
pytablereader Documentation

Release 0.31.1

Tsuyoshi Hombashi

Oct 17, 2021

TABLE OF CONTENTS

1	pytablereader	1
1.1	Summary	1
1.2	Features	1
2	Installation	3
2.1	Install from PyPI	3
2.2	Install from PPA (for Ubuntu)	3
3	Dependencies	5
3.1	Optional Python packages	5
3.2	Optional packages (other than Python packages)	5
4	Examples	7
4.1	Load table data from CSV	7
4.2	Load table data from a web page	8
4.3	Load table data from Google Sheets	9
4.4	Get loaded table data as pandas.DataFrame	9
5	Reference	11
5.1	Table Loader Wrapper Classes	11
5.1.1	File Loader Wrapper	11
5.1.2	Text Loader Wrapper	12
5.1.3	URL Loader Wrapper	13
5.2	Format Specific Table Loader Classes	14
5.2.1	AbstractTableReader class	14
5.2.2	CSV Loader Classes	15
5.2.2.1	CSV Table Loader	15
5.2.2.2	CSV File Loader	15
5.2.2.3	CSV Text Loader	16
5.2.3	HTML Loader Classes	16
5.2.3.1	HTML File Loader	16
5.2.3.2	HTML Text Loader	17
5.2.4	JSON Loader Classes	18
5.2.4.1	Json File Loader	18
5.2.4.2	Json Text Loader	24
5.2.4.3	Line-delimited Json File Loader	25
5.2.4.4	Line-delimited Json Text Loader	25
5.2.5	LTSV Loader Classes	26
5.2.5.1	LTSV File Loader	26
5.2.5.2	LTSV Text Loader	26

5.2.6	Markdown Loader Classes	27
5.2.6.1	Markdown File Loader	27
5.2.6.2	Markdown Text Loader	28
5.2.7	MediaWiki Loader Classes	28
5.2.7.1	MediaWiki File Loader	28
5.2.7.2	MediaWiki Text Loader	29
5.2.8	Spread Sheet Loader Classes	30
5.2.8.1	Excel File Loader	30
5.2.8.2	Google Sheets Loader	30
5.2.9	Database Loader Classes	31
5.2.9.1	SQLite File Loader	31
5.3	Table Loader Factory Classes	32
5.3.1	File Loader Factory	32
5.3.2	Text Loader Factory	33
5.3.3	Url Loader Factory	35
5.4	Exceptions	36
6	Changelog	39
7	Sponsors	41
8	Indices and tables	43
9	Links	45
10	Indices and tables	47
	Index	49

PYTABLEREADER

1.1 Summary

pytablereader is a Python library to load structured table data from files/strings/URL with various data format: CSV / Excel / Google-Sheets / HTML / JSON / LDJSON / LTSV / Markdown / SQLite / TSV.

1.2 Features

- **Extract structured tabular data from various data format:**
 - CSV / Tab separated values (TSV) / Space separated values (SSV)
 - Microsoft Excel™ file
 - Google Sheets
 - HTML (table tags)
 - JSON
 - Labeled Tab-separated Values (LTSV)
 - Line-delimited JSON(LDJSON) / NDJSON / JSON Lines
 - Markdown
 - MediaWiki
 - SQLite database file
- **Supported data sources are:**
 - Files on a local file system
 - Accessible URLs

- str instances
- **Loaded table data can be used as:**
 - `pandas.DataFrame` instance
 - dict instance

INSTALLATION

2.1 Install from PyPI

```
pip install pytablereader
```

Some of the formats require additional dependency packages, you can install the dependency packages as follows:

- **Excel**
 - pip install pytablereader[excel]
- **Google Sheets**
 - pip install pytablereader[gs]
- **Markdown**
 - pip install pytablereader[md]
- **Mediawiki**
 - pip install pytablereader[mediawiki]
- **SQLite**
 - pip install pytablereader[sqlite]
- **Load from URLs**
 - pip install pytablereader[url]
- **All of the extra dependencies**
 - pip install pytablereader[all]

2.2 Install from PPA (for Ubuntu)

```
sudo add-apt-repository ppa:thombashi/ppa  
sudo apt update  
sudo apt install python3-pytablereader
```


DEPENDENCIES

- Python 3.6+
- Python package dependencies (automatically installed)

3.1 Optional Python packages

- **logging extras**
 - `loguru`: Used for logging if the package installed
- **excel extras**
 - `excelrd`
- **md extras**
 - `Markdown`
- **mediawiki extras**
 - `py pandoc`
- **sqlite extras**
 - `SimpleSQLite`
- **url extras**
 - `retryrequests`
- **pandas**
 - required to get table data as a pandas data frame
- `lxml`

3.2 Optional packages (other than Python packages)

- `libxml2` (faster HTML conversion)
- `pandoc` (required when loading MediaWiki file)

4.1 Load table data from CSV

Following example shows how to extract `TableData` from CSV data by using `CsvTableFileLoader` and `CsvTableTextLoader` classes.

Sample Code

Listing 1: Load table from CSV

```
import pytablereader as ptr
import pytablewriter as ptw

# prepare data ---
file_path = "sample_data.csv"
csv_text = "\n".join([
    "attr_a","attr_b","attr_c",
    '1,4,"a"',
    '2,2.1,"bb"',
    '3,120.9,"ccc"',
])

with open(file_path, "w") as f:
    f.write(csv_text)

# load from a csv file ---
loader = ptr.CsvTableFileLoader(file_path)
for table_data in loader.load():
    print("\n".join([
        "load from file",
        "=====",
        "{:s}".format(ptw.dumps_tabledata(table_data)),
    ]))

# load from a csv text ---
loader = ptr.CsvTableTextLoader(csv_text)
for table_data in loader.load():
    print("\n".join([
        "load from text",
        "=====",
```

(continues on next page)

(continued from previous page)

```
    "{:s}".format(ptw.dumps_tabledata(table_data)),
  ]))
```

Output

```
load from file
=====
.. table:: sample_data

=====  =====  =====
attr_a  attr_b  attr_c
=====  =====  =====
      1      4.0  a
      2      2.1  bb
      3     120.9  ccc
=====  =====  =====

load from text
=====
.. table:: csv2

=====  =====  =====
attr_a  attr_b  attr_c
=====  =====  =====
      1      4.0  a
      2      2.1  bb
      3     120.9  ccc
=====  =====  =====
```

4.2 Load table data from a web page

Following example shows how to extract `TableData` from a web page by using `TableUrlLoader` class.

Sample Code

Listing 2: Load table from a web page

```
import io

import pytablereader as ptr
import pytablewriter as ptw

loader = ptr.TableUrlLoader(
    "https://en.wikipedia.org/wiki/List_of_unit_testing_frameworks",
    "html")

writer = ptw.TableWriterFactory.create_from_format_name("rst")
writer.stream = io.open("load_url_result.rst", "w", encoding=loader.
    encoding)
for table_data in loader.load():
```

(continues on next page)

(continued from previous page)

```
writer.from_tabledata(table_data)
writer.write_table()
```

Output

```
$ ./load_table_from_url.py
$ head load_url_result.rst -n 8
.. table:: List of unit testing frameworks - Wikipedia_html1

+-----+-----+-----+-----+
| Name  |xUnit|Source|      Remarks      |
+-----+-----+-----+-----+
|ABAP Unit|Yes  |[1]  |since SAP NetWeaver 2004|
+-----+-----+-----+-----+
```

4.3 Load table data from Google Sheets

Following example shows how to extract TableData from Google Sheets by using *GoogleSheetsTableLoader* class.

Listing 3: Load table data from Google Sheets

```
import io

import pytablereader as ptr
import pytablewriter as ptw

loader = ptr.TableUrlLoader(
    "https://en.wikipedia.org/wiki/List_of_unit_testing_frameworks",
    "html")

writer = ptw.TableWriterFactory.create_from_format_name("rst")
writer.stream = io.open("load_url_result.rst", "w", encoding=loader.encoding)
for table_data in loader.load():
    writer.from_tabledata(table_data)
    writer.write_table()
```

4.4 Get loaded table data as pandas.DataFrame

A TableData instance can be converted to a pandas.DataFrame instance by `as_dataframe()`.

Sample Code

Listing 4: Convert from loaded tabledata.TableData to pandas.DataFrame

```
import pytablereader as ptr

loader = ptr.CsvTableTextLoader(
    "\n".join([
```

(continues on next page)

(continued from previous page)

```
        "a,b",
        "1,2",
        "3.3,4.4",
    ]])
for table_data in loader.load():
    print(table_data.as_dataframe())
```

Output

	a	b
0	1	2
1	3.3	4.4

5.1 Table Loader Wrapper Classes

5.1.1 File Loader Wrapper

class `pytablereader.TableFileLoader`(*file_path*, *format_name=None*, *encoding=None*,
type_hint_rules=None)

Loader class to loading tables from a file.

Parameters

- **file_path** (*str*) – Path to the file to load.
- **format_name** (*str*) – Data format name to load. Supported formats are as follows: "csv", "excel", "html", "json", "ltsv", "markdown", "mediawiki", "sqlite", "ssv", "tsv". If the value is `None`, automatically detect file format from the `file_path`.

Raises

- `pytablereader.InvalidFilePathError` – If `file_path` is an invalid file path.
- `pytablereader.LoaderNotFoundError` – If an appropriate loader not found for loading the file.

`load()`

Loading table data from a file as `format_name` format. Automatically detect file format if `format_name` is `None`.

Returns Loaded table data iterator.

Return type `TableData` iterator

See also:

- `pytablereader.factory.TableFileLoaderFactory.create_from_format_name()`
- `pytablereader.factory.TableFileLoaderFactory.create_from_path()`

classmethod `get_format_names()`

Returns Available format names. These names can use by `TableFileLoader` class constructor.

Return type `list`

Example

```
>>> from pytablereader import TableFileLoader
>>> for format_name in TableFileLoader.get_format_names():
...     print(format_name)
...
csv
excel
html
json
json_lines
jsonl
ldjson
ltsv
markdown
mediawiki
ndjson
sqlite
ssv
tsv
```

5.1.2 Text Loader Wrapper

class `pytablereader.TableTextLoader`(*source: str, format_name: str, encoding: Optional[str] = None, type_hint_rules=None*)

Loader class to loading tables from URL.

Parameters

- **url** (*str*) – URL to load.
- **format_name** (*str*) – Data format name to load. Supported formats can be get by `get_format_names()`
- **proxies** (*dict*) – http/https proxy information.

See also:

`requests proxies`

Raises `pytablereader.LoaderNotFoundError` – If an appropriate loader not found for loading the URL.

`load()`

Load tables from text as `format_name` format.

Returns Loaded table data iterator.

Return type `TableData` iterator

See also:

- `pytablereader.factory.TableTextLoaderFactory.create_from_format_name()`
- `pytablereader.factory.TableTextLoaderFactory.create_from_path()`

classmethod `get_format_names()` → `Sequence[str]`

Returns Available format names. These names can use by `TableTextLoader` class constructor.

Return type `list`

Example

```
>>> from pytablereader import TableTextLoader
>>> for format_name in TableTextLoader.get_format_names():
...     print(format_name)
...
csv
excel
html
json
json_lines
jsonl
ldjson
ltsv
markdown
mediawiki
ndjson
sqlite
ssv
tsv
```

5.1.3 URL Loader Wrapper

class `pytablereader.TableUrlLoader`(*url*, *format_name=None*, *encoding=None*, *type_hint_rules=None*, *proxies=None*)

Loader class to loading tables from URL.

Parameters

- **url** (*str*) – URL to load.
- **format_name** (*str*) – Data format name to load. Supported formats are: "csv", "excel", "html", "json", "ltsv", "markdown", "mediawiki", "sqlite", "ssv", "tsv". If the value is `None`, automatically detect file format from the url.
- **proxies** (*dict*) – http/https proxy information.

See also:

`requests proxies`

Raises

- `pytablereader.LoaderNotFoundError` – If an appropriate loader not found for loading the URL.
- `pytablereader.HTTPError` – If loader received an HTTP error when access to the URL.

Example *Load table data from a web page*

`load()`

Load tables from URL as `format_name` format.

Returns Loaded table data iterator.

Return type `TableData` iterator

See also:

- `pytablereader.factory.TableUrlLoaderFactory.create_from_format_name()`
- `pytablereader.factory.TableUrlLoaderFactory.create_from_path()`

classmethod `get_format_names()`

Returns Available format names. These names can use by `TableUrlLoader` class constructor.

Return type `list`

Example

```
>>> from pytablereader import TableUrlLoader
>>> for format_name in TableUrlLoader.get_format_names():
...     print(format_name)
...
csv
excel
html
json
json_lines
jsonl
ldjson
ltsv
markdown
mediawiki
ndjson
sqlite
ssv
tsv
```

5.2 Format Specific Table Loader Classes

5.2.1 AbstractTableReader class

class `pytablereader.interface.AbstractTableReader`(*source, quoting_flags, type_hints, type_hint_rules=None*)

Bases: `pytablereader.interface.TableLoaderInterface`

The abstract class of table data file loader.

table_name

Table name string.

source

Table data source to load.

5.2.2 CSV Loader Classes

5.2.2.1 CSV Table Loader

class `pytablereader.csv.core.CsvTableLoader`(*source*, *quoting_flags*, *type_hints*, *type_hint_rules*)
The abstract class of CSV table loaders.

headers

Attribute names of the table. Use the first line of the CSV file as attribute list if `headers` is empty.

delimiter

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

quotechar

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

encoding

Encoding of the CSV data.

5.2.2.2 CSV File Loader

class `pytablereader.CsvTableFileLoader`(*file_path*, *quoting_flags=None*, *type_hints=None*,
type_hint_rules=None)

Bases: `pytablereader.csv.core.CsvTableLoader`

A file loader class to extract tabular data from CSV files.

Parameters `file_path` (*str*) – Path to the loading CSV file.

table_name

Table name string. Defaults to `%(filename)s`.

Examples *Load table data from CSV*

load()

Extract tabular data as `TableData` instances from a CSV file. `source` attribute should contain a path to the file to load.

Returns

Loaded table data. Table name determined by the value of `table_name`. Following format specifiers in the `table_name` are replaced with specific strings:

Format specifier	Value after the replacement
<code>%(filename)s</code>	Filename (without extension)
<code>%(format_name)s</code>	<code>"csv"</code>
<code>%(format_id)s</code>	A unique number between the same format.
<code>%(global_id)s</code>	A unique number between all of the format.

Return type `TableData` iterator

Raises `pytablereader.DataError` – If the CSV data is invalid.

See also:

`csv.reader()`

5.2.2.3 CSV Text Loader

class `pytablereader.CsvTableTextLoader`(*text*, *quoting_flags=None*, *type_hints=None*,
type_hint_rules=None)

Bases: `pytablereader.csv.core.CsvTableLoader`

A text loader class to extract tabular data from CSV text data.

Parameters `text` (*str*) – CSV text to load.

table_name

Table name string. Defaults to `%(format_name)s%(format_id)s`.

Examples *Load table data from CSV*

load()

Extract tabular data as `TableData` instances from a CSV text object. `source` attribute should contain a text object to load.

Returns

Loaded table data. Table name determined by the value of `table_name`. Following format specifiers in the `table_name` are replaced with specific strings:

Format specifier	Value after the replacement
<code>%(filename)s</code>	<code>""</code>
<code>%(format_name)s</code>	<code>"csv"</code>
<code>%(format_id)s</code>	A unique number between the same format.
<code>%(global_id)s</code>	A unique number between all of the format.

Return type `TableData` iterator

Raises `pytablereader.DataError` – If the CSV data is invalid.

See also:

`csv.reader()`

5.2.3 HTML Loader Classes

5.2.3.1 HTML File Loader

class `pytablereader.HtmlTableFileLoader`(*file_path=None*, *quoting_flags=None*, *type_hints=None*,
type_hint_rules=None)

A file loader class to extract tabular data from HTML files.

Parameters `file_path` (*str*) – Path to the loading HTML file.

table_name

Table name string. Defaults to `%(title)s_(key)s`.

encoding

HTML file encoding. Defaults to `"utf-8"`.

load()

Extract tabular data as `TableData` instances from HTML table tags in a HTML file. `source` attribute should contain a path to the file to load.

Returns

Loaded table data iterator. Table name determined by the value of `table_name`. Following format specifiers in the `table_name` are replaced with specific strings:

Format specifier	Value after the replacement
<code>%(filename)s</code>	Filename (without extension)
<code>%(title)s</code>	<title> tag value of the HTML.
<code>%(key)s</code>	This replaced to: (1) id attribute of the table tag (2) <code>%(format_name)s%(format_id)s</code> if id attribute not present in the table tag.
<code>%(format_name)s</code>	"html"
<code>%(format_id)s</code>	A unique number between the same format.
<code>%(global_id)s</code>	A unique number between all of the format.

Return type TableData iterator

Raises `pytablereader.DataError` – If the HTML data is invalid or empty.

Note: Table tag attributes ignored with loaded TableData.

5.2.3.2 HTML Text Loader

class `pytablereader.HtmlTableTextLoader`(*text*, *quoting_flags=None*, *type_hints=None*, *type_hint_rules=None*)

A text loader class to extract tabular data from HTML text data.

Parameters `text` (*str*) – HTML text to load.

table_name

Table name string. Defaults to `%(title)s_%(key)s`.

load()

Extract tabular data as TableData instances from HTML table tags in a HTML text object. `source` attribute should contain a text object to load.

Returns

Loaded table data iterator. Table name determined by the value of `table_name`. Following format specifiers in the `table_name` are replaced with specific strings:

Format specifier	Value after the replacement
%(filename)s	""
%(title)s	<title> tag value of the HTML.
%(key)s	This replaced to: (1) id attribute of the table tag (2) %(format_name)s%(format_id)s if id attribute is not included in the table tag.
%(format_name)s	"html"
%(format_id)s	A unique number between the same format.
%(global_id)s	A unique number between all of the format.

Return type TableData iterator

Raises `pytablereader.DataError` – If the HTML data is invalid or empty.

5.2.4 JSON Loader Classes

5.2.4.1 Json File Loader

`class pytablereader.JsonTableFileLoader`(*file_path=None, quoting_flags=None, type_hints=None, type_hint_rules=None*)

A file loader class to extract tabular data from JSON files.

Parameters `file_path` (*str*) – Path to the loading JSON file.

table_name

Table name string. Defaults to `%(filename)s_%(key)s`.

load()

Extract tabular data as TableData instances from a JSON file. `source` attribute should contain a path to the file to load.

This method can be loading four types of JSON formats:

(1) Single table data in a file:

Listing 1: Acceptable JSON Schema (1): single table

```
{
  "type": "array",
  "items": {
    "type": "object",
    "additionalProperties": {
      "anyOf": [
        {"type": "string"},
        {"type": "number"},
        {"type": "boolean"},
        {"type": "null"}
      ]
    }
  }
}
```

(continues on next page)

(continued from previous page)

```
}
}
```

Listing 2: Acceptable JSON example for the JSON schema (1)

```
[
  {"attr_b": 4, "attr_c": "a", "attr_a": 1},
  {"attr_b": 2.1, "attr_c": "bb", "attr_a": 2},
  {"attr_b": 120.9, "attr_c": "ccc", "attr_a": 3}
]
```

The example data will be loaded as the following tabular data:

attr_a	attr_b	attr_c
1	4.0	a
2	2.1	bb
3	120.9	ccc

(2) Single table data in a file:

Listing 3: Acceptable JSON Schema (2): single table

```
{
  "type": "object",
  "additionalProperties": {
    "type": "array",
    "items": {
      "anyOf": [
        {"type": "string"},
        {"type": "number"},
        {"type": "boolean"},
        {"type": "null"}
      ]
    }
  }
}
```

Listing 4: Acceptable JSON example for the JSON schema (2)

```
{
  "attr_a": [1, 2, 3],
  "attr_b": [4, 2.1, 120.9],
  "attr_c": ["a", "bb", "ccc"]
}
```

The example data will be loaded as the following tabular data:

attr_a	attr_b	attr_c
1	4.0	a
2	2.1	bb
3	120.9	ccc

(3) Single table data in a file:

```
:caption: Acceptable JSON Schema (3): single table

{
  "type": "object",
  "additionalProperties": {
    "anyOf": [
      {"type": "string"},
      {"type": "number"},
      {"type": "boolean"},
      {"type": "null"}
    ]
  }
}
```

Listing 5: Acceptable JSON example for the JSON schema (3)

```
{
  "num_ratings": 27,
  "support_threads": 1,
  "downloaded": 925716,
  "last_updated": "2017-12-01 6:22am GMT",
  "added": "2010-01-20",
  "num": 1.1,
  "hoge": null
}
```

The example data will be loaded as the following tabular data:

key	value
num_ratings	27
support_threads	1
downloaded	925716
last_updated	2017-12-01 6:22am GMT
added	2010-01-20
num	1.1
hoge	None

(4) Multiple table data in a file:

Listing 6: Acceptable JSON Schema (4): multiple tables

```
{
  "type": "object",
  "additionalProperties": {
    "type": "array",
    "items": {
      "type": "object",
      "additionalProperties": {
        "anyOf": [
          {"type": "string"},
          {"type": "number"},

```

(continues on next page)

(continued from previous page)

```

        {"type": "boolean"},
        {"type": "null"}
    ]
}

```

Listing 7: Acceptable JSON example for the JSON schema (4)

```

{
  "table_a" : [
    {"attr_b": 4, "attr_c": "a", "attr_a": 1},
    {"attr_b": 2.1, "attr_c": "bb", "attr_a": 2},
    {"attr_b": 120.9, "attr_c": "ccc", "attr_a": 3}
  ],
  "table_b" : [
    {"a": 1, "b": 4},
    {"a": 2 },
    {"a": 3, "b": 120.9}
  ]
}

```

The example data will be loaded as the following tabular data:

Table 1: table_a

attr_a	attr_b	attr_c
1	4.0	a
2	2.1	bb
3	120.9	ccc

Table 2: table_b

a	b
1	4.0
2	None
3	120.9

(5) Multiple table data in a file:

Listing 8: Acceptable JSON Schema (5): multiple tables

```

{
  "type": "object",
  "additionalProperties": {
    "type": "object",
    "additionalProperties": {
      "type": "array",
      "items": {
        "anyOf": [
          {"type": "string"},

```

(continues on next page)

(continued from previous page)

```

        {"type": "number"},
        {"type": "boolean"},
        {"type": "null"}
    ]
}

```

Listing 9: Acceptable JSON example for the JSON schema (5)

```

{
  "table_a" : {
    "attr_a": [1, 2, 3],
    "attr_b": [4, 2.1, 120.9],
    "attr_c": ["a", "bb", "ccc"]
  },
  "table_b" : {
    "a": [1, 3],
    "b": [4, 120.9]
  }
}

```

The example data will be loaded as the following tabular data:

Table 3: table_a

attr_a	attr_b	attr_c
1	4.0	a
2	2.1	bb
3	120.9	ccc

Table 4: table_b

a	b
1	4.0
3	120.9

(6) Multiple table data in a file:

Listing 10: Acceptable JSON Schema (6): multiple tables

```

{
  "type": "object",
  "additionalProperties": {
    "type": "object",
    "additionalProperties": {
      "anyOf": [
        {"type": "string"},
        {"type": "number"},
        {"type": "boolean"},
        {"type": "null"}
      ]
    }
  }
}

```

(continues on next page)

(continued from previous page)

```

    }
  ]
}

```

Listing 11: Acceptable JSON example for the JSON schema (6)

```

{
  "table_a": {
    "num_ratings": 27,
    "support_threads": 1,
    "downloaded": 925716,
    "last_updated": "2017-12-01 6:22am GMT",
    "added": "2010-01-20",
    "num": 1.1,
    "hoge": null
  },
  "table_b": {
    "a": 4,
    "b": 120.9
  }
}

```

The example data will be loaded as the following tabular data:

Table 5: table_a

key	value
num_ratings	27
support_threads	1
downloaded	925716
last_updated	2017-12-01 6:22am GMT
added	2010-01-20
num	1.1
hoge	None

Table 6: table_b

key	value
a	4.0
b	120.9

Returns

Loaded table data iterator. Table name determined by the value of `table_name`. Following format specifiers in the `table_name` are replaced with specific strings:

Format specifier	Value after the replacement
<code>%(filename)s</code>	Filename (without extension)
<code>%(key)s</code>	This replaced the different value for each single/multiple JSON tables: [single JSON table] <code>%(format_name)s%(format_id)s</code> [multiple JSON table] Table data key.
<code>%(format_name)s</code>	"json"
<code>%(format_id)s</code>	A unique number between the same format.
<code>%(global_id)s</code>	A unique number between all of the format.

Return type TableData iterator

Raises

- `pytablereader.DataError` – If the data is invalid JSON.
- `pytablereader.error.ValidationError` – If the data is not acceptable JSON format.

5.2.4.2 Json Text Loader

class `pytablereader.JsonTableTextLoader`(*text*, *quoting_flags=None*, *type_hints=None*, *type_hint_rules=None*)

A text loader class to extract tabular data from JSON text data.

Parameters `text` (*str*) – JSON text to load.

table_name

Table name string. Defaults to `%(key)s`.

load()

Extract tabular data as TableData instances from a JSON text object. `source` attribute should contain a text object to load.

Returns

Loaded table data iterator. Table name determined by the value of `table_name`. Following format specifiers in the `table_name` are replaced with specific strings:

Format specifier	Value after the replacement
<code>%(filename)s</code>	""
<code>%(key)s</code>	This replaced the different value for each single/multiple JSON tables: [single JSON table] <code>%(format_name)s%(format_id)s</code> [multiple JSON table] Table data key.
<code>%(format_name)s</code>	"json"
<code>%(format_id)s</code>	A unique number between the same format.
<code>%(global_id)s</code>	A unique number between all of the format.

Return type TableData iterator

See also:

JsonTableFileLoader.load()

5.2.4.3 Line-delimited Json File Loader

class `pytablereader.JsonLinesTableFileLoader`(*file_path=None, quoting_flags=None, type_hints=None, type_hint_rules=None*)

A file loader class to extract tabular data from Line-delimited JSON files.

Parameters `file_path` (*str*) – Path to the loading Line-delimited JSON file.

table_name

Table name string. Defaults to `%(filename)s_%(key)s`.

load()

Extract tabular data as TableData instances from a Line-delimited JSON file. `source` attribute should contain a path to the file to load.

Returns Loaded table data iterator. Table name determined by the value of `table_name`. Following format specifiers in the `table_name` are replaced with specific strings:

Return type TableData iterator

Raises

- `pytablereader.DataError` – If the data is invalid Line-delimited JSON.
- `pytablereader.error.ValidationError` – If the data is not acceptable Line-delimited JSON format.

5.2.4.4 Line-delimited Json Text Loader

class `pytablereader.JsonLinesTableTextLoader`(*text=None, quoting_flags=None, type_hints=None, type_hint_rules=None*)

A text loader class to extract tabular data from Line-delimited JSON text data.

Parameters `text` (*str*) – Line-delimited JSON text to load.

table_name

Table name string. Defaults to `%(key)s`.

load()

Extract tabular data as TableData instances from a Line-delimited JSON text object. `source` attribute should contain a text object to load.

Returns Loaded table data iterator. Table name determined by the value of `table_name`. Following format specifiers in the `table_name` are replaced with specific strings:

Return type TableData iterator

See also:

JsonLinesTableFileLoader.load()

5.2.5 LTSV Loader Classes

5.2.5.1 LTSV File Loader

class `pytablereader.LtsvTableFileLoader`(*file_path*, *quoting_flags=None*, *type_hints=None*,
type_hint_rules=None)

Bases: `pytablereader.ltsv.core.LtsvTableLoader`

Labeled Tab-separated Values (LTSV) format file loader class.

Parameters `file_path` (*str*) – Path to the loading LTSV file.

table_name

Table name string. Defaults to `%(filename)s`.

load()

Extract tabular data as `TableData` instances from a LTSV file. `source` attribute should contain a path to the file to load.

Returns

Loaded table data. Table name determined by the value of `table_name`. Following format specifiers in the `table_name` are replaced with specific strings:

Format specifier	Value after the replacement
<code>%(filename)s</code>	Filename (without extension)
<code>%(format_name)s</code>	"ltsv"
<code>%(format_id)s</code>	A unique number between the same format.
<code>%(global_id)s</code>	A unique number between all of the format.

Return type `TableData` iterator

Raises

- `pytablereader.InvalidHeaderNameError` – If an invalid label name is included in the LTSV file.
- `pytablereader.DataError` – If the LTSV data is invalid.

5.2.5.2 LTSV Text Loader

class `pytablereader.LtsvTableTextLoader`(*text=None*, *quoting_flags=None*, *type_hints=None*,
type_hint_rules=None)

Bases: `pytablereader.ltsv.core.LtsvTableLoader`

Labeled Tab-separated Values (LTSV) format text loader class.

Parameters `text` (*str*) – LTSV text to load.

table_name

Table name string. Defaults to `%(format_name)s%(format_id)s`.

load()

Extract tabular data as `TableData` instances from a LTSV text object. `source` attribute should contain a text object to load.

Returns

Loaded table data. Table name determined by the value of `table_name`. Following format specifiers in the `table_name` are replaced with specific strings:

Format specifier	Value after the replacement
%(filename)s	""
%(format_name)s	"ltsv"
%(format_id)s	A unique number between the same format.
%(global_id)s	A unique number between all of the format.

Return type TableData iterator

Raises

- **pytablereader.InvalidHeaderNameError** – If an invalid label name is included in the LTSV file.
- **pytablereader.DataError** – If the LTSV data is invalid.

5.2.6 Markdown Loader Classes

5.2.6.1 Markdown File Loader

class `pytablereader.MarkdownTableFileLoader`(*file_path=None, quoting_flags=None, type_hints=None, type_hint_rules=None*)

A file loader class to extract tabular data from Markdown files.

Parameters `file_path` (*str*) – Path to the loading Markdown file.

table_name

Table name string. Defaults to `%(filename)s_%(key)s`.

load()

Extract tabular data as TableData instances from a Markdown file. *source* attribute should contain a path to the file to load.

Returns

Loaded table data iterator. Table name determined by the value of `table_name`. Following format specifiers in the `table_name` are replaced with specific strings:

Format specifier	Value after the replacement
%(filename)s	Filename (without extension)
%(key)s	%(format_name)s%(format_id)s
%(format_name)s	"markdown"
%(format_id)s	A unique number between the same format.
%(global_id)s	A unique number between all of the format.

Return type TableData iterator

Raises `pytablereader.DataError` – If the Markdown data is invalid or empty.

5.2.6.2 Markdown Text Loader

class `pytablereader.MarkdownTableTextLoader`(*text=None, quoting_flags=None, type_hints=None, type_hint_rules=None*)

A text loader class to extract tabular data from Markdown text data.

Parameters `text` (*str*) – Markdown text to load.

table_name

Table name string. Defaults to `%(key)s`.

load()

Extract tabular data as `TableData` instances from a Markdown text object. `source` attribute should contain a text object to load.

Returns

Loaded table data iterator. Table name determined by the value of `table_name`. Following format specifiers in the `table_name` are replaced with specific strings:

Format specifier	Value after the replacement
<code>%(filename)s</code>	<code>""</code>
<code>%(key)s</code>	<code>%(format_name)s%(format_id)s</code>
<code>%(format_name)s</code>	<code>"markdown"</code>
<code>%(format_id)s</code>	A unique number between the same format.
<code>%(global_id)s</code>	A unique number between all of the format.

Return type `TableData` iterator

Raises `pytablereader.DataError` – If the Markdown data is invalid or empty.

5.2.7 MediaWiki Loader Classes

5.2.7.1 MediaWiki File Loader

class `pytablereader.MediaWikiTableFileLoader`(*file_path=None, quoting_flags=None, type_hints=None, type_hint_rules=None*)

A file loader class to extract tabular data from MediaWiki files.

Parameters `file_path` (*str*) – Path to the loading file.

table_name

Table name string. Defaults to `%(filename)s_%(key)s`.

load()

Extract tabular data as `TableData` instances from a MediaWiki file. `source` attribute should contain a path to the file to load.

Returns

Loaded table data iterator. Table name determined by the value of `table_name`. Following format specifiers in the `table_name` are replaced with specific strings:

Format specifier	Value after the replacement
<code>%(filename)s</code>	Filename (without extension)
<code>%(key)s</code>	This replaced to: (1) caption mark of the table (2) <code>%(format_name)s%(format_id)s</code> if caption mark not included in the table.
<code>%(format_name)s</code>	"mediawiki"
<code>%(format_id)s</code>	A unique number between the same format.
<code>%(global_id)s</code>	A unique number between all of the format.

Return type TableData iterator

Raises `pytablereader.DataError` – If the MediaWiki data is invalid or empty.

5.2.7.2 MediaWiki Text Loader

`class pytablereader.MediaWikiTableTextLoader`(*text=None, quoting_flags=None, type_hints=None, type_hint_rules=None*)

A text loader class to extract tabular data from MediaWiki text data.

Parameters `text` (*str*) – MediaWiki text to load.

table_name

Table name string. Defaults to `%(key)s`.

load()

Extract tabular data as `TableData` instances from a MediaWiki text object. `source` attribute should contain a text object to load.

Returns

Loaded table data iterator. Table name determined by the value of `table_name`. Following format specifiers in the `table_name` are replaced with specific strings:

Format specifier	Value after the replacement
<code>%(filename)s</code>	""
<code>%(key)s</code>	This replaced to: (1) caption mark of the table (2) <code>%(format_name)s%(format_id)s</code> if caption mark not included in the table.
<code>%(format_name)s</code>	"mediawiki"
<code>%(format_id)s</code>	A unique number between the same format.
<code>%(global_id)s</code>	A unique number between all of the format.

Return type TableData iterator

Raises `pytablereader.DataError` – If the MediaWiki data is invalid or empty.

5.2.8 Spread Sheet Loader Classes

5.2.8.1 Excel File Loader

class `pytablereader.ExcelTableFileLoader`(*file_path=None, quoting_flags=None, type_hints=None, type_hint_rules=None*)

A file loader class to extract tabular data from Microsoft Excel™ files.

Parameters `file_path` (*str*) – Path to the loading Excel workbook file.

table_name

Table name string. Defaults to `%(sheet)s`.

start_row

The first row to search header row.

load()

Extract tabular data as `TableData` instances from an Excel file. This method automatically search the header row of the table start from `start_row`. The header row requires all of the columns has value (except empty columns).

Returns

Loaded `TableData` iterator. `TableData` created for each sheet in the workbook. Table name determined by the value of `table_name`. Following format specifiers in the `table_name` are replaced with specific strings:

Format specifier	Value after the replacement
<code>%(filename)s</code>	Filename of the workbook
<code>%(sheet)s</code>	Name of the sheet
<code>%(format_name)s</code>	"spreadsheet"
<code>%(format_id)s</code>	A unique number between the same format.
<code>%(global_id)s</code>	A unique number between all of the format.

Return type `TableData` iterator

Raises

- `pytablereader.DataError` – If the header row is not found.
- `pytablereader.error.OpenError` – If failed to open the source file.

5.2.8.2 Google Sheets Loader

class `pytablereader.GoogleSheetsTableLoader`(*file_path=None, quoting_flags=None, type_hints=None, type_hint_rules=None*)

Concrete class of Google Spreadsheet loader.

table_name

Table name string. Defaults to `%(sheet)s`.

Parameters `file_path` (*str*) – Path to the Google Sheets credential JSON file.

Dependency Packages

- `gspread`
- `SimpleSQLite`

- `oauth2client`
- `pyOpenSSL`

Examples *Load table data from Google Sheets*

`load()`

Load table data from a Google Spreadsheet.

This method consider `source` as a path to the credential JSON file to access Google Sheets API.

The method automatically search the header row start from `start_row`. The condition of the header row is that all of the columns have value (except empty columns).

Returns Loaded table data. Return one `TableData` for each sheet in the workbook. The table name for data will be determined by `make_table_name()`.

Return type iterator of `TableData`

Raises

- `pytablereader.DataError` – If the header row is not found.
- `pytablereader.OpenError` – If the spread sheet not found.

5.2.9 Database Loader Classes

5.2.9.1 SQLite File Loader

class `pytablereader.SQLiteFileLoader`(*file_path=None, quoting_flags=None, type_hints=None, type_hint_rules=None*)

A file loader class to extract tabular data from SQLite database files.

Parameters `file_path` (*str*) – Path to the loading SQLite database file.

table_name

Table name string. Defaults to `%(filename)s_%(key)s`.

Dependency Packages

- `SimpleSQLite`

`load()`

Extract tabular data as `TableData` instances from a SQLite database file. `source` attribute should contain a path to the file to load.

Returns

Loaded table data iterator. Table name determined by the value of `table_name`. Following format specifiers in the `table_name` are replaced with specific strings:

Format specifier	Value after the replacement
<code>%(filename)s</code>	Filename (without extension)
<code>%(key)s</code>	<code>%(format_name)s%(format_id)s</code>
<code>%(format_name)s</code>	"sqlite"
<code>%(format_id)s</code>	A unique number between the same format.
<code>%(global_id)s</code>	A unique number between all of the format.

Return type `TableData` iterator

Raises `pytablereader.DataError` – If the SQLite database file data is invalid or empty.

5.3 Table Loader Factory Classes

5.3.1 File Loader Factory

`class pytablereader.factory.TableFileLoaderFactory(source, encoding=None)`

Parameters `file_path` (*str*) – Path to the loading file.

Raises `pytablereader.InvalidFilePathError` – If the `file_path` is an empty path.

`create_from_format_name(format_name)`

Create a file loader from a format name. Supported file formats are as follows:

Format name	Loader
"csv"	<i>CsvTableFileLoader</i>
"excel"	<i>ExcelTableFileLoader</i>
"html"	<i>HtmlTableFileLoader</i>
"json"	<i>JsonTableFileLoader</i>
"json"	<i>JsonTableFileLoader</i>
"json_lines"	<i>JsonTableFileLoader</i>
"jsonl"	<i>JsonLinesTableFileLoader</i>
"ltsv"	<i>LtsvTableFileLoader</i>
"markdown"	<i>MarkdownTableFileLoader</i>
"mediawiki"	<i>MediaWikiTableFileLoader</i>
"ndjson"	<i>JsonLinesTableFileLoader</i>
"sqlite"	<i>SqliteFileLoader</i>
"ssv"	<i>CsvTableFileLoader</i>
"tsv"	<i>TsvTableFileLoader</i>

Parameters `format_name` (*str*) – Format name string (case insensitive).

Returns Loader that coincides with the `format_name`:

Raises `pytablereader.LoaderNotFoundError` – If an appropriate loader not found for the format.

`create_from_path()`

Create a file loader from the file extension to loading file. Supported file extensions are as follows:

Extension	Loader
"csv"	<i>CsvTableFileLoader</i>
"xls"/"xlsx"	<i>ExcelTableFileLoader</i>
"htm"/"html"	<i>HtmlTableFileLoader</i>
"json"	<i>JsonTableFileLoader</i>
"jsonl"	<i>JsonLinesTableFileLoader</i>
"ldjson"	<i>JsonLinesTableFileLoader</i>
"ltsv"	<i>LtsvTableFileLoader</i>
"md"	<i>MarkdownTableFileLoader</i>
"ndjson"	<i>JsonLinesTableFileLoader</i>
"sqlite"/"sqlite3"	<i>SqliteFileLoader</i>
"tsv"	<i>TsvTableFileLoader</i>

Returns Loader that coincides with the file extension of the *file_extension*.

Raises *pytablereader.LoaderNotFoundError* – If an appropriate loader not found for loading the file.

property file_extension

File extension of the *source* (without period). :rtype: str

Type return

get_extension_list()

get_extensions()

Returns Available format file extensions.

Return type list

get_format_name_list()

get_format_names()

Returns Available format names.

Return type list

property source

Data source to load. :rtype: str

Type return

5.3.2 Text Loader Factory

class `pytablereader.factory.TableTextLoaderFactory`(*source*, *encoding=None*)

create_from_format_name(*format_name*)

Create a file loader from a format name. Supported file formats are as follows:

Format name	Loader
"csv"	<i>CsvTableTextLoader</i>
"html"	<i>HtmlTableTextLoader</i>
"json"	<i>JsonTableTextLoader</i>
"json_lines"	<i>JsonLinesTableTextLoader</i>
"jsonl"	<i>JsonLinesTableTextLoader</i>
"ldjson"	<i>JsonLinesTableTextLoader</i>
"ltsv"	<i>LtsvTableTextLoader</i>
"markdown"	<i>MarkdownTableTextLoader</i>
"mediawiki"	<i>MediaWikiTableTextLoader</i>
"ndjson"	<i>JsonLinesTableTextLoader</i>
"ssv"	<i>CsvTableTextLoader</i>
"tsv"	<i>TsvTableTextLoader</i>

Parameters `format_name` (*str*) – Format name string (case insensitive).

Returns Loader that coincide with the `format_name`:

Raises

- *pytablereader.LoaderNotFoundError* – If an appropriate loader not found for the format.
- *TypeError* – If `format_name` is not a string.

`create_from_path()`

`get_extension_list()`

`get_extensions()`

Returns Available format file extensions.

Return type *list*

`get_format_name_list()`

`get_format_names()`

Returns Available format names.

Return type *list*

property source

Data source to load. `:rtype:` *str*

Type *return*

5.3.3 Url Loader Factory

`class pytablereader.factory.TableUrlLoaderFactory(url, encoding=None, proxies=None)`

`create_from_format_name(format_name)`

Create a file loader from a format name. Supported file formats are as follows:

Format name	Loader
"csv"	<i>CsvTableTextLoader</i>
"excel"	<i>ExcelTableFileLoader</i>
"html"	<i>HtmlTableTextLoader</i>
"json"	<i>JsonTableTextLoader</i>
"json_lines"	<i>JsonLinesTableTextLoader</i>
"jsonl"	<i>JsonLinesTableTextLoader</i>
"ldjson"	<i>JsonLinesTableTextLoader</i>
"ltsv"	<i>LtsvTableTextLoader</i>
"markdown"	<i>MarkdownTableTextLoader</i>
"mediawiki"	<i>MediaWikiTableTextLoader</i>
"ndjson"	<i>JsonLinesTableTextLoader</i>
"sqlite"	<i>SqliteFileLoader</i>
"ssv"	<i>CsvTableTextLoader</i>
"tsv"	<i>TsvTableTextLoader</i>

Parameters `format_name` (*str*) – Format name string (case insensitive).

Returns Loader that coincide with the `format_name`:

Raises

- *pytablereader.LoaderNotFoundError* – If an appropriate loader not found for the format.
- *TypeError* – If `format_name` is not a string.

`create_from_path()`

Create a file loader from the file extension to loading file. Supported file extensions are as follows:

Extension	Loader
"csv"	<i>CsvTableTextLoader</i>
"xls"/"xlsx"	<i>ExcelTableFileLoader</i>
"htm"/"html"/"asp"/"aspx"	<i>HtmlTableTextLoader</i>
"json"	<i>JsonTableTextLoader</i>
"jsonl"/"ldjson"/"ndjson"	<i>JsonLinesTableTextLoader</i>
"ltsv"	<i>LtsvTableTextLoader</i>
"md"	<i>MarkdownTableTextLoader</i>
"sqlite"/"sqlite3"	<i>SqliteFileLoader</i>
"tsv"	<i>TsvTableTextLoader</i>

Returns Loader that coincides with the file extension of the URL.

Raises

- *pytablereader.UrlError* – If unacceptable URL format.

- *pytablereader.LoaderNotFoundError* – If an appropriate loader not found for loading the URL.

get_extension_list()

get_extensions()

Returns Available format file extensions.

Return type list

get_format_name_list()

get_format_names()

Returns Available format names.

Return type list

property source

Data source to load. :rtype: str

Type return

5.4 Exceptions

exception `pytablereader.ValidationError`

Bases: `Exception`

Exception raised when data is not properly formatted.

exception `pytablereader.PathError`

Bases: `Exception`

Base path exception class.

exception `pytablereader.InvalidFilePathError`

Bases: `pytablereader.error.PathError`

Exception raised when invalid file path used.

TODO: rename the error class

exception `pytablereader.UrlError`

Bases: `pytablereader.error.PathError`

Exception raised when invalid URL used.

exception `pytablereader.OpenError`

Bases: `OSError`

Exception raised when failed to open a file.

exception `pytablereader.LoaderNotFoundError`

Bases: `Exception`

Exception raised when loader not found.

exception `pytablereader.HTTPError(*args, **kwargs)`

Bases: `requests.exceptions.RequestException`

An HTTP error occurred.

See also:

<http://docs.python-requests.org/en/master/api/#exceptions>

exception `pytablereader.ProxyError(*args, **kwargs)`

Bases: `requests.exceptions.ProxyError`

A proxy error occurred.

See also:

http://docs.python-requests.org/en/master/_modules/requests/exceptions/

CHANGELOG

<https://github.com/thombashi/pytablereader/releases>

SPONSORS

Become a sponsor

INDICES AND TABLES

- genindex

LINKS

- [GitHub repository](#)
- [Issue tracker](#)
- [pip](#): A tool for installing Python packages

INDICES AND TABLES

- `genindex`

INDEX

- A**
- AbstractTableReader (class in pytablereader.interface), 14
- C**
- create_from_format_name() (pytablereader.factory.TableFileLoaderFactory method), 32
- create_from_format_name() (pytablereader.factory.TableTextLoaderFactory method), 33
- create_from_format_name() (pytablereader.factory.TableUrlLoaderFactory method), 35
- create_from_path() (pytablereader.factory.TableFileLoaderFactory method), 32
- create_from_path() (pytablereader.factory.TableTextLoaderFactory method), 34
- create_from_path() (pytablereader.factory.TableUrlLoaderFactory method), 35
- CsvTableFileLoader (class in pytablereader), 15
- CsvTableLoader (class in pytablereader.csv.core), 15
- CsvTableTextLoader (class in pytablereader), 16
- D**
- delimiter (pytablereader.csv.core.CsvTableLoader attribute), 15
- E**
- encoding (pytablereader.csv.core.CsvTableLoader attribute), 15
- encoding (pytablereader.HtmlTableFileLoader attribute), 16
- ExcelTableFileLoader (class in pytablereader), 30
- F**
- file_extension (pytablereader.factory.TableFileLoaderFactory property), 33
- G**
- get_extension_list() (pytablereader.factory.TableFileLoaderFactory method), 33
- get_extension_list() (pytablereader.factory.TableTextLoaderFactory method), 34
- get_extension_list() (pytablereader.factory.TableUrlLoaderFactory method), 36
- get_extensions() (pytablereader.factory.TableFileLoaderFactory method), 33
- get_extensions() (pytablereader.factory.TableTextLoaderFactory method), 34
- get_extensions() (pytablereader.factory.TableUrlLoaderFactory method), 36
- get_format_name_list() (pytablereader.factory.TableFileLoaderFactory method), 33
- get_format_name_list() (pytablereader.factory.TableTextLoaderFactory method), 34
- get_format_name_list() (pytablereader.factory.TableUrlLoaderFactory method), 36
- get_format_names() (pytablereader.factory.TableFileLoaderFactory method), 33
- get_format_names() (pytablereader.factory.TableTextLoaderFactory method), 34
- get_format_names() (pytablereader.factory.TableUrlLoaderFactory method), 36
- get_format_names() (pytablereader.TableFileLoader class method), 11
- get_format_names() (pytablereader.TableTextLoader class method), 12
- get_format_names() (pytablereader.TableUrlLoader class method), 14
- GoogleSheetsTableLoader (class in pytablereader), 30
- H**
- headers (pytablereader.csv.core.CsvTableLoader attribute), 15
- HtmlTableFileLoader (class in pytablereader), 16

HtmlTableTextLoader (class in pytablereader), 17
 HTTPError, 36

I

InvalidFilePathError, 36

J

JsonLinesTableFileLoader (class in pytablereader), 25
 JsonLinesTableTextLoader (class in pytablereader), 25
 JsonTableFileLoader (class in pytablereader), 18
 JsonTableTextLoader (class in pytablereader), 24

L

load() (pytablereader.CsvTableFileLoader method), 15
 load() (pytablereader.CsvTableTextLoader method), 16
 load() (pytablereader.ExcelTableFileLoader method), 30
 load() (pytablereader.GoogleSheetsTableLoader method), 31
 load() (pytablereader.HtmlTableFileLoader method), 16
 load() (pytablereader.HtmlTableTextLoader method), 17
 load() (pytablereader.JsonLinesTableFileLoader method), 25
 load() (pytablereader.JsonLinesTableTextLoader method), 25
 load() (pytablereader.JsonTableFileLoader method), 18
 load() (pytablereader.JsonTableTextLoader method), 24
 load() (pytablereader.LtsvTableFileLoader method), 26
 load() (pytablereader.LtsvTableTextLoader method), 26
 load() (pytablereader.MarkdownTableFileLoader method), 27
 load() (pytablereader.MarkdownTableTextLoader method), 28
 load() (pytablereader.MediaWikiTableFileLoader method), 28
 load() (pytablereader.MediaWikiTableTextLoader method), 29
 load() (pytablereader.SQLiteFileLoader method), 31
 load() (pytablereader.TableFileLoader method), 11
 load() (pytablereader.TableTextLoader method), 12
 load() (pytablereader.TableUrlLoader method), 13
 LoaderNotFoundError, 36
 LtsvTableFileLoader (class in pytablereader), 26
 LtsvTableTextLoader (class in pytablereader), 26

M

MarkdownTableFileLoader (class in pytablereader), 27
 MarkdownTableTextLoader (class in pytablereader), 28

MediaWikiTableFileLoader (class in pytablereader), 28
 MediaWikiTableTextLoader (class in pytablereader), 29

O

OpenError, 36

P

PathError, 36
 ProxyError, 37

Q

quotechar (pytablereader.csv.core.CsvTableLoader attribute), 15

S

source (pytablereader.factory.TableFileLoaderFactory property), 33
 source (pytablereader.factory.TableTextLoaderFactory property), 34
 source (pytablereader.factory.TableUrlLoaderFactory property), 36
 source (pytablereader.interface.AbstractTableReader attribute), 14
 SQLiteFileLoader (class in pytablereader), 31
 start_row (pytablereader.ExcelTableFileLoader attribute), 30

T

table_name (pytablereader.CsvTableFileLoader attribute), 15
 table_name (pytablereader.CsvTableTextLoader attribute), 16
 table_name (pytablereader.ExcelTableFileLoader attribute), 30
 table_name (pytablereader.GoogleSheetsTableLoader attribute), 30
 table_name (pytablereader.HtmlTableFileLoader attribute), 16
 table_name (pytablereader.HtmlTableTextLoader attribute), 17
 table_name (pytablereader.interface.AbstractTableReader attribute), 14
 table_name (pytablereader.JsonLinesTableFileLoader attribute), 25
 table_name (pytablereader.JsonLinesTableTextLoader attribute), 25
 table_name (pytablereader.JsonTableFileLoader attribute), 18
 table_name (pytablereader.JsonTableTextLoader attribute), 24
 table_name (pytablereader.LtsvTableFileLoader attribute), 26

`table_name` (*pytablereader.LtsvTableTextLoader* attribute), 26

`table_name` (*pytablereader.MarkdownTableFileLoader* attribute), 27

`table_name` (*pytablereader.MarkdownTableTextLoader* attribute), 28

`table_name` (*pytablereader.MediaWikiTableFileLoader* attribute), 28

`table_name` (*pytablereader.MediaWikiTableTextLoader* attribute), 29

`table_name` (*pytablereader.SQLiteFileLoader* attribute), 31

`TableFileLoader` (*class in pytablereader*), 11

`TableFileLoaderFactory` (*class in pytablereader.factory*), 32

`TableTextLoader` (*class in pytablereader*), 12

`TableTextLoaderFactory` (*class in pytablereader.factory*), 33

`TableUrlLoader` (*class in pytablereader*), 13

`TableUrlLoaderFactory` (*class in pytablereader.factory*), 35

U

`UrlError`, 36

V

`ValidationError`, 36