Synthale Documentation

Release 0.1.1

Mike Shoup

Jan 04, 2019

Contents:

1 Installing 3

2 Using Synthale 5

3 Code 7

4 Contributing 11

5 License 15

6 Open Source 17

7 Indices and tables 19

Python Module Index 21

A python application to convert valid BeerXML files into Markdown files. You can then use those Markdown files in your favorite static website generator.

CHAPTER 1

Installing

First thing's first, you need Python 3.5 or newer. Python 2 is not supported.

Then, you can simply use pip to install Synthale:

```
pip install synthale
```

If that doesn't work, you may need to put *sudo* in front of your command.

Some things to note, many (all?) Linux distributions still default to Python 2. Python 2 is not supported. Ensure your *pip* command is using a new enough version of Python by running *pip –version*:

```
$ pip --version
pip 9.0.1 from /usr/lib/python2.7/dist-packages (python 2.7)
```

This *pip* is installed using Python 2.7. If your *pip* is using Python 2, try running the *pip3* command instead:

```
$ pip3 --version
pip 9.0.1 from /usr/local/lib/python3.5/site-packages (python 3.5)
$ pip3 install synthale
```

As long as *pip* succesfully installs Synthale, you can simply type the *synthale* command to make sure it's installed:

```
$ synthale
Usage: synthale [OPTIONS] COMMAND [ARGS]...

  Synthale converts BeerXML files to markdown.

  Copyright (C) 2019 Mike Shoup

Options:
  --help  Show this message and exit.

Commands:
  generate  Generate markdown files from BeerXML files.
  version   Print version and exit.
```

Now, move onto *Using Synthale*

CHAPTER 2

Using Synthale

2.1 synthale

Synthale converts BeerXML files to markdown.

```
synthale [OPTIONS] COMMAND [ARGS]...
```

2.1.1 generate

Generate markdown files from BeerXML files.

INPUT_PATH is either a directory containing XML files, or an individual XML file. OUTPUT_PATH is the directory to write the markdown files to.

```
synthale generate [OPTIONS] INPUT_PATH OUTPUT_PATH
```

Options

-v, --vol-unit <vol_unit>
Unit to display volumes in. Default is gallons.

-H, --hop-unit <hop_unit>
Unit to display hop masses in. Default is ounces.

-f, --fermentable-unit <fermentable_unit>
Unit to display fermentable masses in. Default is pounds.

-f, --temp-unit <temp_unit>
Unit to display temperatures in. Default is fahrenheit.

Arguments

INPUT_PATH
Required argument

OUTPUT_PATH
Required argument

2.1.2 version

Print version and exit.

```
synthale version [OPTIONS]
```

CHAPTER 3

Code

3.1 synthale package

3.1.1 Submodules

3.1.2 synthale.cli module

Contains the commands for the CLI interface.

3.1.3 synthale.convert module

Contains functions used to convert units.

Input units for volumes is liters. Input units for mass is kilograms. These units are the defaults used in the BeerXML format, and as a result, in *pybeerxml*.

Functions in this module accept a format_spec parameter. The format_spec parameter is a string conforming to the Python "Format Specification Mini Language".

See format_spec documentation.

`synthale.convert.`**`celsius`**(*celsius*, *format_spec=''*)
Return a string with the unit appended.

`synthale.convert.`**`fahrenheit`**(*celsius*, *format_spec=''*)
Convert celsius to fahrenheit, return a string with unit appended.

`synthale.convert.`**`gallons`**(*liters*, *format_spec=''*)
Convert liters to gallons and return a string with the unit appended.

`synthale.convert.`**`grams`**(*kilograms*, *format_spec=''*)
Convert kilograms to grams, return a string with the unit appended.

`synthale.convert.`**`kilograms`**(*kilograms*, *format_spec=''*)
Return a string with the unit appended.

synthale.convert.**liters**(*liters*, *format_spec=''*)
Return a string with the unit appended.

synthale.convert.**ounces**(*kilograms*, *format_spec=''*)
Convert kilograms to ounces, return a string with the unit appended.

synthale.convert.**pounds**(*kilograms*, *format_spec=''*)
Convert kilograms to pounds, return a string with the unit appended.

3.1.4 synthale.markdown module

This module contains functions to generate markdown elements.

synthale.markdown.**emphasis**(*text*)
Wrap text with asterisks.

synthale.markdown.**setext_heading**(*text*, *level*)
Return an setext heading.

text is the text to include in the heading. Leading and trailing whitespace is trimmed from *text*. If *level* is the number *1*, a first level (h1) heading is returned. If *level* is the number *2* (or anything else), a second level (h2) heading is returned.

See https://github.github.com/gfm/#setext-heading for more information.

synthale.markdown.**strong**(*text*)
Wrap text with double asterisks.

synthale.markdown.**table**(*headers*, *rows*)
Generate a table.

headers is a list/tuple of header cells. *rows* is a list of lists containing each cell. If any row has more cells than there are headers, the extra cells are silently dropped.

See https://github.github.com/gfm/#tables-extension- for syntax of a GFM table.

3.1.5 synthale.recipes module

Use this module to parse BeerXML files.

class synthale.recipes.**MarkdownRecipe**(*recipe*, *vol_unit='gallons'*, *hop_unit='ounces'*, *fermentable_unit='pounds'*, *temp_unit='fahrenheit'*)
Bases: object

A recipe in markdown form.

details
Return markdown for the recipe's details.

fermentables
Return markdown to represent the recipe's fermentables.

filename
Return the filename for the recipe.

Converts the recipe name to lowercase and replaces all non-word characters with an underscore. Trailing underscores are removed. *.md* is appended to the name.

hops
Return markdown to represent the recipe's hops.

markdown
Return generated markdown for the recipe.

mash
Return the markdown to represent the recipe's mash steps.

miscs
Return the markdown to represent the recipe's other ingredients.

name
Return markdown for the recipe's name.

style
Return markdown for the recipe's style.

yeast
Return markdown to represent the recipe's yeast.

synthale.recipes.**load_all_files**(*path*, *units={}*)
Parse all XML files in a directory.

path is a path to a directory with .xml files. *units* is a dictionary defining the units used. *units* will be unpacked and used during the creation of a *MarkdownRecipe* object.

Returns a list of MarkdownRecipe objects.

synthale.recipes.**load_file**(*path*, *units={}*)
Parse BeerXML file.

path is the path to a BeerXML file. *units* is a dictionary defining the units used. *units* will be unpacked and used during the creation of a *MarkdownRecipe* object.

Return a list of MarkdownRecipe objects. If an exception is raised during parsing, the message is printed to stderr and an empty list is returned.

synthale.recipes.**write_recipes**(*recipes*, *output_path*)
Write *recipes* to *output_path*.

recipes is a list of MarkdownRecipe objects. *output_path* is a directory to write the recipes to.

3.1.6 Module contents

Synthale package.

CHAPTER 4

Contributing

Synthale is a little project of mine that I hope others will use. As a user and fan of open source software, I welcome any contributions you have!

This guide is intended to help you contribute to this project.

4.1 Issues

Please open a GitHub Issue for any bugs, support requests, or feature requests you have. For bugs and support requests, please describe what you were doing when you encountered the issue, and include any sample BeerXML recipes.

If you wish to contribute a feature, or have a request for a feature, pleasea also open a new GitHub Issue.

4.2 Pull Requests

I welcome all pull requests! Expect a back and forth conversation while we make sure your contribution is the best it can be. If you plan to spend significant time contributing a new feature, pelase open a GitHub Issue first to ensure it is something that will be accepted.

I ask the following of all pull requests:

1. Your code must be written for the versions of Python supported. Currently, Synthale supports Python 3.5 and newer.
2. Your code should pass all the *unit tests*.
3. New code should not decrease the coverage of the *unit tests*. This means may need to write new *unit tests* before your pull request can be merged.
4. Your code should follow the same *style guidelines* as the rest of the project.
5. You agree to release your code under the GNU GPLv3 license. You do not transfer your copyright to me, and you own your code. Please add your copyright statement to the top of the file. (If more people than just me contribute, I may move the copyright notices to a separate file.)

4.3 Developing

Create a virtual environment with Python 3.5 or newer. Clone the GitHub Repository and install synthale w/ *pip*, and install the required tools for testing:

```
pip install -e .
pip install -r test-requirements.txt
```

4.4 Unit Tests

All code should be covered by automated unit tests. The tools *pytest* and *coverage* are used, and tests are located in the *tests/* folder.

Before submitting your code, run the unit tests. You can do it one of two ways:

On your computer

See *Developing* for setting up your development environment. Run the tests:

```
coverage run -m pytest
coverage report -m
```

Ensure your new code is included in the tests.

Using cirlceci

You can add your forked repo to CircleCI and run the tests when you push to your repo.

4.5 Style Guidelines

In general, code should conform to PEP 8.

You can confirm that your code generally follows my preferred style by running the *flake8* command, after you've installed the test requirements.

The following are things that PEP 8 either doesn't talk about, or are ambiguous:

Indentation

Indent with 4 spaces. Don't use tabs.

Line length

Lines should not be longer than 79 characters. I frequently like to put two code files side by side on one monitor, and 79 characters is the perfect cutoff.

String formatting

I prefer to wrap strings in 'single quotes' instead of "double quotes". The exception is if a string will have an apostrophe or single quote inside, and it will look better to use double quotes.

Do this:

```
'Just a plain old string.'
"You're an aweomse contributor."
```

Don't do this:

```
"Double quotes for plain strings."
'Single quotes where you\'ll use an apostrophe.'
```

I prefer *str.format* over formatting strings with the % operator.

Do this:

```
"{} is the loneliest number that you'll ever do.".format(1)
```

Don't do this:

```
'%d can be as bad as %d' % (2, 1)
```

Docstrings

Everything should have a docstring, and the *flake8-docstrings* package will help make sure the docstring is well-formatted. The docstrings should describe the class/method/function/whatever. Docstrings should explain the parameters, and what is returned.

CHAPTER 5

License

5.1 Documentation

Documentation is released under the CC BY-SA 4.0 license.

5.2 Source Code

The Synthale source code is released under the GNU GPLv3 license.

CHAPTER 6

Open Source

Synthale is free and open source! Get the source code at the GitHub repository.

CHAPTER 7

Indices and tables

- genindex
- modindex
- search

Python Module Index

S

synthale, 9
synthale.cli, 7
synthale.convert, 7
synthale.markdown, 8
synthale.recipes, 8

Index

Symbols

-H, –hop-unit <hop_unit>
    synthale-generate command line option, 5
-f, –fermentable-unit <fermentable_unit>
    synthale-generate command line option, 5
-f, –temp-unit <temp_unit>
    synthale-generate command line option, 5
-v, –vol-unit <vol_unit>
    synthale-generate command line option, 5

C

celsius() (in module synthale.convert), 7

D

details (synthale.recipes.MarkdownRecipe attribute), 8

E

emphasis() (in module synthale.markdown), 8

F

fahrenheit() (in module synthale.convert), 7
fermentables (synthale.recipes.MarkdownRecipe attribute), 8
filename (synthale.recipes.MarkdownRecipe attribute), 8

G

gallons() (in module synthale.convert), 7
grams() (in module synthale.convert), 7

H

hops (synthale.recipes.MarkdownRecipe attribute), 8

I

INPUT_PATH
    synthale-generate command line option, 6

K

kilograms() (in module synthale.convert), 7

L

liters() (in module synthale.convert), 7
load_all_files() (in module synthale.recipes), 9
load_file() (in module synthale.recipes), 9

M

markdown (synthale.recipes.MarkdownRecipe attribute), 8
MarkdownRecipe (class in synthale.recipes), 8
mash (synthale.recipes.MarkdownRecipe attribute), 9
miscs (synthale.recipes.MarkdownRecipe attribute), 9

N

name (synthale.recipes.MarkdownRecipe attribute), 9

O

ounces() (in module synthale.convert), 8
OUTPUT_PATH
    synthale-generate command line option, 6

P

pounds() (in module synthale.convert), 8

S

setext_heading() (in module synthale.markdown), 8
strong() (in module synthale.markdown), 8
style (synthale.recipes.MarkdownRecipe attribute), 9
synthale (module), 9
synthale-generate command line option
    -H, –hop-unit <hop_unit>, 5
    -f, –fermentable-unit <fermentable_unit>, 5
    -f, –temp-unit <temp_unit>, 5
    -v, –vol-unit <vol_unit>, 5
    INPUT_PATH, 6
    OUTPUT_PATH, 6
synthale.cli (module), 7
synthale.convert (module), 7
synthale.markdown (module), 8
synthale.recipes (module), 8

T

table() (in module synthale.markdown), 8

W

write_recipes() (in module synthale.recipes), 9

Y

yeast (synthale.recipes.MarkdownRecipe attribute), 9