nbpreview

Paulo S. Costa

Sep 26, 2022

CONTENTS

A terminal viewer for Jupyter notebooks. It's like cat for ipynb files.

nbpreview can be installed through pipx or pip from PyPI.

pipx provides an easy way to install Python applications in isolated environments. See the documentation for how to install pipx.

```
% pipx install nbpreview
```

If pipx is not installed, nbpreview may also be installed via pip:

```
% python -m pip install nbpreview
```

CHAPTER ONE

USAGE

nbpreview has only one required argument—*FILE*—which expects a Jupyter notebook (`.ipynb`) file path. *FILE* is a flexible argument. It can take:

- A Jupyter notebook (`ipynb`) file path
- Multiple notebook paths
- Take in input from stdin

For more details, see features.

nbpreview also comes with a convenient alias—`nbp`. Invoke either `nbpreview`

```
% nbpreview notebook.ipynb
```

or `nbp`

```
% nbp notebook.ipynb
```

on the command-line to run the program.

`--help`

To read the documentation on all options, their effects, values, and environmental variables, run

```
% nbpreview --help
```

1.1 nbpreview

Render a Jupyter Notebook in the terminal.

```
nbpreview [OPTIONS] [FILE]...
```

Options

`-t, --theme <theme>`

The theme to use for syntax highlighting. Call `--list-themes` to preview all available themes.

Default
dark

Options
default | emacs | friendly | friendly_grayscale | colorful | autumn | murphy | manni | material | monokai | perldoc | pastie | borland | trac | native | fruity | bw | vim | vs | tango | rrt | xcode | igor | paraiso-light | paraiso-dark | lovelace | algol | algol_nu | arduino | rainbow_dash | abap | solarized-dark | solarized-light | sas | staroffice | stata | stata-light | stata-dark | inkpot | zenburn | gruvbox-dark | gruvbox-light | dracula | one-dark | lilypond | nord | nord-darker | github-dark | light | dark | ansi_light | ansi_dark

`--list-themes, --lt`

Display a preview of all available themes.

`-p, --plain, -d, --decorated`

Whether to render in a plain style with no boxes, execution counts, or spacing. By default detected depending on usage context.

`-u, --unicode, -x, --no-unicode`

Force the display or replacement of Unicode characters instead of determining automatically.

`-h, --hide-output`

Whether to hide the notebook outputs.

Default
False

`-n, --nerd-font`

Whether to use Nerd Font icons.

Default
False

`-l, --no-files`

Do not write temporary files for previews.

Default
False

`-s, --positive-space`

Draw character images in positive space. Generally, negative space works best on charts or images with light backgrounds, while positive space will look best on dark background images. Only affects character drawings. By default set to negative space.

Default
False

`-k, --hyperlinks, -r, --no-hyperlinks`

Whether to use terminal hyperlinks when rendering content. By default autodetects.

`-y, --hide-hyperlink-hints`

Hide text hints that hyperlinks are clickable.

Default
False

`-i, --images, -e, --no-images`

Whether to render images. By default will autodetect. May significantly affect performance.

`--image-drawing, --id` `<image_drawing>`

The type of image drawing. Accepted values are 'block', 'character', or 'braille'.

Options
block | character | braille

`-c, --color, -o, --no-color`

Whether to render with color. By default will autodetect. Additionally respects NO_COLOR, NBPREVIEW_NO_COLOR, and TERM='dumb'.

`--color-system, --cs` `<color_system>`

The type of color system to use.

Options
standard | 256 | truecolor | windows | none | auto

`-w, --width` `<width>`

Explicitly set the width of the render instead of determining automatically.

`-V, --version`

Display the version and exit.

`-m, --line-numbers`

Show line numbers for code in cells.

Default
False

`-q, --code-wrap`

Wrap code onto the next line if it does not fit in width. May be used with `--line-numbers` for clarity.

Default
False

`-g, --paging, -f, --no-paging`

Whether to display the output in a pager. By default autodetects.

`--install-completion` `<install_completion>`

Install completion for the specified shell.

Options
bash | zsh | fish | powershell | pwsh

`--show-completion` `<show_completion>`

Show completion for the specified shell, to copy it or customize the installation.

Options
bash | zsh | fish | powershell | pwsh

Arguments

`FILE`

Optional argument(s)

Environment variables

`NBPREVIEW_THEME`

Provide a default for *`--theme`*

`NBPREVIEW_PLAIN`

Provide a default for *`--plain`*

`NBPREVIEW_UNICODE`

Provide a default for *`--unicode`*

`NBPREVIEW_HIDE_OUTPUT`

Provide a default for *`--hide-output`*

`NBPREVIEW_NERD_FONT`

Provide a default for *`--nerd-font`*

`NBPREVIEW_NO_FILES`

Provide a default for *`--no-files`*

`NBPREVIEW_POSITIVE_SPACE`

Provide a default for *`--positive-space`*

`NBPREVIEW_HYPERLINKS`

Provide a default for *`--hyperlinks`*

`NBPREVIEW_HIDE_HYPERLINK_HINTS`

Provide a default for *`--hide-hyperlink-hints`*

`NBPREVIEW_IMAGES`

Provide a default for *`--images`*

`NBPREVIEW_IMAGE_DRAWING`

Provide a default for *`--image-drawing`*

`NBPREVIEW_COLOR`

Provide a default for *`--color`*

`NBPREVIEW_COLOR_SYSTEM`

Provide a default for *`--color-system`*

`NBPREVIEW_WIDTH`

Provide a default for *`--width`*

`NBPREVIEW_LINE_NUMBERS`

Provide a default for *`--line-numbers`*

`NBPREVIEW_CODE_WRAP`

Provide a default for *`--code-wrap`*

`NBPREVIEW_PAGING`

Provide a default for *`--paging`*

CHAPTER

TWO

FEATURES

2.1 Flexible FILE argument

nbpreview has only one required argument—*FILE*—which expects a Jupyter notebook (.ipynb) file path.

```
% nbpreview notebook.ipynb
```

FILE is a flexible argument. It can take in multiple files and render them all at once. nbpreview will accept multiple file paths manually listed out,

```
% nbpreview notebook1.ipynb notebook2.ipynb
```

or a glob that expands to one or more notebook files.

```
% nbpreview notebooks/*.ipynb
```

FILE also accepts text from stdin and treats it as the contents of a notebook file. This can be used to easily view notebooks from the web[1] using curl[2].

```
% curl https://raw.githubusercontent.com/paw-lu/nbpreview/main/tests/unit/assets/
↪notebook.ipynb | nbpreview
```

This can even be used to filter cells before rendering them. For example, jq[3] can be used to select only the markdown cells from a notebook. These cells are then passed on to nbpreview to render.

```
% jq 'with_entries(if .key == "cells" then .value |= map(select(.cell_type == "markdown
↪")) else . end)' tests/unit/assets/notebook.ipynb | nbp
```

2.2 Smart output

2.2.1 Automatic plain output

nbpreview is smart about its output. By default it will strip out decorations—such as boxes, execution counts, and extra spacing—when its output is piped to stdout. This makes nbpreview usable as a preprocessor for other command-line tools. For example, if fgrep[4] is used to search a notebook file for the string 'parietal', the output can be difficult to parse.

[1] Like always, do not view notebooks from untrusted sources.
[2] curl is a command-line tool to transfer data from servers. In this example it was used to download the file contents from an address.
[3] jq is a command-line JSON processor. Since Jupyter notebook (ipynb) files are in a JSON format, it can be used to filter and transform cells.
[4] fgrep is equivalent to running grep -F—which searches an input file for the literal text given.

```
% fgrep parietal notebook.ipynb
      "      <td>parietal</td>\n",
      "      <td>parietal</td>\n",
      "      <td>parietal</td>\n",
      "      <td>parietal</td>\n",
      "      <td>parietal</td>\n",
      "0     s13         18  stim  parietal -0.017552\n",
      "1      s5         14  stim  parietal -0.080883\n",
      "2     s12         18  stim  parietal -0.081033\n",
      "3     s11         18  stim  parietal -0.046134\n",
      "4     s10         18  stim  parietal -0.037970"
```

Instead, if the notebook is run through nbpreview first, it will process the file before passing it onto fgrep, creating a more human-readable output.

```
% nbpreview notebook.ipynb | fgrep parietal
0     s13         18  stim  parietal -0.017552
1      s5         14  stim  parietal -0.080883
2     s12         18  stim  parietal -0.081033
3     s11         18  stim  parietal -0.046134
4     s10         18  stim  parietal -0.037970
```

Plain rendering can be manually forced by using the *--plain* (or *-p*) option,

```
% nbpreview --plain notebook.ipynb
```

or completely disabled by using the *--decorated* (or *-d*) option.

```
% nbpreview --decorated notebook.ipynb
```

This can be configured by setting the *NBPREVIEW_PLAIN* environmental variable. For example, to set the default rendering to be plain, run:

```
% export NBPREVIEW_PLAIN=1
```

2.2.2 Automatic paging

nbpreview will automatically view the output in a pager if the output is longer than the terminal—which is often. Similar to the *automatic plain output*, this will be automatically disabled when piping to other commands.

Thanks to `Click`, nbpreview attempts to choose a pager that renders the notebook in color. If the `PAGER` environmental variable is set, nbpreview will use the value as the pager command. To disable the automatic paging, use the *--no-paging* (or *-f*) option.

```
% nbpreview --no-paging notebook.ipynb
```

Conversely, to manually force paging, use the *--paging* (or *-g*) option. This can be configured by setting the *NBPREVIEW_PAGING* environmental variable.

2.3 Syntax highlighting

2.3.1 Themes

Thanks to Pygments and `Rich`, nbpreview comes with many different syntax highlighting themes. They can be applied using the *`--theme`* (or *`-t`*) option. Some themes may clash with the terminal theme, but `'dark'`—the default theme—and `'light'` will match the terminal's colors.

material

dracula

one-dark

monokai

paraiso-light

rainbow_dash

For a list of all available themes along with a preview of how they look on the terminal use the *`--list-themes`* option.

```
% nbpreview --list-themes
```

2.3.2 Cell magic

Certain cell magics may be used to run other languages in a Jupyter Notebook cell. nbpreview detects the use of these magic commands and adjusts its syntax highlighting to match it. For example, here it switches to bash syntax highlighting when the `%%bash` cell magic is used.

2.3.3 Multi-language support

Jupyter Notebooks are not Python exclusive. nbpreview will detect the usage of other languages—such as Julia.

2.3.4 Wrapping and line numbers

Depending on your terminal size, code cell contents might be too long to fit on the terminal. By default, nbpreview truncates the long code. But if *`--code-wrap`* (or *`-q`*) is used, nbpreview will wrap the code around so that it's all visible. It's usually best to use this will *`--line-numbers`* (or *`-m`*) to enable line numbers—so that wrapping is clearly distinguished from a line break.

2.4 Markdown rendering

Thanks to Rich, markdown-it-py, and pylatexenc, nbpreview renders markdown content with some extensions. In addition to typical CommonMark, nbpreview will also render markdown tables, create clickable hyperlinks (if it's supported by the terminal), syntax highlight code blocks (which respect `--theme`), and render block math equations. It will even render images—which respect `--image-drawing`. For example,

~~~
# Lorem ipsum

Lorem ipsum dolor sit amet,
consectetur **adipiscing** elit,
sed do eiusmod tempor incididunt
ut labore et dolore magna [aliqua](https://github.com/paw-lu/nbpreview).

$$
\alpha \sim \text{Normal(0, 1)}
$$

_Ut enim ad minim veniam_,
quis nostrud exercitation ullamco
Excepteur sint occaecat `cupidatat` non proident,
sunt in culpa qui.

![Turtle](emoji_u1f422.png)

## At ultrices

```python
def add(x: float, y: float) -> float:
    """Add two numbers."""
    return x + y
```

| Lorep | ipsum | doret |
| ----- | ----- | ----- |
| 1     | 2     | 3     |
| 4     | 5     | 6     |
~~~

renders as

2.5 Images

Thanks to Picharsso and `term-image`, nbpreview renders images.

2.5.1 Drawing types

The *`--image-drawing`* (or *`--id`*) option can be used to control the method nbpreview uses to draw images.

block

character

braille

2.5.2 Negative and positive space

By default, nbpreview draws figures in negative space—meaning characters are used to draw the dark portions of the image. This works well as a default since most charts have a light background by default. However, when working with darker images—like if a dark theme is being used on a plot—the drawing can be switched to positive space using the *`--positive-space`* (or *`-s`*) option.

Attention: *`--positive-space`* only works on *`--image-drawing='character'`*. *`--image-drawing='braille'`* only draws in positive space.

character (positive space)

character (negative space)

braille

block

2.5.3 Enabling and disabling image rendering

By default, nbpreview will attempt to detect if images can be viewed on the terminal. This can be manually controlled via the *`--images`* or *`--no-images`* options.

Caution: Rendering images can impact nbpreview's performance—especially if the notebook contains many images. The drawing type selected via *`--image-drawing`* can play a role in how severe the performance impact is.

2.6 DataFrame rendering

Thanks to `Rich` and `lxml`, nbpreview renders Pandas DataFrame as a table.

2.7 Vega and VegaLite charts

nbpreview will renderer static previews of Vega and VegaLite charts along with a link to an interactive version (thanks to justcharts).

2.8 $\LaTeX$

Thanks to `pylatexenc`, nbpreview can render $\LaTeX$ as unicode characters.

2.9 HTML

Thanks to html2text, nbpreview renders basic HTML. It will also generate a link to the output so it can be easily previewed in the browser.

2.10 Hyperlinks

With certain complex content—such as images and HTML—nbpreview will display hyperlinks to them in the render.

The hyperlinks will only work if supported by the terminal. nbpreview attempts to detect this, but it can be manually controlled through the *`--hyperlinks`* or *`--no-hyperlinks`* options. If hyperlinks are not enabled, the link address will instead be directly printed to the terminal so that it's easy to click or copy.

By default, nbpreview displays a hint message that prompts the user to click on the link. These hints may be removed by using the *`--hide-hyperlink-hints`* (or *`-y`*) option.

To create previews, nbpreview will write the content to temporary files as the notebook is rendered. To prevent nbpreview from writing files to your machine, use the *`--no-files`* (or *`-l`*) option.

2.11 Nerd Fonts

By default, nbpreview uses emoji to highlight certain content (*like clickable links*). Instead of using emoji, nbpreview also supports using icons from Nerd Fonts[5]. Simply use the *`--nerd-font`* option to enable them.

Attention: You'll need to have a Nerd Font installed and applied to your terminal to view the Nerd Font icons—or else you'll get tofu () characters where the icons should be.

[5] Nerd Fonts are fonts patched with support for extra icons.

2.12 Stderr

Similar to Jupyter Notebooks, stderr text is highlighted in a bright red box.

2.13 Tracebacks

Tracebacks are rendered with syntax highlighting.

CHAPTER

THREE

CONFIGURE

Every option in **nbpreview** has an associated environmental variable that can be set to provide a default value. For example, to set the theme to `'material'`, run:

```
% nbpreview --theme='material' notebook.ipynb
```

To apply the `'material'` theme without having to specify it in the *--theme* option, set the environmental variable associated with the command-line option. The environmental variables for each option are explicitly listed at the end of the *command-line usage*. They may also be found in the *--help* message under `env var:`.

```
% nbpreview --help

  -t, --theme
                                  The theme to use for syntax highlighting.
                                  Call '--list-themes' to preview all
                                  available themes.  [env var:
                                  NBPREVIEW_THEME; default: dark]
```

In the case of *--theme*, the environmental variable is *NBPREVIEW_THEME*. Set it by running

```
% export NBPREVIEW_THEME='material'
```

Now, whenever nbpreview is run, it will automatically set the *--theme* value to `'material'`. To set this permanently, set the environmental variable in the shell's startup file—such as `~/.zshrc`, `~/.zshenv`, `~/.bashrc`, `~/.bash_profile`, etc. Environmental variables set the new default for nbpreview, but they can still be overridden anytime by manually the relevant command-line option.

REFERENCE

class nbpreview.notebook.**Notebook**(*notebook_node*, *theme='ansi_dark'*, *plain=None*, *unicode=None*, *hide_output=False*, *nerd_font=False*, *files=True*, *negative_space=True*, *hyperlinks=None*, *hide_hyperlink_hints=False*, *images=None*, *image_drawing=None*, *color=None*, *relative_dir=None*, *line_numbers=False*, *code_wrap=False*)

Construct a Notebook object to render Jupyter Notebooks.

Parameters

- **notebook_node** (*NotebookNode*) – A NotebookNode of the notebook to render.
- **theme** (*str*) – The theme to use for syntax highlighting. May be `'ansi_light'`, `'ansi_dark'`, or any Pygments theme. By default `'ansi_dark'`.
- **plain** (*Optional[bool]*) – Only show plain style. No decorations such as boxes or execution counts. If set to `None` will autodetect. By default `None`.
- **unicode** (*Optional[bool]*) – Whether to use unicode characters to render the notebook. If set to `None` will autodetect. By default `None`.
- **hide_output** (*bool*) – Do not render the notebook outputs. By default `False`.
- **nerd_font** (*bool*) – Use nerd fonts when appropriate. By default `False`.
- **files** (*bool*) – Create files when needed to render HTML content. By default `True`.
- **negative_space** (*bool*) – Whether render character images in negative space. By default `True`
- **hyperlinks** (*Optional[bool]*) – Whether to use hyperlinks. If `False` will explicitly print out path. If set to `None` will autodetect. By default `None`.
- **hide_hyperlink_hints** (*bool*) – Hide text hints of when content is clickable. By default `False`.
- **images** (*Optional[bool]*) – Whether to render images. If set to `None` will autodetect. By default `None`.
- **image_drawing** (*Optional[Union[ImageDrawingEnum, Literal['block', 'character', 'braille']]]*) – The characters used to render images. Options are `'block'`, `'character'`, `'braille'` or `None`. If set to `None` will autodetect. By default `None`.
- **color** (*Optional[bool]*) – Whether to use color. If set to `None` will autodetect. By default `None`.

- **relative_dir** (*dataclasses.InitVar[Optional[pathlib.Path]]*) – The directory to prefix relative paths to convert them to absolute. If `None` will assume current directory is relative prefix. By default `None`.
- **line_numbers** (*bool*) – Whether to render line numbers in code cells. By default `False`.
- **code_wrap** (*bool*) – Whether to wrap code if it does not fit. By default `False`.

classmethod from_file(*file*, *theme='ansi_dark'*, *plain=None*, *unicode=None*, *hide_output=False*, *nerd_font=False*, *files=True*, *negative_space=True*, *hyperlinks=None*, *hide_hyperlink_hints=False*, *images=None*, *image_drawing=None*, *color=None*, *line_numbers=False*, *code_wrap=False*)

Create a Notebook from notebook file.

Parameters

- **file** (`Union`[`Path`, `IO`[`AnyStr`], `KeepOpenFile`]) – A path to a Jupyter Notebook file.
- **theme** (`str`) – The theme to use for syntax highlighting. May be `'ansi_light'`, `'ansi_dark'`, or any Pygments theme. By default `'ansi_dark'`.
- **plain** (`Optional`[`bool`]) – Only show plain style. No decorations such as boxes or execution counts. If set to `None` will autodetect. By default `None`.
- **unicode** (`Optional`[`bool`]) – Whether to use unicode characters to render the notebook. If set to `None` will autodetect. By default `None`.
- **hide_output** (`bool`) – Do not render the notebook outputs. By default `False`.
- **nerd_font** (`bool`) – Use nerd fonts when appropriate. By default `False`.
- **files** (`bool`) – Create files when needed to render HTML content. By default `True`.
- **negative_space** (`bool`) – Whether render character images in negative space. By default `True`.
- **hyperlinks** (`Optional`[`bool`]) – Whether to use hyperlinks. If `False` will explicitly print out path. If set to `None` will autodetect. By default `None`.
- **hide_hyperlink_hints** (`bool`) – Hide text hints of when content is clickable. By default `False`.
- **images** (`Optional`[`bool`]) – Whether to render images. If set to `None` will autodetect. By default `None`.
- **image_drawing** (`Union`[`ImageDrawingEnum`, `Literal`['block', 'character', 'braille'], `None`]) – The characters used to render images. Options are `'block'`, `'character'`, `'braille'` or `None`. If set to `None` will autodetect. By default `None`.
- **color** (`Optional`[`bool`]) – Whether to use color. If set to `None` will autodetect. By default `None`.
- **line_numbers** (`bool`) – Whether to render line numbers in code cells. By default `False`.
- **code_wrap** (`bool`) – Whether to wrap code if it does not fit. By default `False`.

Returns

A Notebook object created from the file.

Return type

Notebook

Raises

InvalidNotebookError – If the file is not a valid Jupyter notebook.

CHAPTER

FIVE

CONTRIBUTOR GUIDE

Thank you for your interest in improving this project. This project is open-source under the MIT license and welcomes contributions in the form of bug reports, feature requests, and pull requests.

Here is a list of important resources for contributors:

- Source Code
- Documentation
- Issue Tracker
- *Code of Conduct*

5.1 How to report a bug

Report bugs on the Issue Tracker.

When filing an issue, make sure to answer these questions:

- Which operating system and Python version are you using?
- Which version of this project are you using?
- What did you do?
- What did you expect to see?
- What did you see instead?

The best way to get your bug fixed is to provide a test case, and/or steps to reproduce the issue.

5.2 How to request a feature

Request features on the Issue Tracker.

5.3 How to set up your development environment

You need Python 3.8+ and the following tools:

- Poetry
- Nox
- nox-poetry

Install the package with development requirements:

```
$ poetry install
```

You can now run an interactive Python session, or the command-line interface:

```
$ poetry run python
$ poetry run nbpreview
```

5.4 How to test the project

Run the full test suite:

```
$ nox
```

List the available Nox sessions:

```
$ nox --list-sessions
```

You can also run a specific Nox session. For example, invoke the unit test suite like this:

```
$ nox --session=tests
```

Unit tests are located in the `tests` directory, and are written using the pytest testing framework.

5.5 How to submit changes

Open a pull request to submit changes to this project.

Your pull request needs to meet the following guidelines for acceptance:

- The Nox test suite must pass without errors and warnings.
- Include unit tests. This project maintains 100% code coverage.
- If your changes add functionality, update the documentation accordingly.

Feel free to submit early, though—we can always iterate on this.

To run linting and code formatting checks before committing your change, you can install pre-commit as a Git hook by running the following command:

```
$ nox --session=pre-commit -- install
```

It is recommended to open an issue before starting work on anything. This will allow a chance to talk it over with the owners and validate your approach.

CHAPTER

SIX

PRIOR ART

6.1 Similar tools

Thanks to @joouha for maintaining a list of these tools. Many of the projects here were found directly on their page.

- ipynb-term
- ipynbat
- ipynbviewer
- jcat
- jupview
- jupytui
- jut
- nbcat
- nbtui
- nbv
- Read-Jupyter-Notebook

6.2 Complimentary tools

If you're interested in complimentary tools that help improve the terminal experience for notebooks, there are many amazing projects out there.

- **bat** is not a tool for notebooks specifically. But similar to nbpreview, it provides a rich output for many types of files on the terminal, and is the primary inspiration for nbpreview.
- **euporie** is a really exciting project that allows you to edit and run Jupyter notebooks on the terminal.
- **nbclient** is a library for executing notebooks from the command line.
- **nbpreview** is another project that coincidentally shares a name with this one. It allows for Jupyter notebooks to be rendered without running a notebook server.
- **nbqa** allows the use of linters and formatters on notebooks. It's also used by this project.
- **jpterm** is and up-and-coming successor to nbterm which will be accompanied by a web client. Looking forward to seeing this develop.
- **nbtermix** is an actively-developed fork of nbterm.

- **nbterm** lets you edit and execute Jupyter Notebooks on the terminal.
- **papermill** allows the parameterization and execution of Jupyter Notebooks.

CHAPTER SEVEN

CREDITS

nbpreview relies on a lot of fantastic projects. Check out the *dependencies* for a complete list of libraries that are leveraged.

Besides the direct dependencies, there are some other projects that directly enabled the development of nbpreview.

- **bat** is not explicitly used in this project, but served as the primary inspiration. This projects strives to be bat—but for notebooks. Many of nbpreview's features and command-line options are directly adopted from bat.
- **Hypermodern Python Cookiecutter** is the template this project was generated on. It is a fantastic project that integrates Poetry, Nox, and pre-commit. It's responsible for most of this project's CI.
- **justcharts** is directly used by this project to generate the Vega and Vega-Lite charts.

CHAPTER

EIGHT

DEPENDENCIES

```
[tool.poetry.dependencies]
python = "^3.8.0"
rich = ">=12.4.1"
typer = ">=0.4.1,<0.6.0"
nbformat = { extras = ["fast"], version = ">=5.2.0" }
Pygments = ">=2.10.0"
ipython = ">=7.27,<9.0"
lxml = ">=4.6.3"
pylatexenc = ">=2.10"
httpx = ">=0.19,<0.24"
Jinja2 = ">=3.0.1"
html2text = ">=2020.1.16"
types-click = ">=7.1.5"
Pillow = ">=8.3.1,<10.0.0"
picharsso = ">=2.0.1"
validators = ">=0.18.2,<0.21.0"
yarl = ">=1.6.3"
markdown-it-py = ">=1.1,<3.0"
mdit-py-plugins = ">=0.3.0"
click-help-colors = ">=0.9.1"
term-image = ">=0.3.0"

[tool.poetry.dev-dependencies]
pytest = ">=7.1.2"
coverage = { extras = ["toml"], version = ">=6.4" }
safety = ">=2.0.0"
mypy = ">=0.961"
typeguard = ">=2.13.3"
xdoctest = { extras = ["colors"], version = ">=1.0.0" }
sphinx = ">=5.0.2"
sphinx-autobuild = ">=2021.3.14"
pre-commit = ">=2.19.0"
flake8 = ">=4.0.1"
black = { extras = ["jupyter"], version = ">=21.12b0" }
flake8-bandit = ">=3.0.0"
flake8-bugbear = ">=22.6.22"
flake8-docstrings = ">=1.5.0"
flake8-rst-docstrings = ">=0.2.6"
pep8-naming = ">=0.13.0"
darglint = ">=1.8.1"
```

(continues on next page)

(continued from previous page)

```
pre-commit-hooks = ">=4.3.0"
sphinx-click = ">=4.2.0"
Pygments = ">=2.10.0"
pyupgrade = ">=2.34.0"
furo = ">=2021.11.12"
pdbpp = ">=0.10.3"
ipykernel = ">=6.15.0"
pytest-mock = ">=3.8.1"
interrogate = ">=1.5.0"
isort = ">=5.10.1"
nbqa = ">=1.3.1"
click = ">=8.1.3"
autoflake = ">=1.4"
myst-parser = ">=0.18.0"
sphinxext-opengraph = ">=0.6.3"
sphinx-copybutton = ">=0.5.0"
sphinx-design = ">=0.2.0"
sphinx-autodoc-typehints = ">=1.18.3"
tomli = ">=2.0.1"
sphinx-favicon = ">=0.2"
```

CHAPTER

NINE

CONTRIBUTOR COVENANT CODE OF CONDUCT

9.1 Our Pledge

We as members, contributors, and leaders pledge to make participation in our community a harassment-free experience for everyone, regardless of age, body size, visible or invisible disability, ethnicity, sex characteristics, gender identity and expression, level of experience, education, socio-economic status, nationality, personal appearance, race, religion, or sexual identity and orientation.

We pledge to act and interact in ways that contribute to an open, welcoming, diverse, inclusive, and healthy community.

9.2 Our Standards

Examples of behavior that contributes to a positive environment for our community include:

- Demonstrating empathy and kindness toward other people
- Being respectful of differing opinions, viewpoints, and experiences
- Giving and gracefully accepting constructive feedback
- Accepting responsibility and apologizing to those affected by our mistakes, and learning from the experience
- Focusing on what is best not just for us as individuals, but for the overall community

Examples of unacceptable behavior include:

- The use of sexualized language or imagery, and sexual attention or advances of any kind
- Trolling, insulting or derogatory comments, and personal or political attacks
- Public or private harassment
- Publishing others' private information, such as a physical or email address, without their explicit permission
- Other conduct which could reasonably be considered inappropriate in a professional setting

9.3 Enforcement Responsibilities

Community leaders are responsible for clarifying and enforcing our standards of acceptable behavior and will take appropriate and fair corrective action in response to any behavior that they deem inappropriate, threatening, offensive, or harmful.

Community leaders have the right and responsibility to remove, edit, or reject comments, commits, code, wiki edits, issues, and other contributions that are not aligned to this Code of Conduct, and will communicate reasons for moderation decisions when appropriate.

9.4 Scope

This Code of Conduct applies within all community spaces, and also applies when an individual is officially representing the community in public spaces. Examples of representing our community include using an official e-mail address, posting via an official social media account, or acting as an appointed representative at an online or offline event.

9.5 Enforcement

Instances of abusive, harassing, or otherwise unacceptable behavior may be reported to the community leaders responsible for enforcement at mailto:Paulo.S.Costa5@gmail.com. All complaints will be reviewed and investigated promptly and fairly.

All community leaders are obligated to respect the privacy and security of the reporter of any incident.

9.6 Enforcement Guidelines

Community leaders will follow these Community Impact Guidelines in determining the consequences for any action they deem in violation of this Code of Conduct:

9.6.1 1. Correction

Community Impact: Use of inappropriate language or other behavior deemed unprofessional or unwelcome in the community.

Consequence: A private, written warning from community leaders, providing clarity around the nature of the violation and an explanation of why the behavior was inappropriate. A public apology may be requested.

9.6.2 2. Warning

Community Impact: A violation through a single incident or series of actions.

Consequence: A warning with consequences for continued behavior. No interaction with the people involved, including unsolicited interaction with those enforcing the Code of Conduct, for a specified period of time. This includes avoiding interactions in community spaces as well as external channels like social media. Violating these terms may lead to a temporary or permanent ban.

9.6.3 3. Temporary Ban

Community Impact: A serious violation of community standards, including sustained inappropriate behavior.

Consequence: A temporary ban from any sort of interaction or public communication with the community for a specified period of time. No public or private interaction with the people involved, including unsolicited interaction with those enforcing the Code of Conduct, is allowed during this period. Violating these terms may lead to a permanent ban.

9.6.4 4. Permanent Ban

Community Impact: Demonstrating a pattern of violation of community standards, including sustained inappropriate behavior, harassment of an individual, or aggression toward or disparagement of classes of individuals.

Consequence: A permanent ban from any sort of public interaction within the community.

9.7 Attribution

This Code of Conduct is adapted from the Contributor Covenant, version 2.0, available at https://www.contributor-covenant.org/version/2/0/code_of_conduct.html.

Community Impact Guidelines were inspired by Mozilla's code of conduct enforcement ladder.

For answers to common questions about this code of conduct, see the FAQ at https://www.contributor-covenant.org/faq. Translations are available at https://www.contributor-covenant.org/translations.

CHAPTER TEN

LICENSE

INDEX