Panargs.py: Minimal CLI Argument Specific for Pandoc Markdown YAML

Githubonline_1396529

Contents

1 Description

`panargs.py` is a Minimal Python script designed to control CLI Arguments when exporting Markdown files to various formats using Pandoc. It combines options specified in the YAML front matter of the Markdown file and additional command-line arguments for enhanced flexibility and customization.

There are already some excellent ready-to-use solutions like htdebeer/pandocomatic and msprev/panzer, both tools are powerful and

well-maintained, offering extensive options and flexibility. However, this comes at the cost of a steeper learning curve, which this script aims to avoid by **providing a direct and idiot-proof solution**.

To be honest most of this script was completed by my dear ChatGPT, with only minimal modifications by me. It was so well-designed from the start that there's not much thing for me to intervene.

2 Features

The script supports specifying Pandoc options in the YAML front matter under the `pandoc_args` key.

Dictionary format:

```yaml
pandoc_args:
  pdf_engine: xelatex
  top_level_division: chapter
```

The script automatically converts YAML-style keys with underscores (`_`) to Pandoc CLI-compatible hyphenated keys (`-`) and run the `pandoc` command with arguments

```
--pdf-engine=xelatex --top-level-division=chapter
```

The script handles multiple input formats for `pandoc_args`

List format:

```yaml
pandoc_args:
  - pdf-engine=xelatex
  - top-level-division=chapter
```

String format:

```yaml
pandoc_args: "--pdf-engine=xelatex --top-level-division=chapter"
```

The script also accepts additional Pandoc options via CLI arguments `--pandoc-arguments` for dynamic configuration.

3 Requirements

1. Python 3.8 or higher
2. Pandoc installed and accessible in your system's PATH

4 Installation

Clone the repository or download `panargs.py`:

```
git clone https://github.com/yourusername/panargs.git
cd panargs
```

5 Usage

```
python panargs.py [options] input.md
```

5.1 Command-line Arguments

- `input.md`: The Markdown file to be processed.
- `--output, -o`: Specify the output file path (optional; default: inferred from input).
- `--pandoc-arguments`: Provide additional Pandoc options via CLI (optional).

5.2 Examples

5.2.1 Basic Conversion Using YAML Options

Add the following to your Markdown file's YAML front matter:

```yaml
pandoc_args:
  pdf_engine: xelatex
  top_level_division: chapter
```

and run:

```
python panargs.py input.md
```

5.2.2 Override YAML Options with CLI:

```
python panargs.py input.md --pandoc-arguments="--pdf-engine=luatex"
```

5.2.3 Specify Output File:

```
python panargs.py -o output.pdf input.md
```

6 How `pandoc_args` Works

The script interprets the `pandoc_args` field in the YAML header, the name of the script is a combination of 'pandoc' and 'arguments'. Keys in the dictionary or list should use underscores (_) instead of hyphens (-) to maintain YAML compliance. The script converts these keys to Pandoc's hyphenated format when constructing the command.

Example YAML:

```yaml
pandoc_args:
  pdf_engine: xelatex
  top_level_division: chapter
```

Generated command:

```
pandoc input.md --pdf-engine=xelatex --top-level-division=chapter
```

7 Development

Feel free to modify or extend the script to fit your needs. Contributions are welcome!

7.1 Running Tests

To verify the script, create test Markdown files with various `pandoc_args` configurations and run the script against them.

7.2 Packaging as `.exe` (Optional)

Use `PyInstaller` to package the script into a standalone executable:

```
pip install pyinstaller
pyinstaller --onefile panargs.py
```

The `.exe` file will be located in the `dist/` folder.

8 License

This project is licensed under the MIT License. See the LICENSE file for details.