Poetry Merge Lock

Claudio Jolowicz

Dec 30, 2021

CONTENTS

CHAPTER ONE

REFERENCE

1.1 poetry_merge_lock.__main__

Command-line interface.

1.2 poetry_merge_lock.core

Core module.

`poetry_merge_lock.core.`**`activate_dependencies`**(*packages*)

Activate the optional dependencies of every package.

Activating optional dependencies ensures their inclusion when the lock file is written. Normally, optional dependencies are activated by the solver if another package depends on them. But invoking the solver would result in regenerating the lock file from scratch, losing the information in the original lock file. So we activate the dependencies manually instead. We know the solver would activate them because they would not be present in the lock file otherwise.

Parameters **`packages`** (*`List[poetry.packages.package.Package]`*) – The list of packages.

Return type None

`poetry_merge_lock.core.`**`load`**(*locker*)

Load a lock file with merge conflicts.

Parameters **`locker`** (*`poetry.packages.locker.Locker`*) – The locker object.

Returns The merged TOML document.

Return type tomlkit.toml_document.TOMLDocument

`poetry_merge_lock.core.`**`load_packages`**(*locker*, *lock_data*)

Load the packages from a TOML document with lock data.

Parameters

- **`locker`** (*`poetry.packages.locker.Locker`*) – The locker object.
- **`lock_data`** (*`tomlkit.toml_document.TOMLDocument`*) – The lock data.

Returns The list of packages.

Return type List[poetry.packages.package.Package]

`poetry_merge_lock.core.`**`load_toml_versions`**`(`*toml_file*`)`

Load a pair of TOML documents from a TOML file with merge conflicts.

Parameters **`toml_file`** (*`pathlib.Path`*) – Path to the lock file.

Returns A pair of TOML documents, corresponding to *our* version and *their* version.

Return type Tuple[tomlkit.toml_document.TOMLDocument, tomlkit.toml_document.TOMLDocument]

`poetry_merge_lock.core.`**`merge_lock`**`(`*poetry*`)`

Resolve merge conflicts in Poetry's lock file.

Parameters **`poetry`** (*`poetry.poetry.Poetry`*) –

Return type None

`poetry_merge_lock.core.`**`save`**`(`*locker*, *lock_data*, *root*`)`

Validate the lock data and write it to disk.

Parameters

- **`locker`** (*`poetry.packages.locker.Locker`*) – The locker object.
- **`lock_data`** (*`tomlkit.toml_document.TOMLDocument`*) – The lock data.
- **`root`** (*`poetry.packages.package.Package`*) – The root package of the Poetry project.

Return type None

1.3 poetry_merge_lock.parser

Line-based parser for files with merge conflicts.

`class` `poetry_merge_lock.parser.`**`State`**`(`*value*`)`

Parser state for files with merge conflicts.

`class` `poetry_merge_lock.parser.`**`Token`**`(`*value*`)`

Token for parsing files with merge conflicts.

`exception` `poetry_merge_lock.parser.`**`UnexpectedTokenError`**`(`*token*`)`

The parser encountered an unexpected token.

Parameters **`token`** (`poetry_merge_lock.parser.Token`) –

Return type None

`poetry_merge_lock.parser.`**`parse`**`(`*lines*`)`

Parse a sequence of lines with merge conflicts.

Parameters **`lines`** (*`Sequence[str]`*) – The sequence of lines to be parsed.

Returns A pair of sequences of lines. The first sequence corresponds to *our* version, and the second, to *their* version.

Return type Tuple[Sequence[str], Sequence[str]]

`poetry_merge_lock.parser.`**`parse_line`**(*line*, *state*)
Parse a single line in a file with merge conflicts.

Parameters

- **`line`** (*`str`*) – The line to be parsed.
- **`state`** (`poetry_merge_lock.parser.State`) – The current parser state.

Returns A pair, consisting of the token for the line, and the new parser state.

Raises ***`UnexpectedTokenError`*** – The parser encountered an unexpected token.

Return type Tuple[*poetry_merge_lock.parser.Token*, *poetry_merge_lock.parser.State*]

`poetry_merge_lock.parser.`**`parse_lines`**(*lines*)
Parse a sequence of lines with merge conflicts.

Parameters **`lines`** (*`Sequence[str]`*) – The sequence of lines to be parsed.

Yields Pairs, where first item in each pair is a line in *our* version, and the second, in *their* version. An item is `None` if the line does not occur in that version.

Raises **`ValueError`** – A conflict marker was not terminated.

Return type Iterator[Tuple[Optional[str], Optional[str]]]

`poetry_merge_lock.parser.`**`tokenize`**(*line*)
Return the token for the line.

Parameters **`line`** (*`str`*) –

Return type *poetry_merge_lock.parser.Token*

1.4 poetry_merge_lock.mergetool

Merge tool for Poetry lock files at the TOML level.

`exception` `poetry_merge_lock.mergetool.`**`MergeConflictError`**(*keys*, *ours*, *theirs*)
An item in the TOML document cannot be merged.

Parameters

- **`keys`** (*`List[tomlkit.items.Key]`*) –
- **`ours`** (*`Any`*) –
- **`theirs`** (*`Any`*) –

Return type None

`poetry_merge_lock.mergetool.`**`merge`**(*value*, *other*)
Merge two versions of lock data.

This function returns a TOML document with the following merged entries:

- `package`
- `metadata.files`

Any other entries, e.g. `metadata.content-hash`, are omitted. They are generated from pyproject.toml when the lock data is written to disk.

Parameters

- **value** (*tomlkit.toml_document.TOMLDocument*) – Our version of the lock data.
- **other** (*tomlkit.toml_document.TOMLDocument*) – Their version of the lock data.

Returns The merged lock data.

Return type tomlkit.toml_document.TOMLDocument

poetry_merge_lock.mergetool.**merge_locked_package_files**(*value*, *other*)
Merge two TOML tables containing package files.

Parameters

- **value** (*tomlkit.items.Table*) – The package files in *our* version of the lock file.
- **other** (*tomlkit.items.Table*) – The package files in *their* version of the lock file.

Returns The package files obtained from merging both versions.

Raises ***MergeConflictError*** – The tables contain different files for the same package.

Return type tomlkit.items.Table

poetry_merge_lock.mergetool.**merge_locked_packages**(*value*, *other*)
Merge two TOML arrays containing locked packages.

Parameters

- **value** (*List[tomlkit.items.Table]*) – The packages in *our* version of the lock file.
- **other** (*List[tomlkit.items.Table]*) – The packages in *their* version of the lock file.

Returns The packages obtained from merging both versions.

Raises ***MergeConflictError*** – The lists contain different values for the same package.

Return type List[tomlkit.items.Table]

CHAPTER

TWO

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

2.1 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.

2.2 How to request a feature

Request features on the Issue Tracker.

2.3 How to set up your development environment

You need Python 3.6+ and the following tools:

- Poetry
- Nox

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 poetry-merge-lock
```

2.4 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.

2.5 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.

You can ensure that your changes adhere to the code style by reformatting with Black:

```
$ nox --session=black
```

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

THREE

CONTRIBUTOR COVENANT CODE OF CONDUCT

3.1 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.

3.2 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

3.3 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.

3.4 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.

3.5 Enforcement

Instances of abusive, harassing, or otherwise unacceptable behavior may be reported to the community leaders responsible for enforcement at mail@claudiojolowicz.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.

3.6 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:

3.6.1 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.

3.6.2 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.

3.6.3 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.

3.6.4 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.

3.7 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

FOUR

MIT LICENSE

This is a tool for resolving merge conflicts in the lock file of Poetry, a packaging and dependency manager for Python. If the merge conflicts cannot be resolved by this tool, you can use the *--print-content-hash* option to compute the content hash for the `metadata.content-hash` entry, and resolve the conflicts manually.

CHAPTER FIVE

INSTALLATION

To install poetry-merge-lock, run this command in your terminal:

```
$ pip install poetry-merge-lock
```

CHAPTER

SIX

USAGE

poetry-merge-lock's usage looks like:

```
$ poetry-merge-lock [OPTIONS]
```

`--print-content-hash`
Print the content hash (`metadata.content-hash`).

`--version`
Display the version and exit.

`--help`
Display a short usage message and exit.

PYTHON MODULE INDEX

INDEX