
brew-tools Documentation

Release unknown

Sven Steinbauer

Jan 12, 2023

Contents

1	Contents	3
1.1	Features	3
1.2	Install Brew Tools	5
1.3	License	5
1.4	Contributors	6
1.5	Changelog	6
1.6	brew_tools	7
2	Indices and tables	13
	Python Module Index	15
	Index	17

This is the documentation for

—

Welcome to the documentation for Brew-Tools, the CLI toolset for homebrewers

Brew-Tools is a small commandline utility that offers quick access to a set of calculators and tools to help homebrewers create their brews.

Granted, the CLI is not everyone's favourite interface, and this is by no means intended to replace other GUI based tools.

Its aim is to provide simple and quick access to tools that are usually available in a larger piece of software with all the bells and whistles. Instead of having to click around a desktop app, or wait for web pages to load, you can complete these tasks very quickly with Brew-tools.

For example to calculate the amount of priming sugar needed

```
$> brew-tools prime
Volume of beer to prime (liter): 19
Desired volumes of CO2: 2.3
Temperature of beer (C): 15

Use only one of the following:
Table sugar: 98.50g
Corn Sugar: 108.29g
DME: 144.84g
```

All values can also be passed in as arguments directly

```
$> brew-tools prime -beer 19 -vol 2.3 -temp 15

Use only one of the following:
Table sugar: 98.50g
Corn Sugar: 108.29g
DME: 144.84g
```

It is written in Python 3 and has minimal dependencies on external packages.

Brew-tools is opensource and contributions and suggestions are welcomed.

Note: All values and calculations are provided as guidelines only. Brew-tools should not be used for professional brewing. No warranty or guarantee of accuracy is provided on the information provided by this calculator.

1.1 Features

Brew Tools comes with the following tools (from the help)

```
Usage: brew_tools [OPTIONS] COMMAND [ARGS]...
```

Brew-Tools **is** a small commandline utility that offers quick access to a **set** of calculators **and** tools to help homebrewers create their brews.

All values **and** calculations are provided **as** guidelines only. Brew-tools should **not** be used **for** professional brewing. No warranty **or** guarantee of accuracy **is** provided on the information provided by this calculator.

Options:

```
--version  Show the version and exit.
--unit [metric|imperial]  Ignore config and use a different unit.
--help      Show this message and exit.
```

Commands:

abv	Calculates the ABV from the original and final gravity...
adjust-gravity	Calculate the amount of liquid to boil off/dilute with to...
adjust-volume	Calculate the new gravity after a change in wort volume...
attenuation	Calculates the apparent and real attenuation from the...
convert	Convert a value between given measurements.
dme	Given the current volume of the mash, work out how much...
fg-from-att	Given a starting gravity and a desired attenuation level,...
infuse	Given the current mash temperature, work out how much...
kegpsi	Calculates the regulator pressure required to achieve...
prime	Calculates the amount of table sugar, corn sugar, or DME...

<<<<<<< **Updated upstream** strike Calculate the required strike water temperature given...

The full command descriptions are below

1.1.1 abv

Calculates the ABV from the original and final gravity readings. By default the wort and alcohol correction factor is not applied. If you are using a hydrometer add the `adjust` flag to automatically correct the final gravity.

1.1.2 adjust-gravity

Calculate the amount of liquid to boil off/dilute with to achieve a desired gravity.

1.1.3 adjust-volume

Calculate the new gravity after a change in wort volume either through dilution or boil off

1.1.4 attenuation

Calculates the apparent and real attenuation from the provided original and final/current gravity. Real attenuation is the adjusted value taking into account the alcohol in the beer

1.1.5 convert

Convert a value between given measurements. Supported types are:

mass, volume, gravity, colour

1.1.6 dme

Given the current volume of the mash, work out how much Dry Malt Extract(DME) to add to reach your target gravity

1.1.7 fg-from-att

Given a starting gravity and a desired attenuation level, will return the specific gravity for that percentage of attenuation. Useful if you have to action something at a given attenuation point and need to know what the gravity is when that point is reached

1.1.8 infuse

Given the current mash temperature, work out how much water of a given temp needs to be added to adjust the temperature

1.1.9 kegpsi

Calculates the regulator pressure required to achieve desired CO2 volumes.

1.1.10 prime

Calculates the amount of table sugar, corn sugar, or DME needed to achieve the requested CO2 volumes for bottle priming

1.1.11 strike

Calculate the strike water temperature given the mass of grain, volume of water, and desired mash temperature

Using brew-tools in your own project

All these tools are available to use in your own Python application by importing the `brew_maths` module into your code

```
import brew_maths from brew_tools

new_gravity = brew_maths.adjust_gravity(1.050, 1.020)
```

Not that the `brew_maths` module does not do any bounds checking on the values passed. It is up to the calling code to ensure that the values are within valid bounds if needed

1.2 Install Brew Tools

Use pip

```
pip install brew-tools
```

then run with *brew_tools*

If you are planning to work on it, you can use `poetry` to install it once you've cloned the project

```
$> git clone git@github.com:Svenito/brew-tools.git
$> cd brew-tools
$> poetry shell
$> poetry install
```

Alternatively you can use the virtualenv and local install method too.

1.3 License

The MIT License (MIT)

Copyright (c) 2018 Sven Steinbauer

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.

1.4 Contributors

- Sven Steinbauer <<https://github.com/Svenito>>
- SlayterDev <<https://github.com/SlayterDev>>
- Szczyp <<https://github.com/Szczyp>>

1.5 Changelog

1.5.1 Version 0.2.9

- Adds user config for default unit type (metric or imperial)

1.5.2 Version 0.2.8

- Add a strike water temperature calculator

1.5.3 Version 0.2.7

- Add a simple unit converter for mass, volume, gravity, and colour

1.5.4 Version 0.2.6

- Only a change to the CI config to make poetry work

1.5.5 Version 0.2.5

- Update install docs to reflect poetry changes

1.5.6 Version 0.2.3

- Move project to pyproject.toml and poetry build/release system

1.5.7 Version 0.2.1

- Fix error converting Fahrenheit to Celcius

1.5.8 Version 0.2.0

- Make wort and alcohol correction factor for abv calculation optional

1.5.9 Version 0.1.0

- Bump to 0.1.0
- Update commandline help

1.5.10 Version 0.0.8

- Add adjust gravity by volume calculator
- Add new gravity by volume change calculator
- Refactor prompt to use loop instead of recursion
- Fix various typos

1.5.11 Version 0.0.7

- Add attenuation calculators
- Rename utils module to inputs and refactor input functions into into

1.5.12 Version 0.0.6

- Add sg to plato function
- Refactor volume and gravity prompts
- Add disclaimer to README and docs

1.5.13 Version 0.0.5

- Added gravity adjustment calculator

1.5.14 Version 0.0.4

- Fixes to testing and release process

1.5.15 Version 0.0.1

- Initial release

1.6 brew_tools

1.6.1 brew_tools package

Submodules

brew_tools.brew_maths module

`brew_tools.brew_maths.abv(og: float, fg: float, adjust: bool) → float`

Calculate the ABV from the given `og` and `fg`. Will automatically adjust the `fg` for wort correction and alcohol

Parameters

- **og** – The original gravity
- **fg** – The final gravity

Returns The ABV value

`brew_tools.brew_maths.adjust_gravity(og: float, fg: float) → float`

Adjust final gravity for wort correction and alcohol

Parameters

- **og** – original gravity as specific gravity
- **fg** – final gravity as specific gravity

Returns adjusted specific gravity value

`brew_tools.brew_maths.adjust_gravity_volume(vol: float, og: float, ng: float) → float`

Returns the new volume needed to achieve the desired new gravity. This is unit independent and the return value can be used for liters and or gallons.

New Volume = (Volume * original Gravity) / new Gravity

Parameters

- **vol** – Original volume of wort
- **og** – The current gravity of the wort
- **ng** – The desired gravity of the wort

Returns The amount to adjust the wort volume by

`brew_tools.brew_maths.adjust_volume_gravity(vol: float, og: float, new_vol: float) → float`

Calculate the new gravity after boil off or dilution to `new_vol` This is unit independent and the volume can be used for liters and or gallons.

Ending Gravity = (Beginning Volume * Beginning Gravity) / End Volume

Parameters

- **vol** – Original volume of wort
- **og** – The current gravity of the wort
- **new_vol** – The new volume of the wort

Returns The new gravity after boiloff or dilution

`brew_tools.brew_maths.apparent_attenuation(og: float, fg: float) → float`

Calculate the apparent attenuation from the current and original gravity. via <http://realbeer.com/spencer/attenuation.html>

$AA = 1 - AE / OE$

Parameters

- **og** – The original gravity of the wort (1.0 to 1.2)
- **fg** – The current gravity of the beer

Returns The apparent attenuation as a decimal (multiply by 100 to get percentage value)

`brew_tools.brew_math.c_to_f(c: float) → float`
Convert celcius to fahrenheit

`brew_tools.brew_math.ebc_to_l(ebc: float) → float`
Convert EBC to Lovibond https://en.wikipedia.org/wiki/Standard_Reference_Method

`brew_tools.brew_math.ebc_to_srm(ebc: float) → float`
Convert the EBC value to SRM https://en.wikipedia.org/wiki/Standard_Reference_Method

`brew_tools.brew_math.f_to_c(f: float) → float`
Convert fahrenheit to celcius

`brew_tools.brew_math.fg_from_attenuation(og: float, attenuation: float) → float`
Calculates the gravity when the beer has reached a given attenuation percentage from the original gravity. Simply an inverse solve of `apparent_attenuation`

Parameters

- **og** – The original gravity of the wort as specific gravity
- **attenuation** – The percentage attenuation to achieve

Returns The gravity when the requested attenuation has been reached

`brew_tools.brew_math.g_to_l(gallon: float) → float`
Convert US gallons to liters

`brew_tools.brew_math.g_to_oz(g: float) → float`
Convert grams to ounces

`brew_tools.brew_math.infusion(ratio: float, curr_temp: float, new_temp: float, water_temp: float, grain: float) → float`

Calculate the amount of hot water required to raise the mash temperature to a specific temperature.

From: <http://howtobrew.com/book/section-3/the-methods-of-mashing/calculations-for-boiling-water-additions>

$$W_a = (T_2 - T_1)(.2G + W_m)/(T_w - T_2)$$

Parameters

- **ratio** – Grist ratio in quarts/lbs
- **curr_temp** – Current mash temperature in fahrenheit
- **new_temp** – The target temperature of the mash in fahrenheit
- **water_temp** – The temperature of the water to be added in fahrenheit
- **grain** – The dry weight of the grain in the mash in pounds

Returns The amount of water at given temperature to add to achieve requested change in mash temperature

`brew_tools.brew_math.keg_psi(temp: float, co2: float) → float`
Calculate require keg pressure to carbonate liquid at temp with co2 volumes of CO2

From http://www.wetnewf.org/pdfs/Brewing_articles/CO2%20Volumes.pdf

$$V = (P + 14.695) * (0.01821 + 0.09011 * \text{EXP}(-(T-32)/43.11)) - 0.003342$$

Parameters

- **temp** – Temperature of liquid in keg in fahrenheit
- **co2** – Volume of CO2 required

Returns The PSI value to set the regulator to

`brew_tools.brew_math.kg_to_lbs(kg: float) → float`
 Convert kilograms to pounds

`brew_tools.brew_math.l_to_ebc(lovibond: float) → float`
 Convert from Lovibond to EBC https://en.wikipedia.org/wiki/Standard_Reference_Method

`brew_tools.brew_math.l_to_g(liter: float) → float`
 Convert liters to gallons US

`brew_tools.brew_math.l_to_q(liter: float) → float`
 Convert liters to quarts US

`brew_tools.brew_math.l_to_srm(lovibond: float) → float`
 Convert from Lovibond to EBC https://en.wikipedia.org/wiki/Standard_Reference_Method

`brew_tools.brew_math.lbs_to_kg(lbs: float) → float`
 Convert kilograms to pounds

`brew_tools.brew_math.lbs_to_oz(lbs: float) → float`
 Convert lbs to ounces

`brew_tools.brew_math.oz_to_g(oz: float) → float`
 Convert ounces to grams

`brew_tools.brew_math.pre_boil_dme(points: float, cur_vol: float) → float`
 Calculate the amount of DME needed to raise the gravity of a given volume of wort by a given number or gravity points. Assumes DME has an extract of 1.044ppg.

Parameters

- **points** – Number of gravity points to raise
- **cur_vol** – The current volume of the wort in gallons.

Returns The amount of DME to add to raise the gravity

`brew_tools.brew_math.priming(temp: float, beer_vol: float, co2: float) → float`
 Calculate the required weight priming (table) sugar for a given volume of beer at a specific temperature for desired CO2 volume. Beer temperature should be the temperature that the beer has been at the longest.

From: <http://www.straighttothepint.com/priming-sugar-calculator/>

$PS = 15.195 * V_{beer} * (V_{CO2} - 3.0378 + (0.050062 * T_{ferm}) - (0.00026555 * (T_{ferm}^2)))$

Parameters

- **temp** – Temperature of beer in fahrenheit
- **beer_vol** – Volume of beer to prime in gallons US
- **co2** – The volume of CO2 required

Returns The amount table sugar required

`brew_tools.brew_math.real_attenuation(og: float, fg: float) → float`
 Calculate the real attenuation from the original and current gravity. Takes into account the alcohol in the beer. Calculates the real extract and uses that to calculate the attenuation via <http://realbeer.com/spencer/attenuation.html>

$RE = .1808 * OE + .8192 * AE$ $RA = 1 - RE / OE$

or

$$RA = 1 - (.1808 * OE + .8192 * AE) / OE$$

Parameters

- **og** – The original gravity of the wort (1.0 to 1.2)
- **fg** – The current gravity of the beer

Returns The real attenuation as a decimal (multiply by 100 to get percentage value)

`brew_tools.brew_math.srm_to_ebc(srm: float) → float`

Convert the EBC value to SRM https://en.wikipedia.org/wiki/Standard_Reference_Method

`brew_tools.brew_math.srm_to_l(srm: float) → float`

Convert the SRM value to Lovibond https://en.wikipedia.org/wiki/Standard_Reference_Method

`brew_tools.brew_math.strike_temp(grain: float, vol: float, temp: float) → float`

W = Strike water temperature °F (?) R = Water to grist ratio in quarts/lb (40 quarts/14 lbs = 2.857) T1 = Temp. of your dry grain °F (70) T2 = Desired mash temp °F (156 – adjusted for thermal loss)

$$W = (.2/R)(T2-T1)+T2$$

`brew_tools.brew_math.to_brix(value: float) → float`

Convert gravity value to brix value

`brew_tools.brew_math.to_plato(sg: float) → float`

Convert specific gravity to plato (extract)

$$(-1 * 616.868) + (1111.14 * sg) - (630.272 * sg^2) + (135.997 * sg^3)$$

`brew_tools.brew_math.to_sg(plato: float) → float`

Convert from plato to specific gravity

brew_tools.command_line module**brew_tools.config module****brew_tools.converter module**

`brew_tools.converter.print_colour(value: float) → None`

`brew_tools.converter.print_gravity(value: float) → None`

`brew_tools.converter.print_mass(value: float) → None`

`brew_tools.converter.print_volume(value: float) → None`

brew_tools.inputs module

`brew_tools.inputs.between(min: float, max: float) → Callable`

Returns a function to test if a value lies between min and max

`brew_tools.inputs.get_choice(prompt: str, choices: List[str]) → int`

`brew_tools.inputs.get_gravity_input(prompt: str) → float`

Prompt for an input for gravity and validated to be between 1.0 and 1.2

Parameters

- **ctx** – Click context
- **prompt** – User prompt. Will be checked for bounds

Returns entered value as float

`brew_tools.inputs.get_unit_input (unit: str, prompt: str) → float`

Prompt for an input for temperature and automatically resolve unit (Celcius or Fahrenheit)

Parameters

- **unit** – unit to use
- **prompt** – User prompt. Correct unit will be appended

Returns entered value as float

Module contents

CHAPTER 2

Indices and tables

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

b

`brew_tools`, [12](#)

`brew_tools.brew_maths`, [8](#)

`brew_tools.converter`, [11](#)

`brew_tools.inputs`, [11](#)

A

abv() (in module *brew_tools.brew_maths*), 8
adjust_gravity() (in module *brew_tools.brew_maths*), 8
adjust_gravity_volume() (in module *brew_tools.brew_maths*), 8
adjust_volume_gravity() (in module *brew_tools.brew_maths*), 8
apparent_attenuation() (in module *brew_tools.brew_maths*), 8

B

between() (in module *brew_tools.inputs*), 11
brew_tools (module), 12
brew_tools.brew_maths (module), 8
brew_tools.converter (module), 11
brew_tools.inputs (module), 11

C

c_to_f() (in module *brew_tools.brew_maths*), 9

E

ebc_to_l() (in module *brew_tools.brew_maths*), 9
ebc_to_srm() (in module *brew_tools.brew_maths*), 9

F

f_to_c() (in module *brew_tools.brew_maths*), 9
fg_from_attenuation() (in module *brew_tools.brew_maths*), 9

G

g_to_l() (in module *brew_tools.brew_maths*), 9
g_to_oz() (in module *brew_tools.brew_maths*), 9
get_choice() (in module *brew_tools.inputs*), 11
get_gravity_input() (in module *brew_tools.inputs*), 11
get_unit_input() (in module *brew_tools.inputs*), 12

I

infusion() (in module *brew_tools.brew_maths*), 9

K

keg_psi() (in module *brew_tools.brew_maths*), 9
kg_to_lbs() (in module *brew_tools.brew_maths*), 10

L

l_to_ebc() (in module *brew_tools.brew_maths*), 10
l_to_g() (in module *brew_tools.brew_maths*), 10
l_to_q() (in module *brew_tools.brew_maths*), 10
l_to_srm() (in module *brew_tools.brew_maths*), 10
lbs_to_kg() (in module *brew_tools.brew_maths*), 10
lbs_to_oz() (in module *brew_tools.brew_maths*), 10

O

oz_to_g() (in module *brew_tools.brew_maths*), 10

P

pre_boil_dme() (in module *brew_tools.brew_maths*), 10
priming() (in module *brew_tools.brew_maths*), 10
print_colour() (in module *brew_tools.converter*), 11
print_gravity() (in module *brew_tools.converter*), 11
print_mass() (in module *brew_tools.converter*), 11
print_volume() (in module *brew_tools.converter*), 11

R

real_attenuation() (in module *brew_tools.brew_maths*), 10

S

srm_to_ebc() (in module *brew_tools.brew_maths*), 11
srm_to_l() (in module *brew_tools.brew_maths*), 11

`strike_temp()` (*in module `brew_tools.brew_maths`*),
[11](#)

T

`to_brix()` (*in module `brew_tools.brew_maths`*), [11](#)

`to_plato()` (*in module `brew_tools.brew_maths`*), [11](#)

`to_sg()` (*in module `brew_tools.brew_maths`*), [11](#)