py2LaTeX

Release 0.0.6

Create LaTeX documents with Python, Markdown and Jinja2.

Dominic Davis-Foster

May 19, 2023

Contents

Chapter

ONE

from PyPI

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

Chapter

TWO

from GitHub

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

Part I

Documentation

Chapter

THREE

py2latex

Create LaTeX documents with Python, Markdown and Jinja2.

Functions:

make_document(outfile, *elements[, glossary])	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 ''.

Chapter

FOUR

py2latex.colors

Functions:

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

Chapter

FIVE

py2latex.core

Core functionality.

Functions:

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

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 be `re`.

Parameters **string** (str)

Return type str

Chapter

SIX

py2latex.formatting

Functions:

bold(val)	Make the given value bold.
italic(val)	Make the given value italic.
latex_subscript(val)	Returns the LaTeX subscript of the given value.
latex_superscript(val)	Returns the LaTeX superscript of the given value.
underline(val)	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.

Chapter

SEVEN

py2latex.glossaries

Chapter

EIGHT

py2latex.packages

Functions:

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

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`

Chapter

NINE

py2latex.sectioning

Chapter

TEN

py2latex.siunit

Functions:

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

Chapter

ELEVEN

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. `"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

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

Parameters

- **cols** (int) – The number of columms 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

Chapter

TWELVE

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
```

Chapter

THIRTEEN

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
```

Chapter

FOURTEEN

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
```

Chapter

FIFTEEN

Type Annotations

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

```
$ tox -e mypy
```

Chapter

SIXTEEN

Build documentation locally

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

```
$ tox -e docs
```

Chapter

SEVENTEEN

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 ⇒

Chapter

EIGHTEEN

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

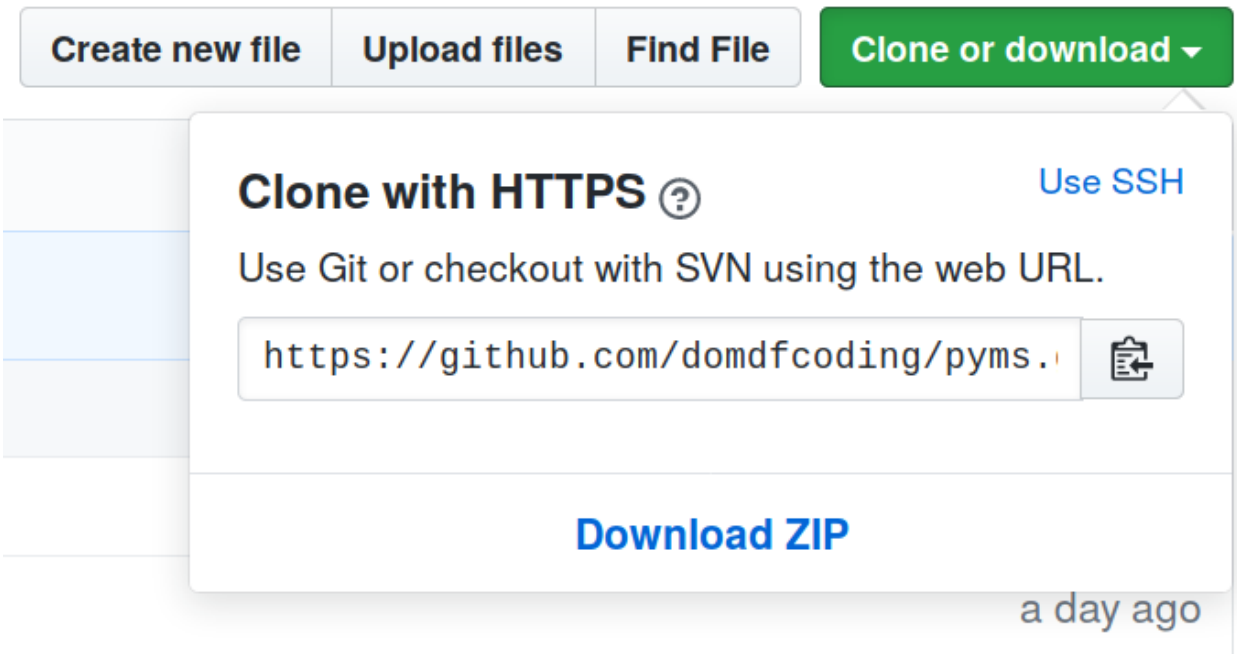

Fig. 1: Downloading a 'zip' file of the source code

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

Index

R

S

T

U

V

W

Y