

---

# **CellarTracker Documentation**

***Release 1.1.1***

**Mathieu Rul**

**Mar 07, 2022**



---

## Contents:

---

<b>1</b>	<b>CellarTracker</b>	<b>1</b>
1.1	Features . . . . .	1
1.2	Credits . . . . .	1
<b>2</b>	<b>Installation</b>	<b>3</b>
2.1	Stable release . . . . .	3
2.2	From sources . . . . .	3
<b>3</b>	<b>Usage</b>	<b>5</b>
<b>4</b>	<b>Contributing</b>	<b>7</b>
4.1	Types of Contributions . . . . .	7
4.2	Get Started! . . . . .	8
4.3	Pull Request Guidelines . . . . .	9
4.4	Tips . . . . .	9
4.5	Deploying . . . . .	9
<b>5</b>	<b>Credits</b>	<b>11</b>
5.1	Development Lead . . . . .	11
5.2	Contributors . . . . .	11
<b>6</b>	<b>History</b>	<b>13</b>
6.1	0.1.0 (2020-05-25) . . . . .	13
<b>7</b>	<b>Indices and tables</b>	<b>15</b>



# CHAPTER 1

---

## CellarTracker

---

Python package to export CellarTracker data.

- Free software: MIT license
- Documentation: <https://cellartracker.readthedocs.io>.

## 1.1 Features

Use the [CellarTracker export data](#) capabilities.

## 1.2 Credits

This package was created with [Cookiecutter](#) and the [audreyr/cookiecutter-pypackage](#) project template.



### 2.1 Stable release

To install CellarTracker, run this command in your terminal:

```
$ pip install cellartracker
```

This is the preferred method to install CellarTracker, as it will always install the most recent stable release.

If you don't have [pip](#) installed, this [Python installation guide](#) can guide you through the process.

### 2.2 From sources

The sources for CellarTracker can be downloaded from the [Github repo](#).

You can either clone the public repository:

```
$ git clone git://github.com/mathroule/cellartracker
```

Or download the [tarball](#):

```
$ curl -OJL https://github.com/mathroule/cellartracker/tarball/master
```

Once you have a copy of the source, you can install it with:

```
$ python setup.py install
```





## CHAPTER 3

---

### Usage

---

To use CellarTracker from the console:

```
$ cellartracker [-h] -u USERNAME -p PASSWORD
                [-t {List,Inventory,Notes,PrivateNotes,Purchase,Pending,Consumed,
↪Availability,Tag,ProReview,Bottles,FoodTag}]
                [-f {html,xml,tab,csv}]
```

To use CellarTracker in a project:

```
from cellartracker import cellartracker

client = cellartracker.CellarTracker(username, password)
client.get_list()           # Return List
client.get_inventory()      # Return Inventory
client.get_notes()          # Return Notes
client.get_private_notes()  # Return PrivateNotes
client.get_purchase()       # Return Purchase
client.get_pending()        # Return Pending
client.get_consumed()       # Return Consumed
client.get_availability()   # Return Availability
client.get_tag()            # Return Tag
client.get_pro_review()     # Return ProReview
client.get_bottles()        # Return Bottles
client.get_food_tag()       # Return FoodTag
```



Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given. You can contribute in many ways:

## 4.1 Types of Contributions

### 4.1.1 Report Bugs

Report bugs at <https://github.com/mathroule/cellartracker/issues>.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

### 4.1.2 Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with “bug” and “help wanted” is open to whoever wants to implement it.

### 4.1.3 Implement Features

Look through the GitHub issues for features. Anything tagged with “enhancement” and “help wanted” is open to whoever wants to implement it.

## 4.1.4 Write Documentation

CellarTracker could always use more documentation, whether as part of the official CellarTracker docs, in docstrings, or even on the web in blog posts, articles, and such.

## 4.1.5 Submit Feedback

The best way to send feedback is to file an issue at <https://github.com/mathroule/cellartracker/issues>.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

## 4.2 Get Started!

Ready to contribute? Here's how to set up *cellartracker* for local development.

1. Fork the *cellartracker* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/cellartracker.git
```

3. Install your local copy into a virtualenv. Assuming you have virtualenvwrapper installed, this is how you set up your fork for local development:

```
$ mkvirtualenv cellartracker
$ cd cellartracker/
$ python setup.py develop
```

4. Create a branch for local development:

```
$ git checkout -b name-of-your-bugfix-or-feature
```

Now you can make your changes locally.

5. When you're done making changes, check that your changes pass flake8 and the tests, including testing other Python versions with tox:

```
$ flake8 cellartracker tests
$ python setup.py test or pytest
$ tox
```

To get flake8 and tox, just pip install them into your virtualenv.

6. Commit your changes and push your branch to GitHub:

```
$ git add .
$ git commit -m "Your detailed description of your changes."
$ git push origin name-of-your-bugfix-or-feature
```

7. Submit a pull request through the GitHub website.

## 4.3 Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.
2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in README.rst.
3. The pull request should work for Python 3.5, 3.6, 3.7 and 3.8, and for PyPy. Check [https://travis-ci.com/mathroule/cellartracker/pull\\_requests](https://travis-ci.com/mathroule/cellartracker/pull_requests) and make sure that the tests pass for all supported Python versions.

## 4.4 Tips

To run a subset of tests:

```
$ python -m unittest tests.test_cellartracker
```

## 4.5 Deploying

A reminder for the maintainers on how to deploy. Make sure all your changes are committed (including an entry in HISTORY.rst). Then run:

```
$ bump2version patch # possible: major / minor / patch
$ git push
$ git push --tags
```

Travis will then deploy to PyPI if tests pass.



### 5.1 Development Lead

- Mathieu Rul <mathroule@gmail.com>

### 5.2 Contributors

None yet. Why not be the first?





#### 6.1 0.1.0 (2020-05-25)

- First release on PyPI.



## CHAPTER 7

---

### Indices and tables

---

- `genindex`
- `modindex`
- `search`