pytextable

Jan 30, 2022

Contents

1. Install

```
pip install pytextable
```

2. Import

```
import pytextable
```

3. Create a latex table from your data :)

```
pytextable.write(data, "table.tex")
```

CHAPTER 1

Contents

1.1 Usage

1.1.1 Installation

Using pip

```
pip install [--user] pytextable
```

Manual installation

First, you have to get the source code:

```
git clone https://github.com/karlch/pytextable
```

Then, switch to the repository:

```
cd pytextable
```

And run:

```
python setup.py install [--user]
```

Grabbing the source file

The `_pytextable.py` source file is self-contained. You can copy it,put it anywhere you like and just use it from there.

1.1.2 Examples

As examples speak louder than words, let's look at some python code and the latex output it produces. First, you will need to import the module and have some data you would like to write to a latex table:

```
import pytextable

# This is usually your 2d numpy array or any sequence of sequences
DATA = [[1.2346, 1, 1.2346], [1.2346, 1.2346, 1.2346], [1.2346, 1.2346, 1.2346]]
```

Default table

Let's start with the defaults:

```
>>> pytextable.tostring(DATA)
\begin{table}
    \centering
    \begin{tabular}{ccc}
        \toprule
        1.2346 & 1 & 1.2346 \\
        1.2346 & 1.2346 & 1.2346 \\
        1.2346 & 1.2346 & 1.2346 \\
        \bottomrule
    \end{tabular}
\end{table}
```

If you would like to write the table to a file instead, you can call write directly:

```
>>> pytextable.write(DATA)
```

Write supports all the options `tostring` does, so we will continue the examples just using `tostring`.

Formatting

You may not like the number of digits used for the formatting and that the `1` has no digits at all. To fix this, you can pass the `fmt` argument:

```
>>> pytextable.tostring(DATA, fmt=".3f")
\begin{table}
    \centering
    \begin{tabular}{ccc}
        \toprule
        1.235 & 1.000 & 1.235 \\
        1.235 & 1.235 & 1.235 \\
        1.235 & 1.235 & 1.235 \\
        \bottomrule
    \end{tabular}
\end{table}
```

The formatting is applied to every single element. If you have a more complicated use case, please pre-format each row as a sequence of strings in the format you like.

Header

Next, let's add a header to our table:

```
>>> pytextable.tostring(DATA, fmt=".3f", header=("first", "second", "third"))
\begin{table}
    \centering
    \begin{tabular}{ccc}
        \toprule
        first & second & third \\
        \midrule
        1.235 & 1.000 & 1.235 \\
        1.235 & 1.235 & 1.235 \\
        1.235 & 1.235 & 1.235 \\
        \bottomrule
    \end{tabular}
\end{table}
```

Caption and label

Pretty neat, but as good citizens we would like to add a caption and a lable to our table:

```
>>> pytextable.tostring(
    DATA,
    fmt=".3f",
    header=("first", "second", "third"),
    caption="My fancy pytextable",
    label="tab:pytextable",
)
\begin{table}
    \centering
    \caption{My fancy pytextable}
    \label{tab:pytextable}
    \begin{tabular}{ccc}
        \toprule
        first & second & third \\
        \midrule
        1.235 & 1.000 & 1.235 \\
        1.235 & 1.235 & 1.235 \\
        1.235 & 1.235 & 1.235 \\
        \bottomrule
    \end{tabular}
\end{table}
```

Table alignment

By default all elements of the table are center-aligned and no seperators are added. You can change this, by passing another alignment, for example:

```
>>> pytextable.tostring(DATA, alignment="l")
\begin{table}
    \centering
    \begin{tabular}{lll}
        \toprule
        1.2346 & 1 & 1.2346 \\
        1.2346 & 1.2346 & 1.2346 \\
        1.2346 & 1.2346 & 1.2346 \\
        \bottomrule
```

(continues on next page)

(continued from previous page)

```
    \end{tabular}
\end{table}
```

This concludes the basic usage of `pytextable`. For a list of all the options, please refer to the *the api documentation*.

1.2 Contributing Guidelines

You want to contribute to pytextable? Great! Every little help counts and is appreciated!

Need help? Feel free to contact me directly or open an issue on github.

Contents

1.2.1 Feedback and Feature Requests

Any feedback is welcome! Did you find something unintuitive? Not clearly documented? Do you have a great idea for a new feature? Let me know! You can either open an issue directly on github or contact me directly if you prefer.

You like pytextable? Share some love and spread the word!

1.2.2 Reporting Bugs

The best way to report bugs is to open an issue on github. If you do not have a github account, feel free to contact me directly. If possible, please include the traceback of the exception and instructions on how to reproduce the bug.

1.2.3 Writing Code

You probably already know what you want to work on as you are reading this page. If you want to implement a new feature, it might be a good idea to open a feature request on the issue tracker first. Otherwise you might be disappointed if I do not accept your pull request because I do not feel like this should be in the scope of pytextable.

If you want to find something to do, check the issue tracker.

If you like, you can also find some more information on the api.

1.2.4 Writing Documentation

More documentation is always useful! Here are some options where this could be done:

- Improving the website. Is something unclear or missing?

- Extending and improve the docstrings in the code base.
- Writing blog posts, articles, ... All of them are appreciated! If you like, they can also be linked here.

In case you choose to update the website, here are some more tips. The website is written in resturctured Text (reST) and built using sphinx. A great introduction is given by the reST Primer of sphinx.

You can find the reST files used to build the website in the project's `docs` folder. If you would like to build a local copy, you can run:

```
tox -e docs -- path/to/copy
```

You can then browse your local build:

```
$BROWSER path/to/copy/index.html
```

1.3 API Documentation

pytextable.**tostring**(*data: Sequence[Sequence[Any]], *, header: Sequence[str] = None, table: bool = True, centering: bool = True, caption: str = '', label: str = '', alignment: str = '', fmt: str = '', indentation: int = 4, booktabs: bool = True, midrule_condition: Callable[[Sequence[Any], Sequence[Any]], bool] = <function _no_extra_midrule>*) → str

Convert python data to a valid tex table string.

This is the heart of pytextable and does all the heavy lifting. You must pass the data argument which containts the rows and columns of the table to create. The other keyword-only arguments allow additional formatting and customization.

Parameters

- **data** – Sequence of sequences containing the rows and columns of the table. Note that the number of columns in each row must match.
- **header** – Column header to add as sequence of strings. The number of elements must match the number of columns of your data.
- **table** – Add a surrounding table environment to the latex tabular.
- **centering** – Add a centering statement to the table. This is only valid if `table=True`.
- **caption** – Add this caption to the table. This is only valid if `table=True`.
- **label** – Add this label to the table. This is only valid if `table=True`.
- **alignment** – String converted to the full table alignment. Examples:

```
>>> _table_alignment("", 3)  # The default
"ccc"
```

```
>>> _table_alignment("l", 3)  # Left-align everything instead
"lll"
```

```
>>> _table_alignment("l|", 3)  # Left-align and add separators in␣
↪the table
"l|l|l"
```

```
>>> _table_alignment("|l|", 3)  # Left-align and add separators␣
↪everywhere
"|l|l|l|"
```

```
>>> _table_alignment("llc", 3)  # Valid-formatter is just accepted
"llc"
```

```
>>> _table_alignment("|ll|l|", 3)  # Separators are fine as-well
"|ll|l|"
```

- **fmt** – Format string to apply to every element in the table data. Example: '.3f'.
- **indentation** – Number of spaces used for environment indentation.
- **booktabs** – Use the booktabs module to neatly format the table.
- **midrule_condition** – Callback to check for additional inserted midrules. This function is called with the current and previous row and should return a boolean. If it returns True, a `\midrule` is applied before the current row. Example:

```
>>> def second_elem_changed(row, last_row):
    return row[1] != last_row[1]
```

  This is useful to separate the current row from the previous one in case something changed. Only valid with `booktabs=True`.

Returns The latex table as formatted string.

`pytextable.`**write**(*data: Sequence[Sequence[Any]], outfile: str, *, writemode: str = 'w', encoding: str = 'utf-8', **kwargs*) → None

Write python data to file as formatted latex table.

Calls *tostring()* to convert the data to a valid tex table passing any additional keyword-arguments on. The retrieved string is then written to the file passed.

Parameters

- **data** – Sequence of sequences containing the rows and columns of the table. Note that the number of columns in each row must match.
- **outfile** – File to write the data to.
- **writemode** – Writemode to use when opening the file. Note that the mode must support writing, passing `r` will fail horribly for obvious reasons. This argument exists to give the option to append to a file with `a`.
- **encoding** – Encoding to use when opening the file. The default of `utf-8` works well with latex code using `\usepackage[utf8]{inputenc}`.
- **kwargs** – Keyword arguments for additional formatting passed to *tostring()*.

1.4 Changelog

All notable changes to pytextable are documented in this file.

1.4.1 v0.3.0 (unreleased)

1.4.2 v0.2.1 (2021-09-04)

Bugfix release due to erroneous tag pushed for v0.2.0.

1.4.3 v0.2.0 (2021-09-04)

Changed:

- Additional midrules from `midrule_condition` are now prepended to the current row instead of appended. This makes more sense as we can check if the current row is different from the previous one. If so, we want to separate the current row from the previous row, not the current row from the next one.
- New `encoding` keyword argument for the `write` function. The default of `utf-8` is sane and works well with latex code using `\usepackage[utf8]{inputenc}`.

1.4.4 v0.1.0 (2020-02-13)

Initial release of pytextable.

CHAPTER 2

License

This program is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program. If not, see <http://www.gnu.org/licenses/>.

Index

T

W