



The tkintermd Documentation

Installation, usage instructions and API reference.

Table of contents

1. tkintermd	3
1.1 About	3
1.2 Contributing	4
1.3 MIT License	5
2. Getting Started	6
2.1 Installation	6
2.2 Usage	6
3. Reference	7
3.1 Reference	7
3.2 Constants	8
3.3 Lexer	9
3.4 TkintermdFrame	10

1. tkintermd

1.1 About

1.1.1 About The Project

An embeddable tkinter based Markdown editor with HTML preview. The editor has syntax highlighting provided by `Pygments` and the HTML preview window is provided by `tkinterweb`.

- Github Link: <https://github.com/hreikin/tkintermd>
- Github v0.1.0 Project: [Github Project - tkintermd v0.1.0](#)
- Github Discussion: [Github Discussions](#)
- PyPi Link: <https://pypi.org/project/tkintermd/>
- PDF Documentation: <https://hreikin.github.io/tkintermd/pdf/tkintermd-documentation-LATEST.pdf>

Built With

- [Pygments](#)
- [Pydownx](#)
- [Python](#)
- [Python Markdown](#)
- [tkinter](#)
- [tkinterweb](#)

Last update: 2023-09-26

1.2 Contributing

Check out the [Github Project - tkintermd v0.1.0](#) for an overview of the work being done towards the release. See the [open issues](#) for a full list of proposed features (and known issues).

Contributions are what make the open source community such an amazing place to learn, inspire, and create. Any contributions you make are **greatly appreciated**.

If you have a suggestion that would make this better, please fork the repo and create a pull request. You can also simply open an issue with the tag "enhancement". Don't forget to give the project a star! Thanks again!

- Fork the Project
- Create your Feature Branch (`git checkout -b feature/AmazingFeature`)
- Commit your Changes (`git commit -m 'Add some AmazingFeature'`)
- Push to the Branch (`git push origin feature/AmazingFeature`)
- Open a Pull Request

Last update: 2023-09-26

1.3 MIT License

Copyright 2022 @hreikin (hreikin@gmail.com)

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.

Last update: 2023-09-26

2. Getting Started

2.1 Installation

To get a local copy up and running choose one of the below install instructions and follow the steps provided.

2.1.1 Install With PIP

The simplest way to install `tkintermd` is to use `pip`:

```
pip install tkintermd
```

2.1.2 Install From Source

Alternatively you can install from source by following the steps below:

1. Clone the repo:

```
git clone https://github.com/hreikin/tkintermd.git
cd tkintermd/
```

1. Create and source a Python virtual environment:

```
python3 -m venv .venv
source .venv/bin/activate
```

1. Install requirements with `pip`:

```
pip install -r requirements.txt
```

2.2 Usage

2.2.1 Embedded

To use the `TkintermdFrame` in one of your own python scripts:

```
from tkintermd.frame import TkintermdFrame

import tkinter as tk
from tkinter.constants import *

root = tk.Tk()
app = TkintermdFrame(root)
app.pack(fill="both", expand=1)
app.mainloop()
```

Last update: 2023-09-26

3. Reference

3.1 Reference

3.1.1 tkintermd

- [Constants](#)
- [Lexer](#)
- [TkintermdFrame](#)

Last update: 2023-09-26

3.2 Constants

3.2.1 tkintermd.constants

Variables and constants to be used by tkintermd.

Parameters:

Name	Type	Description	Default
<code>cur_file</code>	Path	Variable for tracking current open file.	<i>required</i>
<code>edit_warning</code>	str	Warning message displayed to user when enabling edit before export functionality.	<i>required</i>
<code>template_list</code>	list	List of available template names.	<i>required</i>
<code>template_dict</code>	dict	Dictionary containing lists of template values linked with the appropriate name.	<i>required</i>
<code>BOLD_MD_SYNTAX</code>	tuple	Markdown syntax for bold highlighting.	<i>required</i>
<code>BOLD_MD_IGNORE</code>	tuple	Markdown syntax to ignore for bold highlighting.	<i>required</i>
<code>BOLD_MD_SPECIAL</code>	tuple	Markdown syntax to ignore for bold highlighting that requires special handling.	<i>required</i>
<code>ITALIC_MD_SYNTAX</code>	tuple	Markdown syntax for italic highlighting.	<i>required</i>
<code>ITALIC_MD_IGNORE</code>	tuple	Markdown syntax to ignore for italic highlighting.	<i>required</i>
<code>ITALIC_MD_SPECIAL</code>	tuple	Markdown syntax to ignore for italic highlighting that requires special handling.	<i>required</i>
<code>BOLD_ITALIC_MD_SYNTAX</code>	tuple	Markdown syntax for bold-italic highlighting.	<i>required</i>
<code>BOLD_ITALIC_MD_IGNORE</code>	tuple	Markdown syntax to ignore for bold-italic highlighting.	<i>required</i>
<code>BOLD_ITALIC_MD_SPECIAL</code>	tuple	Markdown syntax to ignore for bold-italic highlighting that requires special handling.	<i>required</i>
<code>STRIKETHROUGH_MD_SYNTAX</code>	tuple	Markdown syntax for strikethrough highlighting.	<i>required</i>
<code>STRIKETHROUGH_MD_IGNORE</code>	tuple	Markdown syntax to ignore for strikethrough highlighting.	<i>required</i>
<code>DEFAULT_MD_STRING</code>	str	Default string to show in the editor when it loads.	<i>required</i>
<code>DEFAULT_TEMPLATE_TOP</code>	str	Default template, top portion.	<i>required</i>
<code>DEFAULT_TEMPLATE_MIDDLE</code>	str	Default template, middle portion.	<i>required</i>
<code>DEFAULT_TEMPLATE_BOTTOM</code>	str	Default template, bottom portion.	<i>required</i>
<code>CENTERED_TEMPLATE_TOP</code>	str	Centered template, top portion.	<i>required</i>
<code>CENTERED_TEMPLATE_MIDDLE</code>	str	Centered template, middle portion.	<i>required</i>
<code>CENTERED_TEMPLATE_BOTTOM</code>	str	Centered template, bottom portion.	<i>required</i>

Last update: 2023-09-26

3.3 Lexer

3.3.1 `tkintermd.frame.Lexer`

Bases: `MarkdownLexer`

Extend `MarkdownLexer` to add markup for bold-italic.

This needs extending further before being complete.

Last update: 2023-09-26

3.4 TkintermdFrame

3.4.1 tkintermd.frame.TkintermdFrame

Bases: `tk.Frame`

A Markdown editor with HTML preview for use in tkinter projects.

The editor has syntax highlighting supplied by `Pygments` and the HTML preview window is provided by `tkinterweb`.

Import it into your own scripts like so:

```
from tkintermd.frame import TkintermdFrame

import tkinter as tk
from tkinter.constants import *

root = tk.Tk()
app = TkintermdFrame(root)
app.pack(fill="both", expand=1)
app.mainloop()
```

`apply_markdown_both_sides(selection, md_syntax)`

Apply markdown to both sides of a selection.

Parameters:

Name	Type	Description	Default
<code>selection</code>	<code>str</code>	Text selection from the editor.	<i>required</i>
<code>md_syntax</code>	<code>tuple</code>	Tuple of markdown strings to apply.	<i>required</i>

`change_template(template_name)`

Change the currently selected template.

Get the selected template name from the `StringVar` for the `Combobox` and compare it with the templates dictionary. If the name matches the key then set the relevant template values and update all the previews.

`check_markdown_both_sides(md_syntax, md_ignore, md_special, strikethrough=None)`

Check markdown formatting to be applied to both sides of a selection.

This will ignore items in the `md_ignore` variable and then deal with special syntax individually before applying or removing the markdown formatting.

- If string starts with anything in `md_ignore` do nothing and return from the function.
- If `strikethrough` is set to `True` then apply or remove the markdown.
- If the formatting requires special items which can't go in `md_ignore` because they cause issues with markdown being applied incorrectly do nothing and return from the function.
- Apply or remove the markdown once we reach the end.

Parameters:

Name	Type	Description	Default
<code>selection</code>	<code>str</code>	Text selection from the editor.	<i>required</i>
<code>md_syntax</code>	<code>tuple</code>	Tuple of markdown strings to remove.	<i>required</i>
<code>md_ignore</code>	<code>tuple</code>	Tuple of markdown strings to ignore.	<i>required</i>
<code>md_special</code>	<code>tuple</code>	Tuple of special markdown strings to ignore that cause unexpected issues when included in <code>md_ignore</code> .	<i>required</i>
<code>strikethrough</code>	<code>bool</code>	Set to True for strikethrough. Default is <code>None</code> .	<code>None</code>

`check_markdown_highlighting(start='insert linestart', end='insert lineend')`

Formats editor content using the Pygments style.

`enable_edit()`

Enable editing of HTML before export.

Displays a warning to the user and enables HTML editing prior to export.

`find(*args)`

Simple search dialog for within the editor window.

- Get the current text area content.
- Displays a simple dialog with a field to enter a search string into and two buttons.
- If a string is provided then search for it within the text area.
- Add tag TAGNAME to all characters between INDEX1 and INDEX2.
- Highlight any found strings within the text editor.

`load_style(stylename)`

Load Pygments style for syntax highlighting within the editor.

- Load and configure the text area and `tags` with the Pygments styles and custom `Lexer` tags.
- Create the CSS styling to be merged with the HTML template
- Generate a `<<Modified>>` event to update the styles when a new style is chosen.

`on_input_change(event)`

Converts the text area input into html output for the HTML preview.

When the user types:

- Get the current text area contents.
- Convert the markdown formatted string to HTML.
- Merge the converted markdown with the currently selected template.
- Load the merged HTML into the document preview.
- Update the HTML content/states within the export options edit area.
- Check the markdown and apply formatting to the text area.
- Reset the modified flag.

`open_md_file()`

Open a file and clear/insert the text into the `text_area`.

Opens a native OS dialog and expects markdown formatted files. Shows an error message if it fails.

- Display a native OS dialog and request a filename to open.
- If a filename is provided then `try` to open it in "read" mode.
- Replace the text area content.
- Set the `constants.cur_file` value to the filename that was opened.
- If any of the above fails then display an error message.

`popup(event)`

Right-click popup at mouse location within the text area only.

Provides the following options:

- Cut.
- Copy.
- Paste.
- Undo.
- Redo.
- Find.
- Select All.

`remove_markdown_both_sides(selection, md_syntax)`

Remove markdown from both sides of a selection.

Parameters:

Name	Type	Description	Default
<code>selection</code>	<code>str</code>	Text selection from the editor.	<i>required</i>
<code>md_syntax</code>	<code>tuple</code>	Tuple of markdown strings to remove.	<i>required</i>

`save_as_html_file()`

Exports the current contents of the HTML preview pane to the given filename.

Opens a native OS dialog for saving the file with a html extension. Shows an error message if it fails.

- Display a native OS dialog and request a filename to save.
- If a filename is provided then `try` to open it in "write" mode.
- If any of the above fails then display an error message.

`save_as_md_file()`

Saves the file with the given filename.

Opens a native OS dialog for saving the file with a name and markdown extension. Shows an error message if it fails.

- Display a native OS dialog and request a filename to save.
- If a filename is provided then `try` to open it in "write" mode.
- Set the `constants.cur_file` value to the filename that was opened.
- If any of the above fails then display an error message.

```
save_md_file()
```

Quick saves the file with its current name.

- Get the current text area content.
- `try` to save the file with the `constants.cur_file` variable.
- If it fails because no name exists it calls the "save_as_md_file" function.

```
select_all(*args)
```

Select all text within the editor window.

- Add tag TAGNAME to all characters between INDEX1 and INDEX2.
- Set mark MARKNAME before the character at index.
- Scroll so that the character at INDEX is visible.

Last update: 2023-09-26