

---

# Thymed

**Benjamin Crews**

**May 08, 2024**



# CONTENTS

<b>1</b>	<b>Features</b>	<b>3</b>
<b>2</b>	<b>Requirements</b>	<b>5</b>
<b>3</b>	<b>Installation</b>	<b>7</b>
<b>4</b>	<b>Usage</b>	<b>9</b>
<b>5</b>	<b>Contributing</b>	<b>11</b>
<b>6</b>	<b>License</b>	<b>13</b>
<b>7</b>	<b>Issues</b>	<b>15</b>
<b>8</b>	<b>Credits</b>	<b>17</b>
	<b>Python Module Index</b>	<b>27</b>
	<b>Index</b>	<b>29</b>







## **FEATURES**

- Flexible ChargeCode system with no limits on the number of codes you can define.
- Simple method to “punch” a ChargeCode. Punching logs the current timestamp and changes the “state” of the ChargeCode (active/passive or “on/off the clock”).
- All data is stored locally! It’s yours, you have complete control over it! There’s no online backups, no phoning home, no licensing.
- Command-Line Interface (CLI) for creating, listing, and punching in/out of charge codes.





## REQUIREMENTS

- No major requirements. If you have a Python version  $\geq 3.8$ , you're good to go! Check out the installation section below.
- Being familiar with the command-line is a plus. If the terminal scares you, this might not be the right tool for you.
- Thymed uses [Rich](#) for console markup. A modern terminal will make output much prettier! :wink:



## INSTALLATION

You can install *Thymed* via `pip` from PyPI:

```
$ pip install thymed
```



---

CHAPTER  
**FOUR**

---

**USAGE**

Please see the *Command-line Reference* for details.



## CONTRIBUTING

Contributions are very welcome. To learn more, see the *Contributor Guide*.





## LICENSE

Distributed under the terms of the [MIT license](#), *Thymed* is free and open source software.



## ISSUES

If you encounter any problems, please [file an issue](#) along with a detailed description.



## CREDITS

This project was originally generated from [@cjolowicz's Hypermodern Python Cookiecutter](#) template.

## 8.1 Usage

### 8.1.1 thymed

Thymed.

This command serves as the main entrypoint into the Thymed CLI. Subcommands exist for each specific action.

For more information, try: `thymed hello`

```
thymed [OPTIONS] COMMAND [ARGS]...
```

#### Options

##### **--version**

Show the version and exit.

#### **create**

Create a new ChargeCode.

Create a new ChargeCode object with the given id.

```
thymed create [OPTIONS]
```

#### **hello**

See information about Thymed.

```
thymed hello [OPTIONS]
```

### list

List out all the available charge codes.

```
thymed list [OPTIONS]
```

### punch

Punch a ChargeCode.

Punch the ChargeCode id provided.

If no id provided, grab the default code, and call its *punch* method.

Punches the current time. Write the data and the code, then exit.

```
thymed punch [OPTIONS] [ID]...
```

## Arguments

### ID

Optional argument(s)

### tui

Launch the TUI.

This command launches the Text User Interface (TUI). While the commandline functions are quite efficient, many users may prefer or need the TUI for interacting with Thymed. It's quite robust and polished! Give it a try!

```
thymed tui [OPTIONS]
```

## 8.2 Reference

### 8.2.1 thymed

Thymed.

This file contains basic functions and classes related to time-keeping.

**class** `thymed.ChargeCode`(*name*, *description*, *id*, *times*=<factory>)

A ChargeCode represents a type of work to dedicate hours to.

ChargeCodes have a name, description, identification number, and optional limits on time per week and total time dedicated.

#### Parameters

- **name** (*str*)
- **description** (*str*)
- **id** (*int*)

- **times** (*List[Tuple[datetime, datetime]]*)

**times**

times is the heart of the dataclass, it contains all the time code data in a list of tuples. For example: ::python

```
[(Datetime, Datetime), ... (Datetime, Datetime)]
```

**Type**

```
List[Tuple[datetime.datetime, datetime.datetime]]
```

**property is\_active:** [*<class 'bool'>*, *typing.Any*]

The charge code is active if it has been activated, but not closed.

**punch()**

Punch in/out of chargeable time.

**Return type**

```
None
```

**write\_class()**

Write the class to the Charges json file.

**Return type**

```
None
```

**write\_json**(*data=PosixPath('/home/docs/.thymed/thymed\_punches.dat')*, *log=False*)

Write the times data to a json file.

Read the file first, then append the times to their appropriate charge code number.

**Parameters**

- **data** (*Path*)
- **log** (*bool*)

**Return type**

```
None
```

**class thymed.TimeCard**(*id*)

A TimeCard collects work activity.

TimeCards take a single ChargeCode and collect all punch data for them. This enables filtering, reporting, and exporting data, but only for a single ChargeCode.

**Parameters**

**id** (*int*)

**general\_report**(*start, end*)

A general method to pull times data.

Specify a start and end date. This method will then form a pandas dataframe from the ChargeCode, and filter down for all times inclusively between the two dates.

Start and End can theoretically be any datetime object.

**Parameters**

- **start** (*datetime*)
- **end** (*datetime*)

**Return type***DataFrame***monthly\_report()**

Generates a report of all activity.

This method takes today's date, and returns a filtered set of punch data for the past 4 weeks.

**Return type***DataFrame***pay\_period\_report()**

Generates a report of all activity.

This method takes today's date, and returns a filtered set of punch data for the past 14 days.

**Return type***DataFrame***to\_excel(report, folder)**

This method just saves the report to the directory provided.

**Parameters**

- **report** (*DataFrame*)
- **folder** (*Path*)

**Return type***Path***weekly\_report()**

Generates a report of all activity.

This method takes today's date, and returns a filtered set of punch data for the past 7 days.

**Return type***DataFrame***thymed.get\_code(id)**

Read stored data and return the ChargeCode specified.

**Parameters****id** (*int*)**Return type***Any***thymed.object\_decoder(obj)**

Decoder hook for the ChargeCode class.

**Return type***Any*



## 8.3 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](#)*

### 8.3.1 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.

### 8.3.2 How to request a feature

Request features on the [Issue Tracker](#).

### 8.3.3 How to set up your development environment

You need Python 3.7+ 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 thymed
```

### 8.3.4 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.

### 8.3.5 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.

## 8.4 Contributor Covenant Code of Conduct

### 8.4.1 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, caste, color, 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.

## 8.4.2 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

## 8.4.3 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.

## 8.4.4 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.

## 8.4.5 Enforcement

Instances of abusive, harassing, or otherwise unacceptable behavior may be reported to the community leaders responsible for enforcement at [aceF22@gmail.com](mailto:aceF22@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.

## 8.4.6 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:

### 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.

### 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. 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.

### 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.

## 8.4.7 Attribution

This Code of Conduct is adapted from the Contributor Covenant, version 2.1, available at [https://www.contributor-covenant.org/version/2/1/code\\_of\\_conduct.html](https://www.contributor-covenant.org/version/2/1/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>.

## 8.5 License

### MIT License

Copyright © 2022 Benjamin Crews

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.



## PYTHON MODULE INDEX

### t

thymed, [18](#)





## INDEX

### Symbols

--version  
thymed command line option, 17

### C

ChargeCode (*class in thymed*), 18

### G

general\_report() (*thymed.TimeCard method*), 19  
get\_code() (*in module thymed*), 20

### I

ID  
thymed-punch command line option, 18  
is\_active (*thymed.ChargeCode property*), 19

### M

module  
thymed, 18  
monthly\_report() (*thymed.TimeCard method*), 20

### O

object\_decoder() (*in module thymed*), 20

### P

pay\_period\_report() (*thymed.TimeCard method*), 20  
punch() (*thymed.ChargeCode method*), 19

### T

thymed  
module, 18  
thymed command line option  
--version, 17  
thymed-punch command line option  
ID, 18  
TimeCard (*class in thymed*), 19  
times (*thymed.ChargeCode attribute*), 19  
to\_excel() (*thymed.TimeCard method*), 20

### W

weekly\_report() (*thymed.TimeCard method*), 20

write\_class() (*thymed.ChargeCode method*), 19

write\_json() (*thymed.ChargeCode method*), 19