
py2LaTeX

Release 0.0.6

**Create LaTeX documents with Python, Markdown and
Jinja2.**

Dominic Davis-Foster

May 19, 2023

Contents

1	from PyPI	1
2	from GitHub	3
I	Documentation	5
3	py2latex	7
3.1	make_document	7
4	py2latex.colors	9
4.1	black	9
4.2	blue	9
4.3	brown	9
4.4	colour	10
4.5	cyan	10
4.6	darkgray	10
4.7	darkgrey	10
4.8	gray	10
4.9	green	11
4.10	grey	11
4.11	lime	11
4.12	magenta	11
4.13	olive	11
4.14	orange	11
4.15	pink	12
4.16	purple	12
4.17	red	12
4.18	teal	12
4.19	violet	12
4.20	white	12
4.21	yellow	13
5	py2latex.core	15
5.1	begin	15
5.2	end	15
5.3	make_caption	15
5.4	make_label	15
5.5	re_escape	15
6	py2latex.formatting	17
6.1	bold	17

6.2	<code>italic</code>	17
6.3	<code>latex_subscript</code>	17
6.4	<code>latex_superscript</code>	17
6.5	<code>underline</code>	17
7	<code>py2latex.glossaries</code>	19
8	<code>py2latex.packages</code>	21
8.1	<code>usepackage</code>	21
9	<code>py2latex.sectioning</code>	23
10	<code>py2latex.siunit</code>	25
10.1	<code>SI</code>	25
10.2	<code>si</code>	25
11	<code>py2latex.tables</code>	27
11.1	<code>SubTable</code>	27
11.2	<code>add_longtable_caption</code>	28
11.3	<code>longtable_from_template</code>	28
11.4	<code>multicolumn</code>	29
11.5	<code>parse_column_alignments</code>	30
11.6	<code>parse_hlines</code>	30
11.7	<code>parse_vspace</code>	30
11.8	<code>set_table_widths</code>	30
11.9	<code>subtables_from_template</code>	30
11.10	<code>table_from_template</code>	31
11.11	<code>tabular_from_template</code>	32
II	Contributing	35
12	Overview	37
13	Coding style	39
14	Automated tests	41
15	Type Annotations	43
16	Build documentation locally	45
17	License	47
18	Downloading source code	49
18.1	Building from source	50
	Python Module Index	51
	Index	53

from PyPI

```
$ python3 -m pip install py2latex --user
```


from GitHub

```
$ python3 -m pip install git+https://github.com/domdfcoding/py2latex@master --user
```


Part I

Documentation

py2latex

Create LaTeX documents with Python, Markdown and Jinja2.

Functions:

<code>make_document(outfile, *elements[, glossary])</code>	Construct a LaTeX document from the given elements.
--	---

make_document (*outfile*, **elements*, *glossary=""*)
Construct a LaTeX document from the given elements.

Parameters

- **outfile** (`Union[str, Path, PathLike]`)
- ***elements** (`Iterable[str]`)
- **glossary** (`str`) – Default `''`.

py2latex.colors

Functions:

<i>black</i> (text)	Make the given text black.
<i>blue</i> (text)	Make the given text blue.
<i>brown</i> (text)	Make the given text brown.
<i>colour</i> (text_colour, text)	Make the given text the given colour.
<i>cyan</i> (text)	Make the given text cyan.
<i>darkgray</i> (text)	Make the given text darkgray.
<i>darkgrey</i> (text)	Make the given text darkgrey.
<i>gray</i> (text)	Make the given text gray.
<i>green</i> (text)	Make the given text green.
<i>grey</i> (text)	Make the given text grey.
<i>lime</i> (text)	Make the given text lime.
<i>magenta</i> (text)	Make the given text magenta.
<i>olive</i> (text)	Make the given text olive.
<i>orange</i> (text)	Make the given text orange.
<i>pink</i> (text)	Make the given text pink.
<i>purple</i> (text)	Make the given text purple.
<i>red</i> (text)	Make the given text red.
<i>teal</i> (text)	Make the given text teal.
<i>violet</i> (text)	Make the given text violet.
<i>white</i> (text)	Make the given text white.
<i>yellow</i> (text)	Make the given text yellow.

black (text)

Make the given text black.

Akin to `\color{black}{string}`.

Parameters **text** (str)

Return type str

Returns The formatted string.

blue (text)

Make the given text blue.

Akin to `\color{blue}{string}`.

Parameters **text** (str)

Return type str

Returns The formatted string.

brown (*text*)

Make the given text brown.

Akin to `\color{brown}{string}`.

Parameters `text` (*str*)

Return type *str*

Returns The formatted string.

colour (*text_colour*, *text*)

Make the given text the given colour.

Akin to `\color{text_colour}{string}`.

Parameters

- `text_colour` (*str*) – The colour to make the text
- `text` (*str*) – The text to colour

Return type *str*

Returns The formatted string.

cyan (*text*)

Make the given text cyan.

Akin to `\color{cyan}{string}`.

Parameters `text` (*str*)

Return type *str*

Returns The formatted string.

darkgray (*text*)

Make the given text darkgray.

Akin to `\\color{darkgray}{string}`.

Parameters `text` (*str*)

Return type *str*

Returns The formatted string.

darkgrey (*text*)

Make the given text darkgrey.

Akin to `\color{darkgray}{string}`.

Parameters `text` (*str*)

Return type *str*

Returns The formatted string.

gray (*text*)

Make the given text gray.

Akin to `\color{gray}{string}`.

Parameters `text` (`str`)

Return type `str`

Returns The formatted string.

green (`text`)

Make the given text green.

Akin to `\color{green}{string}`.

Parameters `text` (`str`)

Return type `str`

Returns The formatted string.

grey (`text`)

Make the given text grey.

Akin to `\color{gray}{string}`.

Parameters `text` (`str`)

Return type `str`

Returns The formatted string.

lime (`text`)

Make the given text lime.

Akin to `\color{lime}{string}`.

Parameters `text` (`str`)

Return type `str`

Returns The formatted string.

magenta (`text`)

Make the given text magenta.

Akin to `\color{magenta}{string}`.

Parameters `text` (`str`)

Return type `str`

Returns The formatted string.

olive (`text`)

Make the given text olive.

Akin to `\color{olive}{string}`.

Parameters `text` (`str`)

Return type `str`

Returns The formatted string.

orange (`text`)

Make the given text orange.

Akin to `\color{orange}{string}`.

Parameters `text` (`str`)

Return type `str`

Returns The formatted string.

pink (`text`)

Make the given text pink.

Akin to `\color{pink}{string}`.

Parameters `text` (`str`)

Return type `str`

Returns The formatted string.

purple (`text`)

Make the given text purple.

Akin to `\color{purple}{string}`.

Parameters `text` (`str`)

Return type `str`

Returns The formatted string.

red (`text`)

Make the given text red.

Akin to `\color{red}{string}`.

Parameters `text` (`str`)

Return type `str`

Returns The formatted string.

teal (`text`)

Make the given text teal.

Akin to `\color{teal}{string}`.

Parameters `text` (`str`)

Return type `str`

Returns The formatted string.

violet (`text`)

Make the given text violet.

Akin to `\color{violet}{string}`.

Parameters `text` (`str`)

Return type `str`

Returns The formatted string.

white (*text*)

Make the given text white.

Akin to `\color{white}{string}`.

Parameters `text` (`str`)

Return type `str`

Returns The formatted string.

yellow (*text*)

Make the given text yellow.

Akin to `\color{yellow}{string}`.

Parameters `text` (`str`)

Return type `str`

Returns The formatted string.

Core functionality.

Functions:

<code>begin(environment[, options])</code>	Akin to <code>\begin{environment}</code> .
<code>end(environment)</code>	Akin to <code>\end{environment}</code> .
<code>make_caption(caption)</code>	Akin to <code>\caption{}</code> .
<code>make_label(label)</code>	Akin to <code>\label{}</code> .
<code>re_escape(string)</code>	Escape literal backslashes for use with <code>re</code> .

begin (*environment*, *options=None*)

Akin to `\begin{environment}`.

Parameters

- **environment** (*str*)
- **options** (*Optional[str]*) – Default `None`.

Return type *str*

end (*environment*)

Akin to `\end{environment}`.

Parameters **environment** (*str*)

Return type *str*

make_caption (*caption*)

Akin to `\caption{}`.

Parameters **caption** (*str*)

Return type *str*

make_label (*label*)

Akin to `\label{}`.

Parameters **label** (*str*)

Return type *str*

re_escape (*string*)

Escape literal backslashes for use with `re`.

See also:

`re.escape()`, which escapes all characters treated specially by `re`.

Parameters `string (str)`

Return type `str`

`py2latex.formatting`

Functions:

<code>bold(val)</code>	Make the given value bold.
<code>italic(val)</code>	Make the given value italic.
<code>latex_subscript(val)</code>	Returns the LaTeX subscript of the given value.
<code>latex_superscript(val)</code>	Returns the LaTeX superscript of the given value.
<code>underline(val)</code>	Underline the given value.

`bold(val)`

Make the given value bold.

Akin to `textbf{string}`

Parameters `val` (`Union[str, float]`)

Return type `str`

Returns The formatted string.

`italic(val)`

Make the given value italic.

Akin to `textit{string}`

Parameters `val` (`Union[str, float]`)

Return type `str`

Returns The formatted string.

`latex_subscript(val)`

Returns the LaTeX subscript of the given value.

Parameters `val` (`Union[str, float]`) – The value to superscript.

Return type `str`

`latex_superscript(val)`

Returns the LaTeX superscript of the given value.

Parameters `val` (`Union[str, float]`) – The value to subscript.

Return type `str`

`underline(val)`

Underline the given value.

Akin to `underline{string}`

Parameters `val` (`Union[str, float]`)

Return type `str`

Returns The formatted string.

`py2latex.glossaries`

py2latex.packages

Functions:

<i>usepackage</i> (package_name[, options])	Akin to <code>\usepackage[options]{package_name}</code> .
---	---

usepackage (*package_name*, *options=None*)

Akin to `\usepackage[options]{package_name}`.

Parameters

- **package_name** (*str*) – The name of the package
- **options** (*Optional[str]*) – Options for the package. Default *None*.

Return type *str*

`py2latex.sectioning`

py2latex.siunit

Functions:

<code>SI(quantity[, per_mode])</code>	Create an siunitx-formatted formula from an <code>astropy.units</code> unit.
<code>si(unit[, per_mode])</code>	Create an siunitx-formatted formula from an <code>astropy.units</code> unit.

SI (*quantity*, *per_mode*='symbol')

Create an siunitx-formatted formula from an `astropy.units` unit.

Parameters

- **quantity** (`Quantity`)
- **per_mode** (`Literal`['repeated-symbol', 'symbol', 'fraction', 'symbol-or-fraction', 'reciprocal']) – Default 'symbol'.

Returns

Return type `str`

si (*unit*, *per_mode*='symbol')

Create an siunitx-formatted formula from an `astropy.units` unit.

Parameters

- **unit** (`UnitBase`)
- **per_mode** (`Literal`['repeated-symbol', 'symbol', 'fraction', 'symbol-or-fraction', 'reciprocal']) – Default 'symbol'.

Returns

Return type `str`

py2latex.tables

Classes:

SubTable(tabular_data, *, caption[, label, ...])

type tabular_data `Sequence[Sequence[Any]]`

Functions:

add_longtable_caption(table[, caption, label]) Add a caption to a longtable.

longtable_from_template(tabular_data, *, caption) Create a longtable with booktabs formatting.

multicolumn(cols, pos, text)

type cols `int`

parse_column_alignments(colalign, colwidths, ...)

type colalign
`Optional[Sequence[Optional[str]]]`

parse_hlines(nrows[, hlines])

type nrows `int`

parse_vspace(ncols[, vspace])

type ncols `int`

set_table_widths(table, widths) Override the column widths (and also the column alignments) in a tabular environment, etc.

subtables_from_template(subtables, *, caption) Create a series of subtables with booktabs formatting.

table_from_template(tabular_data, *, caption) Create a table with booktabs formatting.

tabular_from_template(tabular_data, *[, ...]) Create a tabular environment with booktabs formatting.

class **SubTable** (*tabular_data*, *, *caption*, *label*=None, *headers*=(), *floatfmt*='g', *numalign*='decimal',
 stralign='left', *missingval*="", *showindex*='default', *disable_numparse*=False,
 colalign=None, *colwidths*=None, *vlines*=False, *hlines*=False, *vspace*=False, *raw*=True,
 footer=None)

Bases: `object`

Parameters

- **tabular_data** (`Sequence[Sequence[Any]]`)
- **caption** (`str`) – The caption for the table
- **label** (`Optional[str]`) – The label for the table. If undefined the caption is used, in lowercase, with underscores replacing spaces Default `None`.
- **headers** (`Sequence[str]`) – A sequence of column headers. Default `()`.
- **floatfmt** (`Union[str, Iterable[str]]`) – The formatting of `float` values. Default `"g"`. Default `'g'`.
- **numalign** (`Optional[str]`) – Default `'decimal'`.
- **stralign** (`Optional[str]`) – Default `'left'`.
- **missingval** (`Union[str, Iterable[str]]`) – Default `' '`.
- **showindex** (`Union[str, bool, Iterable[Any]]`) – Default `'default'`.
- **disable_numparse** (`Union[bool, Iterable[int]]`) – Default `False`.
- **colalign** (`Optional[Sequence[Optional[str]]]`) – Default `None`.
- **colwidths** (`Optional[Sequence[Optional[str]]]`) – Sequence of column widths, e.g. `3cm`. Values of `None` indicates auto width. Default `None`.
- **vlines** (`Union[Sequence[int], bool]`) – If a sequence of integers a line will be inserted before the specified columns. `-1` indicates a line should be inserted after the last column. If `True` a line will be inserted before every column, and after the last column. If `False` no lines will be inserted. Default `False`.
- **hlines** (`Union[Sequence[int], bool]`) – If a sequence of integers a line will be inserted before the specified rows. `-1` indicates a line should be inserted after the last row. If `True` a line will be inserted before every row, and after the last row. If `False` no lines will be inserted. Default `False`.
- **vspace** (`Union[Sequence[int], bool]`) – If a sequence of integers extra space will be inserted before the specified row. `-1` indicates a space should be inserted after the last row. If `True` a space will be inserted before every row, and after the last row. If `False` no spaces will be inserted. Default `False`.
- **raw** (`bool`) – Whether latex markup in `tabular_data` should be unescaped. Default `False`. Default `True`.
- **footer** (`Optional[str]`) – Optional footer for the table. Inserted as raw LaTeX. Default `None`.

add_longtable_caption (`table`, `caption=None`, `label=None`)

Add a caption to a longtable.

Parameters

- **table** (`str`)
- **caption** (`Optional[str]`) – str. Default `None`.
- **label** (`str`) – Default `None`.

Returns

Return type `str`

longtable_from_template (*tabular_data*, *, *caption*, *label*=None, *headers*=(), *pos*='htpb', *floatfmt*='g', *numalign*='decimal', *stralign*='left', *missingval*="", *showindex*='default', *disable_numparse*=False, *colalign*=None, *colwidths*=None, *vlines*=False, *hlines*=False, *vspace*=False, *raw*=True, *footer*=None)

Create a longtable with booktabs formatting.

Parameters

- **tabular_data** (Sequence[Sequence[Any]])
- **caption** (*str*) – The caption for the table
- **label** (Optional[*str*]) – The label for the table. If undefined the caption is used, in lowercase, with underscores replacing spaces Default None.
- **headers** (Sequence[*str*]) – A sequence of column headers. Default ().
- **pos** (*str*) – The positioning of the table, e.g. "htpb". Default 'htpb'.
- **floatfmt** (Union[*str*, Iterable[*str*]]) – The formatting of float values. Default "g". Default 'g'.
- **numalign** (Optional[*str*]) – Default 'decimal'.
- **stralign** (Optional[*str*]) – Default 'left'.
- **missingval** (Union[*str*, Iterable[*str*]]) – Default ''.
- **showindex** (Union[*str*, bool, Iterable[Any]]) – Default 'default'.
- **disable_numparse** (Union[bool, Iterable[int]]) – Default False.
- **colalign** (Optional[Sequence[Optional[*str*]]]) – Default None.
- **colwidths** (Optional[Sequence[Optional[*str*]]]) – Sequence of column widths, e.g. 3cm. Values of None indicates auto width. Default None.
- **vlines** (Union[Sequence[int], bool]) – If a sequence of integers a line will be inserted before the specified columns. -1 indicates a line should be inserted after the last column. If True a line will be inserted before every column, and after the last column. If False no lines will be inserted. Default False.
- **hlines** (Union[Sequence[int], bool]) – If a sequence of integers a line will be inserted before the specified rows. -1 indicates a line should be inserted after the last row. If True a line will be inserted before every row, and after the last row. If False no lines will be inserted. Default False.
- **vspace** (Union[Sequence[int], bool]) – If a sequence of integers extra space will be inserted before the specified row. -1 indicates a space should be inserted after the last row. If True a space will be inserted before every row, and after the last row. If False no spaces will be inserted. Default False.
- **raw** (*bool*) – Whether latex markup in tabular_data should be unescaped. Default False. Default True.
- **footer** (Optional[*str*]) – Optional footer for the table. Inserted as raw LaTeX. Default None.

Returns

Return type *str*

multicolumn (*cols*, *pos*, *text*)

Parameters

- **cols** (*int*) – The number of columns to span
- **pos** (*str*) – Text alignment: * c for centered * l for flushleft * r for flushright
- **text** (*str*)

Return type *str*

parse_column_alignments (*colalign, colwidths, vlines, ncols*)

Parameters

- **colalign** (*Optional[Sequence[Optional[str]]]*)
- **colwidths** (*Optional[Sequence[Optional[str]]]*)
- **vlines** (*Union[Sequence[int], bool]*)
- **ncols** (*int*)

Return type *str*

parse_hlines (*nrows, hlines=False*)

Parameters

- **nrows** (*int*)
- **hlines** (*Union[Sequence[int], bool]*) – Default *False*.

Return type *Tuple[bool, Sequence[int]]*

parse_vspace (*ncols, vspace=False*)

Parameters

- **ncols** (*int*)
- **vspace** (*Union[Sequence[int], bool]*) – Default *False*.

Return type *Tuple[bool, Sequence[int]]*

set_table_widths (*table, widths*)

Override the column widths (and also the column alignments) in a tabular environment, etc.

Parameters

- **table** (*str*)
- **widths** (*str*)

Returns

Return type *str*

subtables_from_template (*subtables, *, caption, label=None, pos='htpb'*)

Create a series of subtables with booktabs formatting.

Parameters

- **subtables** (`Iterable[SubTable]`)
- **caption** (`str`) – The caption for the table
- **label** (`Optional[str]`) – The label for the table. If undefined the caption is used, in lowercase, with underscores replacing spaces Default `None`.
- **pos** (`str`) – The positioning of the table, e.g. "htp". Default 'htpb'.

Returns

Return type `str`

table_from_template (`tabular_data`, *, `caption`, `label=None`, `headers=()`, `pos='htpb'`, `floatfmt='g'`, `numalign='decimal'`, `stralign='left'`, `missingval=""`, `showindex='default'`, `disable_numparse=False`, `colalign=None`, `colwidths=None`, `vlines=False`, `hlines=False`, `vspace=False`, `raw=True`, `footer=None`)

Create a table with booktabs formatting.

Parameters

- **tabular_data** (`Sequence[Sequence[Any]]`)
- **caption** (`str`) – The caption for the table
- **label** (`Optional[str]`) – The label for the table. If undefined the caption is used, in lowercase, with underscores replacing spaces Default `None`.
- **headers** (`Sequence[str]`) – A sequence of column headers. Default `()`.
- **pos** (`str`) – The positioning of the table, e.g. "htp". Default 'htpb'.
- **floatfmt** (`Union[str, Iterable[str]]`) – The formatting of `float` values. Default "g". Default 'g'.
- **numalign** (`Optional[str]`) – Default 'decimal'.
- **stralign** (`Optional[str]`) – Default 'left'.
- **missingval** (`Union[str, Iterable[str]]`) – Default ''.
- **showindex** (`Union[str, bool, Iterable[Any]]`) – Default 'default'.
- **disable_numparse** (`Union[bool, Iterable[int]]`) – Default `False`.
- **colalign** (`Optional[Sequence[Optional[str]]]`) – Default `None`.
- **colwidths** (`Optional[Sequence[Optional[str]]]`) – Sequence of column widths, e.g. 3cm. Values of `None` indicates auto width. Default `None`.
- **vlines** (`Union[Sequence[int], bool]`) – If a sequence of integers a line will be inserted before the specified columns. -1 indicates a line should be inserted after the last column. If `True` a line will be inserted before every column, and after the last column. If `False` no lines will be inserted. Default `False`.
- **hlines** (`Union[Sequence[int], bool]`) – If a sequence of integers a line will be inserted before the specified rows. -1 indicates a line should be inserted after the last row. If `True` a line will be inserted before every row, and after the last row. If `False` no lines will be inserted. Default `False`.
- **vspace** (`Union[Sequence[int], bool]`) – If a sequence of integers extra space will be inserted before the specified row. -1 indicates a space should be inserted after the last row. If `True` a space will be inserted before every row, and after the last row. If `False` no spaces will be inserted. Default `False`.

- **raw** (*bool*) – Whether latex markup in `tabular_data` should be unescaped. Default `False`. Default `True`.
- **footer** (*Optional[str]*) – Optional footer for the table. Inserted as raw LaTeX. Default `None`.

Returns

Return type `str`

tabular_from_template (*tabular_data*, *, *headers=()*, *floatfmt='g'*, *numalign='decimal'*, *stralign='left'*, *missingval=""*, *showindex='default'*, *disable_numparse=False*, *colalign=None*, *colwidths=None*, *vlines=False*, *hlines=False*, *vspace=False*, *raw=True*, *footer=None*, *no_lines=False*, *left_margin=True*, *right_margin=True*)

Create a `tabular` environment with `booktabs` formatting.

Parameters

- **tabular_data** (*Sequence[Sequence[Any]]*)
- **headers** (*Sequence[str]*) – A sequence of column headers. Default `()`.
- **floatfmt** (*Union[str, Iterable[str]]*) – The formatting of `float` values. Default `"g"`. Default `'g'`.
- **numalign** (*Optional[str]*) – Default `'decimal'`.
- **stralign** (*Optional[str]*) – Default `'left'`.
- **missingval** (*Union[str, Iterable[str]]*) – Default `' '`.
- **showindex** (*Union[str, bool, Iterable[Any]]*) – Default `'default'`.
- **disable_numparse** (*Union[bool, Iterable[int]]*) – Default `False`.
- **colalign** (*Optional[Sequence[Optional[str]]]*) – Default `None`.
- **colwidths** (*Optional[Sequence[Optional[str]]]*) – Sequence of column widths, e.g. `3cm`. Values of `None` indicates auto width. Default `None`.
- **vlines** (*Union[Sequence[int], bool]*) – If a sequence of integers a line will be inserted before the specified columns. `-1` indicates a line should be inserted after the last column. If `True` a line will be inserted before every column, and after the last column. If `False` no lines will be inserted. Default `False`.
- **hlines** (*Union[Sequence[int], bool]*) – If a sequence of integers a line will be inserted before the specified rows. `-1` indicates a line should be inserted after the last row. If `True` a line will be inserted before every row, and after the last row. If `False` no lines will be inserted. Default `False`.
- **vspace** (*Union[Sequence[int], bool]*) – If a sequence of integers extra space will be inserted before the specified row. `-1` indicates a space should be inserted after the last row. If `True` a space will be inserted before every row, and after the last row. If `False` no spaces will be inserted. Default `False`.
- **raw** (*bool*) – Whether latex markup in `tabular_data` should be unescaped. Default `False`. Default `True`.
- **footer** (*Optional[str]*) – Optional footer for the table. Inserted as raw LaTeX. Default `None`.
- **no_lines** (*bool*) – Whether to suppress horizontal lines in the table. Default `False`. Default `False`.

- **left_margin** (`bool`) – Whether to include a margin to the left of the table. Default `True`. Default `True`.
- **right_margin** (`bool`) – Whether to include a margin to the right of the table. Default `True`. Default `True`.

Returns**Return type** `str`

Part II

Contributing

Overview

py2LaTeX uses `tox` to automate testing and packaging, and `pre-commit` to maintain code quality.

Install `pre-commit` with `pip` and install the git hook:

```
$ python -m pip install pre-commit
$ pre-commit install
```


Coding style

`formate` is used for code formatting.

It can be run manually via `pre-commit`:

```
$ pre-commit run formate -a
```

Or, to run the complete autoformatting suite:

```
$ pre-commit run -a
```


Automated tests

Tests are run with `tox` and `pytest`. To run tests for a specific Python version, such as Python 3.6:

```
$ tox -e py36
```

To run tests for all Python versions, simply run:

```
$ tox
```


Type Annotations

Type annotations are checked using `mypy`. Run `mypy` using `tox`:

```
$ tox -e mypy
```


Build documentation locally

The documentation is powered by Sphinx. A local copy of the documentation can be built with `tox`:

```
$ tox -e docs
```


License

py2LaTeX is licensed under the [MIT License](#)

A short and simple permissive license with conditions only requiring preservation of copyright and license notices. Licensed works, modifications, and larger works may be distributed under different terms and without source code.

Permissions

- Commercial use – The licensed material and derivatives may be used for commercial purposes.
- Modification – The licensed material may be modified.
- Distribution – The licensed material may be distributed.
- Private use – The licensed material may be used and modified in private.

Conditions

- License and copyright notice – A copy of the license and copyright notice must be included with the licensed material.

Limitations

- Liability – This license includes a limitation of liability.
- Warranty – This license explicitly states that it does NOT provide any warranty.

[See more information on choosealicense.com](#) ⇒

```
Copyright (c) 2020 Dominic Davis-Foster
```

```
Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:
```

```
The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.
```

```
THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM,
DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR
OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE
OR OTHER DEALINGS IN THE SOFTWARE.
```


Downloading source code

The py2LaTeX source code is available on GitHub, and can be accessed from the following URL: <https://github.com/domdfcoding/py2latex>

If you have git installed, you can clone the repository with the following command:

```
$ git clone https://github.com/domdfcoding/py2latex
```

```
Cloning into 'py2latex'...
remote: Enumerating objects: 47, done.
remote: Counting objects: 100% (47/47), done.
remote: Compressing objects: 100% (41/41), done.
remote: Total 173 (delta 16), reused 17 (delta 6), pack-reused 126
Receiving objects: 100% (173/173), 126.56 KiB | 678.00 KiB/s, done.
Resolving deltas: 100% (66/66), done.
```

Alternatively, the code can be downloaded in a ‘zip’ file by clicking:

Clone or download → Download Zip

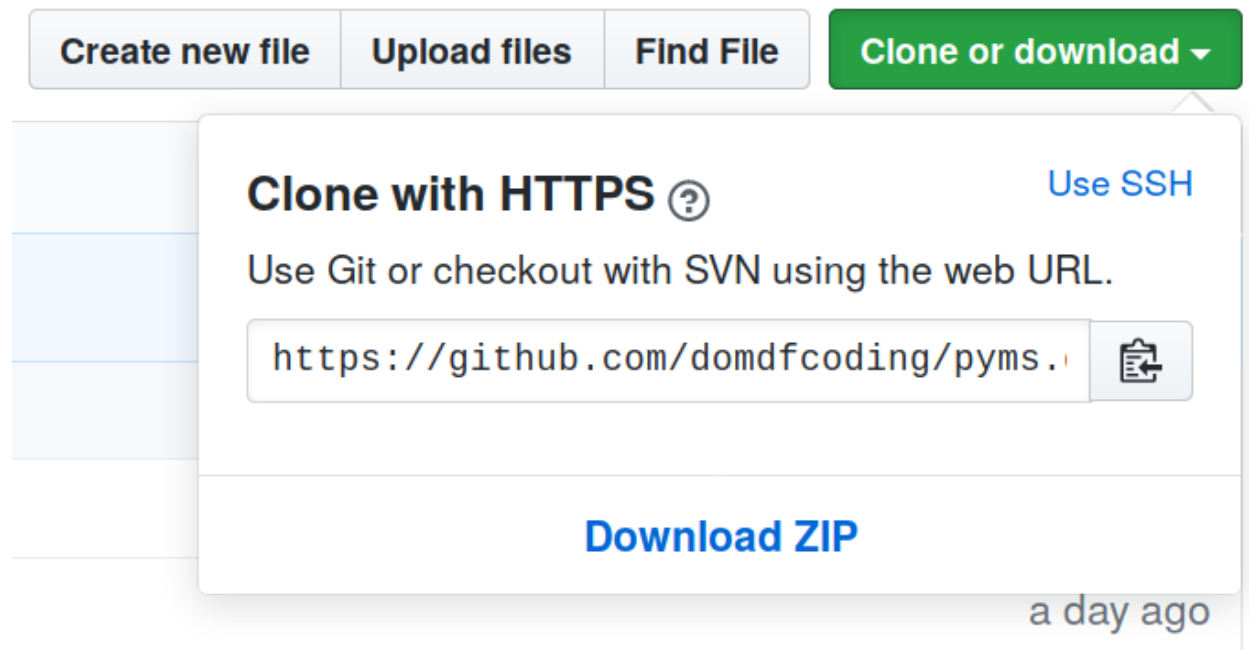


Fig. 1: Downloading a ‘zip’ file of the source code

18.1 Building from source

The recommended way to build py2LaTeX is to use `tox`:

```
$ tox -e build
```

The source and wheel distributions will be in the directory `dist`.

If you wish, you may also use `pep517.build` or another **PEP 517**-compatible build tool.

Python Module Index

p

- `py2latex.__init__`, [7](#)
- `py2latex.colors`, [9](#)
- `py2latex.core`, [15](#)
- `py2latex.formatting`, [17](#)
- `py2latex.glossaries`, [19](#)
- `py2latex.packages`, [21](#)
- `py2latex.sectioning`, [23](#)
- `py2latex.siunit`, [25](#)
- `py2latex.tables`, [27](#)

A

`add_longtable_caption()` (in module `py2latex.tables`), 28

B

`begin()` (in module `py2latex.core`), 15
`black()` (in module `py2latex.colors`), 9
`blue()` (in module `py2latex.colors`), 9
`bold()` (in module `py2latex.formatting`), 17
`brown()` (in module `py2latex.colors`), 9

C

`colour()` (in module `py2latex.colors`), 10
`cyan()` (in module `py2latex.colors`), 10

D

`darkgray()` (in module `py2latex.colors`), 10
`darkgrey()` (in module `py2latex.colors`), 10

E

`end()` (in module `py2latex.core`), 15

G

`gray()` (in module `py2latex.colors`), 10
`green()` (in module `py2latex.colors`), 11
`grey()` (in module `py2latex.colors`), 11

I

`italic()` (in module `py2latex.formatting`), 17

L

`latex_subscript()` (in module `py2latex.formatting`), 17
`latex_superscript()` (in module `py2latex.formatting`), 17
`lime()` (in module `py2latex.colors`), 11
`longtable_from_template()` (in module `py2latex.tables`), 28

M

`magenta()` (in module `py2latex.colors`), 11
`make_caption()` (in module `py2latex.core`), 15

`make_document()` (in module `py2latex.__init__`), 7
`make_label()` (in module `py2latex.core`), 15
MIT License, 47

module

`py2latex.__init__`, 7
`py2latex.colors`, 9
`py2latex.core`, 15
`py2latex.formatting`, 17
`py2latex.glossaries`, 19
`py2latex.packages`, 21
`py2latex.sectioning`, 23
`py2latex.siunit`, 25
`py2latex.tables`, 27

`multicolumn()` (in module `py2latex.tables`), 29

O

`olive()` (in module `py2latex.colors`), 11
`orange()` (in module `py2latex.colors`), 11

P

`parse_column_alignments()` (in module `py2latex.tables`), 30
`parse_hlines()` (in module `py2latex.tables`), 30
`parse_vspace()` (in module `py2latex.tables`), 30
`pink()` (in module `py2latex.colors`), 12
`purple()` (in module `py2latex.colors`), 12
`py2latex.__init__`
 module, 7
`py2latex.colors`
 module, 9
`py2latex.core`
 module, 15
`py2latex.formatting`
 module, 17
`py2latex.glossaries`
 module, 19
`py2latex.packages`
 module, 21
`py2latex.sectioning`
 module, 23
`py2latex.siunit`
 module, 25
`py2latex.tables`
 module, 27

Python Enhancement Proposals
PEP 517, [50](#)

R

`re_escape()` (in module *py2latex.core*), [15](#)
`red()` (in module *py2latex.colors*), [12](#)

S

`set_table_widths()` (in module *py2latex.tables*),
[30](#)
`SI()` (in module *py2latex.siunit*), [25](#)
`si()` (in module *py2latex.siunit*), [25](#)
`SubTable` (class in *py2latex.tables*), [27](#)
`subtables_from_template()` (in module
py2latex.tables), [30](#)

T

`table_from_template()` (in module
py2latex.tables), [31](#)
`tabular_from_template()` (in module
py2latex.tables), [32](#)
`teal()` (in module *py2latex.colors*), [12](#)

U

`underline()` (in module *py2latex.formatting*), [17](#)
`usepackage()` (in module *py2latex.packages*), [21](#)

V

`violet()` (in module *py2latex.colors*), [12](#)

W

`white()` (in module *py2latex.colors*), [12](#)

Y

`yellow()` (in module *py2latex.colors*), [13](#)