

---

# **poetry-merge-lock**

**Claudio Jolowicz**

**Apr 29, 2020**



# CONTENTS

<b>1</b>	<b>License</b>	<b>1</b>
<b>2</b>	<b>Reference</b>	<b>3</b>
<b>3</b>	<b>Installation</b>	<b>7</b>
<b>4</b>	<b>Usage</b>	<b>9</b>
	<b>Python Module Index</b>	<b>11</b>
	<b>Index</b>	<b>13</b>



**LICENSE****MIT License**

Copyright (c) 2020 Claudio Jolowicz

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.



## REFERENCE

- *poetry\_merge\_lock.console*
- *poetry\_merge\_lock.core*
- *poetry\_merge\_lock.parser*
- *poetry\_merge\_lock.mergetool*

## 2.1 poetry\_merge\_lock.console

Command-line interface.

## 2.2 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[Package]`) – The list of packages.

**Return type** `None`

`poetry_merge_lock.core.load (locker)`

Load a lock file with merge conflicts.

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

**Return type** `TOMLDocument`

**Returns** The merged TOML document.

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

Load the packages from a TOML document with lock data.

**Parameters**

- **locker** (Locker) – The locker object.
- **lock\_data** (TOMLDocument) – The lock data.

**Return type** List[Package]

**Returns** The list of packages.

`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` (Path) – Path to the lock file.

**Return type** Tuple[TOMLDocument, TOMLDocument]

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

`poetry_merge_lock.core.merge_lock(poetry)`

Resolve merge conflicts in Poetry's lock file.

**Return type** None

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

Validate the lock data and write it to disk.

**Parameters**

- **locker** (Locker) – The locker object.
- **lock\_data** (TOMLDocument) – The lock data.
- **root** (Package) – The root package of the Poetry project.

**Return type** None

## 2.3 poetry\_merge\_lock.parser

Line-based parser for files with merge conflicts.

**class** `poetry_merge_lock.parser.State`

Parser state for files with merge conflicts.

**class** `poetry_merge_lock.parser.Token`

Token for parsing files with merge conflicts.

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

The parser encountered an unexpected token.

`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.

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

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

`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** (*State*) – The current parser state.

**Return type** `Tuple[Token, State]`

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

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

`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.

**Return type** *Token*

## 2.4 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.

`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** (`TOMLDocument`) – Our version of the lock data.
- **other** (`TOMLDocument`) – Their version of the lock data.

**Return type** `TOMLDocument`

**Returns** The merged lock data.

`poetry_merge_lock.mergetool.merge_locked_package_files (value, other)`

Merge two TOML tables containing package files.

**Parameters**

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

**Return type** `Table`

**Returns** The package files obtained from merging both versions.

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

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

**Parameters**

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

**Return type** `List[Table]`

**Returns** The packages obtained from merging both versions.

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

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.

## INSTALLATION

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

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



## 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

### p

- `poetry_merge_lock.console`, 3
- `poetry_merge_lock.core`, 3
- `poetry_merge_lock.mergetool`, 5
- `poetry_merge_lock.parser`, 4



## Symbols

--help  
     command line option, 9  
 --print-content-hash  
     command line option, 9  
 --version  
     command line option, 9

## A

activate\_dependencies() (in module poetry\_merge\_lock.core), 3

## C

command line option  
     --help, 9  
     --print-content-hash, 9  
     --version, 9

## L

load() (in module poetry\_merge\_lock.core), 3  
 load\_packages() (in module poetry\_merge\_lock.core), 3  
 load\_toml\_versions() (in module poetry\_merge\_lock.core), 4

## M

merge() (in module poetry\_merge\_lock.mergetool), 5  
 merge\_lock() (in module poetry\_merge\_lock.core), 4  
 merge\_locked\_package\_files() (in module poetry\_merge\_lock.mergetool), 5  
 merge\_locked\_packages() (in module poetry\_merge\_lock.mergetool), 6  
 MergeConflictError, 5

## P

parse() (in module poetry\_merge\_lock.parser), 4  
 parse\_line() (in module poetry\_merge\_lock.parser), 4  
 parse\_lines() (in module poetry\_merge\_lock.parser), 5  
 poetry\_merge\_lock.console (module), 3  
 poetry\_merge\_lock.core (module), 3

poetry\_merge\_lock.mergetool (module), 5  
 poetry\_merge\_lock.parser (module), 4

## S

save() (in module poetry\_merge\_lock.core), 4  
 State (class in poetry\_merge\_lock.parser), 4

## T

Token (class in poetry\_merge\_lock.parser), 4  
 tokenize() (in module poetry\_merge\_lock.parser), 5

## U

UnexpectedTokenError, 4