurrant",
                "Origin": "Australia",
                "Quantity": "10"
            }
        ]
    }
}
```

Construct a dictionary with number of rows for each unique “Origin”

```
from csvio.csvreader import CSVReader
from json import dumps

reader = CSVReader("fruit_stock.csv")
```

(continues on next page)

(continued from previous page)

```
col_order = ["Origin"]

origin_fruit_count = {}
dict_tree = reader.rows_to_nested_dicts(col_order)

for origin in dict_tree:
    origin_fruit_count.setdefault(origin, len(dict_tree[origin]))

print(dumps(origin_fruit_count, indent=4))
```

Output:

```
{
    "Spain": 2,
    "Italy": 3,
    "India": 1,
    "France": 2,
    "Australia": 2
}
```

touch(*exist_ok: bool = False*) → bool

Create a blank file at the path provided in the *filename* parameter.

Parameters **exist_ok** (*optional*) – Parameter to pass to the `pathlib.Path.touch()` method.

Returns

`True` If blank file is created successfully.

`False` On failure.

5.1.3 CSVReader

class csvio.**CSVReader**(*filename: str*, *fieldnames: List[str] = []*, *open_kwargs: Dict[str, str] = {}*, *csv_kwargs: Dict[str, Any] = {}*)

Bases: *csvio.csvbase.CSVBase*

This object represents a CSV file for reading.

Parameters

- **filename** (*required*) – Full path to the CSV file for reading.
- **fieldnames** (*optional*) – A list of strings representing the column headings for the CSV file. If this list is specified while initiating an Object of this class then it is used as the column headings. This is handy when the CSV to read does not have column headings. Otherwise this list is populated from the CSV that is set in the *filename* argument of this Class's constructor.
- **open_kwargs** (*optional*) – A dictionary of key, value pairs that should be passed to the open method within this class.
- **csv_kwargs** (*optional*) – A dictionary of key, value pairs that should be passed to the DictReader constructor within this class.

Usage:

```
>>> from csvio import CSVReader
>>> reader = CSVReader("fruit_stock.csv")
>>> reader.fieldnames
['Supplier', 'Fruit', 'Quantity']

>>> len(reader.rows)
4

>>> import json
>>> print(json.dumps(reader.rows, indent=4))
[
    {
        "Supplier": "Big Apple",
        "Fruit": "Apple",
        "Quantity": "1"
    },
    {
        "Supplier": "Big Melons",
        "Fruit": "Melons",
        "Quantity": "2"
    },
    {
        "Supplier": "Big Mangoes",
        "Fruit": "Mango",
        "Quantity": "3"
    },
    {
        "Supplier": "Small Strawberries",
        "Fruit": "Strawberry",
        "Quantity": "4"
    }
]
```

CSV file contents:

```
Supplier,Fruit,Quantity
Big Apple,Apple,1
Big Melons,Melons,2
Long Mangoes,Mango,3
Small Strawberries,Strawberry,4
```

property csv_kwargs: Dict[str, Any]

Returns A dictionary of key, value pairs that should be passed to the DictReader constructor within this class.

delete(*missing_ok: bool = False*) → bool

Delete the file at the path provided in the *filename* parameter

Parameters **missing_ok** (*optional*) – Parameter to pass to the `pathlib.Path.unlink()` method.

Returns

`True` If file is deleted successfully.

`False` On failure.

property fieldnames: List[str]

Returns List of column headings

property file_ext: str

Returns Extension suffix of the file without parent directory and file name.

property filedir: str

Returns Parent directory path of the file (excluding the name of the file)

property filename: str

Returns File name without the parent directory path.

property filename_no_ext: str

Returns File name without parent directory and file extension.

property filepath: str

Returns Complete file path including the parent directory, file name and extension

property num_rows: int

Returns The total number of rows in the CSV (excluding column headings)

property open_kwargs: Dict[str, Any]

Returns A dictionary of key, value pairs that should be passed to the open method within this class.

property path_obj: pathlib.Path

Returns `pathlib.Path` object representing *filename*.

property rows: List[Dict[str, Any]]

Returns A list of dictionaries where each item in it represents a row in the CSV file. Each dictionary in the list maps the column heading (fieldname) to the corresponding value for it from the CSV.

rows_from_column_key(*column_name: str*, *rows: Optional[List[Dict[str, Any]]] = None*) → Dict[str, List[Dict[str, Any]]]

Collect all the rows in the `rows` parameter that have the same values for the column defined in the `column_name` parameter, and construct a dictionary with the `column_name` value as the key and the corresponding rows as a list of dictionaries, as the value of this key.

Parameters

- **column_name** (*required*) – Name of the column that is to be used as the key under which all the rows having the samee value of this column will be collected.
- **rows** (optional. If not provided `self.rows` will be used.) – List of dictionaries representing the rows that will be separated and collected under a the common value of the column name provided in `column_name` parameter.

Returns A dictionary constructed using the logic as explained above.

rows_to_nested_dicts(*column_order: List[str]*, *rows: Optional[List[Dict[str, Any]]] = None*) → Dict[str, Any]

Collect all values of columns that are the same and construct a nested dictionary that has the common values as the keys, in the same order of hierarchy as provided in the *column_order* parameter.

The value of the last column name in the *column_order* list

Parameters

- **column_order** (*required*) – An ordered list of column names, to be used for constructing the dictionary
- **rows** (optional. If not provided `self.rows` will be used.) – List of dictionaries representing the rows that will be transformed to the output Dictionary.

Returns A dictionary with same column values collected under a common key in a hierarchical order.

Example:

CSV Contents: *fruit_stock.csv*

```
Supplier,Fruit,Origin,Quantity
Big Apples,Apple,Spain,1
Big Melons,Melons,Italy,2
Long Mangoes,Mango,India,3
Small Strawberries,Strawberry,France,4
Short Mangoes,Mango,France,5
Sweet Strawberries,Strawberry,Spain,6
Square Apples,Apple,Italy,7
Small Melons,Melons,Italy,8
Dark Berries,Strawberry,Australia,9
Sweet Berries,Blackcurrant,Australia,10
```

Create dictionary with hierarchy `{"Fruit": [rows]}`

```
from csvio.csvreader import CSVReader
from json import dumps

reader = CSVReader("fruit_stock.csv")

col_order = ["Fruit"]

dict_tree= reader.rows_to_nested_dicts(col_order)

print(dumps(dict_tree, indent=4))
```

Output:

```
{
    "Apple": [
        {
            "Supplier": "Big Apples",
            "Fruit": "Apple",
            "Origin": "Spain",
            "Quantity": "1"
        },
        {
            "Supplier": "Square Apples",
            "Fruit": "Apple",
            "Origin": "Italy",
            "Quantity": "7"
        }
```

(continues on next page)

(continued from previous page)


```
    ],
    "Melons": [
        {
            "Supplier": "Big Melons",
            "Fruit": "Melons",
            "Origin": "Italy",
            "Quantity": "2"
        },
        {
            "Supplier": "Small Melons",
            "Fruit": "Melons",
            "Origin": "Italy",
            "Quantity": "8"
        }
    ],
    "Mango": [
        {
            "Supplier": "Long Mangoes",
            "Fruit": "Mango",
            "Origin": "India",
            "Quantity": "3"
        },
        {
            "Supplier": "Short Mangoes",
            "Fruit": "Mango",
            "Origin": "France",
            "Quantity": "5"
        }
    ],
    "Strawberry": [
        {
            "Supplier": "Small Strawberries",
            "Fruit": "Strawberry",
            "Origin": "France",
            "Quantity": "4"
        },
        {
            "Supplier": "Sweet Strawberries",
            "Fruit": "Strawberry",
            "Origin": "Spain",
            "Quantity": "6"
        },
        {
            "Supplier": "Dark Berries",
            "Fruit": "Strawberry",
            "Origin": "Australia",
            "Quantity": "9"
        }
    ],
    "Blackcurrant": [
        {
            "Supplier": "Sweet Berries",
```


(continues on next page)

(continued from previous page)

```
            "Fruit": "Blackcurrant",
            "Origin": "Australia",
            "Quantity": "10"
        }
    ]
}
```

Create dictionary with hierarchy `{"Fruit": "Origin" : [rows]}`

```
from csvio.csvreader import CSVReader
from json import dumps

reader = CSVReader("fruit_stock.csv")

col_order = ["Fruit", "Origin"]

dict_tree= reader.rows_to_nested_dicts(col_order)

print(dumps(dict_tree, indent=4))
```

Output:

```
{
    "Apple": {
        "Spain": [
            {
                "Supplier": "Big Apples",
                "Fruit": "Apple",
                "Origin": "Spain",
                "Quantity": "1"
            }
        ],
        "Italy": [
            {
                "Supplier": "Square Apples",
                "Fruit": "Apple",
                "Origin": "Italy",
                "Quantity": "7"
            }
        ]
    },
    "Melons": {
        "Italy": [
            {
                "Supplier": "Big Melons",
                "Fruit": "Melons",
                "Origin": "Italy",
                "Quantity": "2"
            },
            {
                "Supplier": "Small Melons",
                "Fruit": "Melons",
```

(continues on next page)

(continued from previous page)


```
                "Origin": "Italy",
                "Quantity": "8"
            }
        ]
    },
    "Mango": {
        "India": [
            {
                "Supplier": "Long Mangoes",
                "Fruit": "Mango",
                "Origin": "India",
                "Quantity": "3"
            }
        ],
        "France": [
            {
                "Supplier": "Short Mangoes",
                "Fruit": "Mango",
                "Origin": "France",
                "Quantity": "5"
            }
        ]
    },
    "Strawberry": {
        "France": [
            {
                "Supplier": "Small Strawberries",
                "Fruit": "Strawberry",
                "Origin": "France",
                "Quantity": "4"
            }
        ],
        "Spain": [
            {
                "Supplier": "Sweet Strawberries",
                "Fruit": "Strawberry",
                "Origin": "Spain",
                "Quantity": "6"
            }
        ],
        "Australia": [
            {
                "Supplier": "Dark Berries",
                "Fruit": "Strawberry",
                "Origin": "Australia",
                "Quantity": "9"
            }
        ]
    },
    "Blackcurrant": {
        "Australia": [
            {
```


(continues on next page)

(continued from previous page)

```
                "Supplier": "Sweet Berries",
                "Fruit": "Blackcurrant",
                "Origin": "Australia",
                "Quantity": "10"
            }
        ]
    }
}
```

Construct a dictionary with number of rows for each unique "Origin"

```
from csvio.csvreader import CSVReader
from json import dumps

reader = CSVReader("fruit_stock.csv")

col_order = ["Origin"]

origin_fruit_count = {}
dict_tree = reader.rows_to_nested_dicts(col_order)

for origin in dict_tree:
    origin_fruit_count.setdefault(origin, len(dict_tree[origin]))

print(dumps(origin_fruit_count, indent=4))
```

Output:

```
{
    "Spain": 2,
    "Italy": 3,
    "India": 1,
    "France": 2,
    "Australia": 2
}
```

touch(*exist_ok: bool = False*) → bool

Create a blank file at the path provided in the *filename* parameter.

Parameters **exist_ok** (*optional*) – Parameter to pass to the `pathlib.Path.touch()` method.

Returns

`True` If blank file is created successfully.

`False` On failure.

5.1.4 CSVWriter

class csvio.**CSVWriter**(*filename: str*, *fieldnames: List[str]*, *open_kwargs: Dict[str, str] = {}*, *csv_kwargs: Dict[str, Any] = {}*)

Bases: *csvio.csvbase.CSVBase*

This object represents a CSV file for writing.

Parameters

- **filename** (*required*) – Full path to the CSV file for writing.
- **fieldnames** (*required*) – A list of strings representing the column headings for the CSV file.
- **open_kwargs** (*optional*) – A dictionary of key, value pairs that should be passed to the open method within this class.
- **csv_kwargs** (*optional*) – A dictionary of key, value pairs that should be passed to the DictReader constructor within this class.

add_rows(*rows: Union[Dict[str, Any], List[Dict[str, Any]]]*) → None

Add rows for writing to the output CSV.

All the rows to be written to the output CSV are collected using this method.

This only collects the rows to be written without writing anything to the output CSV. The rows are written to output only when the method *csvio.CSVWriter.flush()* is called.

Parameters **rows** (*required*) – A single dictionary or a list of dictionaries that repsresent the row(s) to be written to the output CSV.

Usage:

```
>>> from csvio import CSVWriter
>>> writer = CSVWriter("fruit_stock.csv", fieldnames=["Supplier", "Fruit",
↪"Quantity"])
>>> row1 = {"Supplier": "Big Apple", "Fruit": "Apple", "Quantity": 1}
>>> writer.add_rows(row1)
>>> rows2_3_4 = [
...     {"Supplier": "Big Melons", "Fruit": "Melons", "Quantity": 2},
...     {"Supplier": "Long Mangoes", "Fruit": "Mango", "Quantity": 3},
...     {"Supplier": "Small Strawberries", "Fruit": "Strawberry", "Quantity": 4}
... ]
>>> writer.add_rows(rows2_3_4)
>>> len(writer.pending_rows)
4

>>> len(writer.rows)
0
```

Notice that the *csvio.CSVWriter.rows* property is still empty. This property is incremented by the number of currently pending rows once they are flushed.

property csv_kwargs: Dict[str, Any]

Returns A dictionary of key, value pairs that should be passed to the DictReader constructor within this class.

delete(*missing_ok: bool = False*) → bool

Delete the file at the path provided in the *filename* parameter

Parameters **missing_ok** (*optional*) – Parameter to pass to the `pathlib.Path.unlink()` method.

Returns

`True` If file is deleted successfully.

`False` On failure.

property fieldnames: List[str]

Returns List of column headings

property file_ext: str

Returns Extension suffix of the file without parent directory and file name.

property filedir: str

Returns Parent directory path of the file (excluding the name of the file)

property filename: str

Returns File name without the parent directory path.

property filename_no_ext: str

Returns File name without parent directory and file extension.

property filepath: str

Returns Complete file path including the parent directory, file name and extension

flush() → None

Write pending rows to the output CSV and reset the *CSVWriter.pending_rows* property to an empty `list`

Usage:

```
>>> from csvio import CSVWriter
>>> writer = CSVWriter("fruit_stock.csv", fieldnames=["Supplier", "Fruit",
↪"Quantity"])
>>> row1 = {"Supplier": "Big Apple", "Fruit": "Apple", "Quantity": 1}
>>> writer.add_rows(row1)
>>> rows2_3_4 = [
...     {"Supplier": "Big Melons", "Fruit": "Melons", "Quantity": 2},
...     {"Supplier": "Long Mangoes", "Fruit": "Mango", "Quantity": 3},
...     {"Supplier": "Small Strawberries", "Fruit": "Strawberry", "Quantity": 4}
... ]
>>> writer.add_rows(rows2_3_4)
>>> len(writer.pending_rows)
4

>>> len(writer.rows)
0

>>> writer.flush()
>>> len(writer.pending_rows)
0

>>> len(writer.rows)
4
```

Once flush is called a CSV file with the name *fruit_stock.csv* will be written with the following contents.

```
Supplier,Fruit,Quantity
Big Apple,Apple,1
Big Melons,Melons,2
Long Mangoes,Mango,3
Small Strawberries,Strawberry,4
```

property num_rows: int

Returns The total number of rows in the CSV (excluding column headings)

property open_kwargs: Dict[str, Any]

Returns A dictionary of key, value pairs that should be passed to the open method within this class.

property path_obj: pathlib.Path

Returns `pathlib.Path` object representing *filename*.

property pending_rows: List[Dict[str, Any]]

Returns List of rows not flushed yet and are pending to be written

property rows: List[Dict[str, Any]]

Returns A list of dictionaries where each item in it represents a row in the CSV file. Each dictionary in the list maps the column heading (fieldname) to the corresponding value for it from the CSV.

rows_from_column_key(*column_name: str*, *rows: Optional[List[Dict[str, Any]]] = None*) → Dict[str, List[Dict[str, Any]]]

Collect all the rows in the `rows` parameter that have the same values for the column defined in the `column_name` parameter, and construct a dictionary with the `column_name` value as the key and the corresponding rows as a list of dictionaries, as the value of this key.

Parameters

- **column_name** (*required*) – Name of the column that is to be used as the key under which all the rows having the samee value of this column will be collected.
- **rows** (optional. If not provided `self.rows` will be used.) – List of dictionaries representing the rows that will be separated and collected under a the common value of the column name provided in `column_name` parameter.

Returns A dictionary constructed using the logic as explained above.

rows_to_nested_dicts(*column_order: List[str]*, *rows: Optional[List[Dict[str, Any]]] = None*) → Dict[str, Any]

Collect all values of columns that are the same and construct a nested dictionary that has the common values as the keys, in the same order of hierarchy as provided in the *column_order* parameter.

The value of the last column name in the *column_order* list

Parameters

- **column_order** (*required*) – An ordered list of column names, to be used for constructing the dictionary
- **rows** (optional. If not provided `self.rows` will be used.) – List of dictionaries representing the rows that will be transformed to the output Dictionary.

Returns A dictionary with same column values collected under a common key in a hierarchical order.

Example:

CSV Contents: *fruit_stock.csv*

```
Supplier,Fruit,Origin,Quantity
Big Apples,Apple,Spain,1
Big Melons,Melons,Italy,2
Long Mangoes,Mango,India,3
Small Strawberries,Strawberry,France,4
Short Mangoes,Mango,France,5
Sweet Strawberries,Strawberry,Spain,6
Square Apples,Apple,Italy,7
Small Melons,Melons,Italy,8
Dark Berries,Strawberry,Australia,9
Sweet Berries,Blackcurrant,Australia,10
```

Create dictionary with hierarchy `{"Fruit": [rows]}`

```
from csvio.csvreader import CSVReader
from json import dumps

reader = CSVReader("fruit_stock.csv")

col_order = ["Fruit"]

dict_tree= reader.rows_to_nested_dicts(col_order)

print(dumps(dict_tree, indent=4))
```

Output:

```
{
    "Apple": [
        {
            "Supplier": "Big Apples",
            "Fruit": "Apple",
            "Origin": "Spain",
            "Quantity": "1"
        },
        {
            "Supplier": "Square Apples",
            "Fruit": "Apple",
            "Origin": "Italy",
            "Quantity": "7"
        }
    ],
    "Melons": [
        {
            "Supplier": "Big Melons",
            "Fruit": "Melons",
            "Origin": "Italy",
            "Quantity": "2"
```

(continues on next page)

(continued from previous page)


```
        },
        {
            "Supplier": "Small Melons",
            "Fruit": "Melons",
            "Origin": "Italy",
            "Quantity": "8"
        }
    ],
    "Mango": [
        {
            "Supplier": "Long Mangoes",
            "Fruit": "Mango",
            "Origin": "India",
            "Quantity": "3"
        },
        {
            "Supplier": "Short Mangoes",
            "Fruit": "Mango",
            "Origin": "France",
            "Quantity": "5"
        }
    ],
    "Strawberry": [
        {
            "Supplier": "Small Strawberries",
            "Fruit": "Strawberry",
            "Origin": "France",
            "Quantity": "4"
        },
        {
            "Supplier": "Sweet Strawberries",
            "Fruit": "Strawberry",
            "Origin": "Spain",
            "Quantity": "6"
        },
        {
            "Supplier": "Dark Berries",
            "Fruit": "Strawberry",
            "Origin": "Australia",
            "Quantity": "9"
        }
    ],
    "Blackcurrant": [
        {
            "Supplier": "Sweet Berries",
            "Fruit": "Blackcurrant",
            "Origin": "Australia",
            "Quantity": "10"
        }
    ]
}
```


Create dictionary with hierarchy `{"Fruit": "Origin" : [rows]}`

```
from csvio.csvreader import CSVReader
from json import dumps

reader = CSVReader("fruit_stock.csv")

col_order = ["Fruit", "Origin"]

dict_tree= reader.rows_to_nested_dicts(col_order)

print(dumps(dict_tree, indent=4))
```

Output:

```
{
    "Apple": {
        "Spain": [
            {
                "Supplier": "Big Apples",
                "Fruit": "Apple",
                "Origin": "Spain",
                "Quantity": "1"
            }
        ],
        "Italy": [
            {
                "Supplier": "Square Apples",
                "Fruit": "Apple",
                "Origin": "Italy",
                "Quantity": "7"
            }
        ]
    },
    "Melons": {
        "Italy": [
            {
                "Supplier": "Big Melons",
                "Fruit": "Melons",
                "Origin": "Italy",
                "Quantity": "2"
            },
            {
                "Supplier": "Small Melons",
                "Fruit": "Melons",
                "Origin": "Italy",
                "Quantity": "8"
            }
        ]
    },
    "Mango": {
        "India": [
            {
                "Supplier": "Long Mangoes",
                "Fruit": "Mango",
```

(continues on next page)

(continued from previous page)


```
                "Origin": "India",
                "Quantity": "3"
            }
        ],
        "France": [
            {
                "Supplier": "Short Mangoes",
                "Fruit": "Mango",
                "Origin": "France",
                "Quantity": "5"
            }
        ]
    },
    "Strawberry": {
        "France": [
            {
                "Supplier": "Small Strawberries",
                "Fruit": "Strawberry",
                "Origin": "France",
                "Quantity": "4"
            }
        ],
        "Spain": [
            {
                "Supplier": "Sweet Strawberries",
                "Fruit": "Strawberry",
                "Origin": "Spain",
                "Quantity": "6"
            }
        ],
        "Australia": [
            {
                "Supplier": "Dark Berries",
                "Fruit": "Strawberry",
                "Origin": "Australia",
                "Quantity": "9"
            }
        ]
    },
    "Blackcurrant": {
        "Australia": [
            {
                "Supplier": "Sweet Berries",
                "Fruit": "Blackcurrant",
                "Origin": "Australia",
                "Quantity": "10"
            }
        ]
    }
}
```


Construct a dictionary with number of rows for each unique “Origin”

```
from csvio.csvreader import CSVReader
from json import dumps

reader = CSVReader("fruit_stock.csv")

col_order = ["Origin"]

origin_fruit_count = {}
dict_tree = reader.rows_to_nested_dicts(col_order)

for origin in dict_tree:
    origin_fruit_count.setdefault(origin, len(dict_tree[origin]))

print(dumps(origin_fruit_count, indent=4))
```

Output:

```
{
    "Spain": 2,
    "Italy": 3,
    "India": 1,
    "France": 2,
    "Australia": 2
}
```

touch(*exist_ok: bool = False*) → bool
Create a blank file at the path provided in the *filename* parameter.

Parameters **exist_ok** (*optional*) – Parameter to pass to the `pathlib.Path.touch()` method.

Returns

`True` If blank file is created successfully.

`False` On failure.

write_blank_csv() → None
Write a blank CSV with only the column headings.

If the CSV already exists with any rows in it, it will be overwritten and its contents will be replaced with only the column headings.

- genindex
- search

5.2 CHANGE LOG

2021-10-03

Version 0.2.0

- Construct a nested dictionary from rows based on a list of ordered column names … read more
- Update/Fix documentation
- Refactor/Add tests

2021-09-24

Version 0.1.0

- First release

INDEX

W