convert_units Documentation

Release 1.1.1

itk

Jul 03, 2020

CONTENTS

CHAPTER

ONE

SOME PARTICULARITIES

We distinguish in this package between "pure" physical units and agronomic, namely yield units.

For instance, in order to convert from m into cm, the only information that we need is the quantity to be converted. Let's say $1\ m$ will get us $100\ cm$ regardless of the object that is $1m$ long (Fig. 1.1). This is what we call a "pure" physical unit.

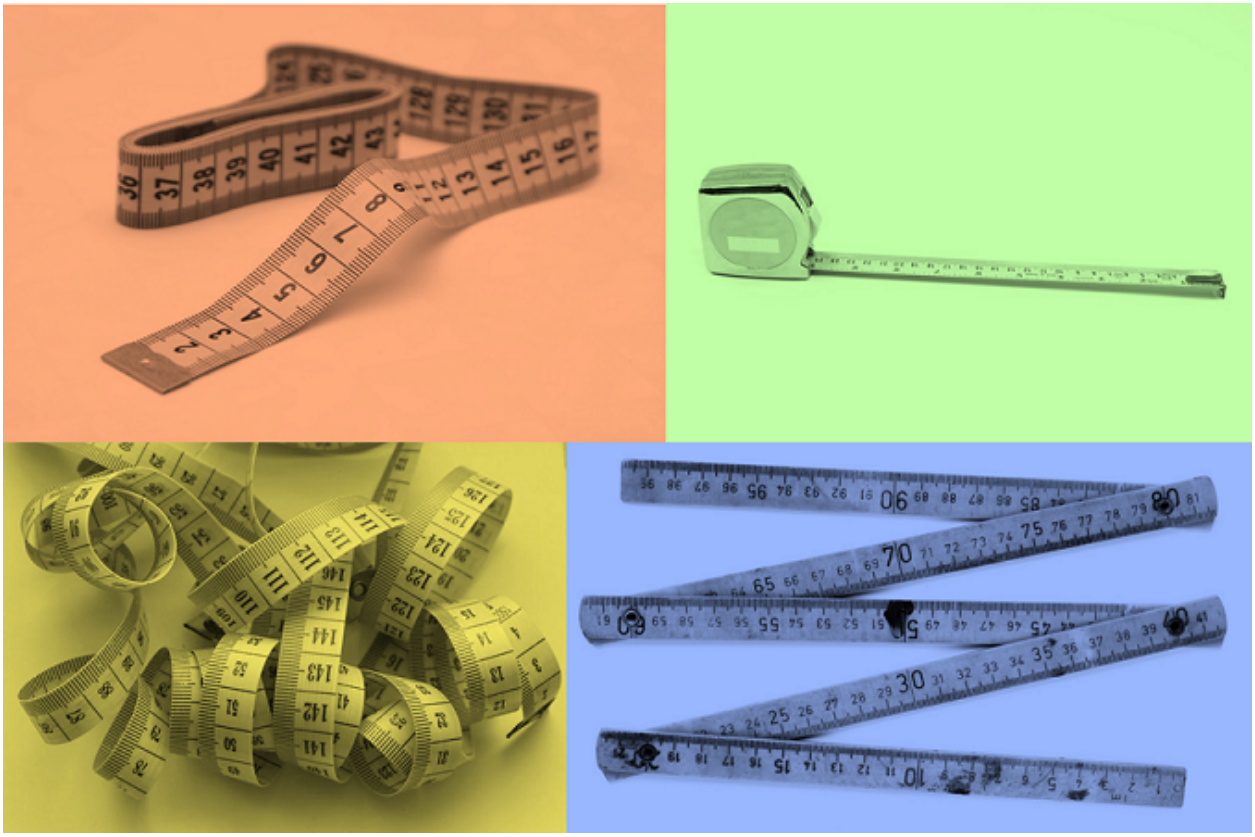

Fig. 1.1: A length of $1\ m$ is equal to $100\ cm$, whether the object of concern is made of wood, tissue, plastic, metal or any other material.

On the contrary, the conversion of yield units will require other information than just quantity. For example, if we want to know how much bushels are equivalent to, let's say, $1\ t$, then we need to know which crop we are dealing with because a bushel is an empirical unit that mixes volume and mass, and hence one bushel of apple is not equal to one bushel of corn (Fig. 1.2). Finally, the user should keep in mind that yield conversion is performed for the standard humidity values for a given crop. For example soybean yield conversion from $1\ kg$ into $1\ bu$ assumes implicitely that the yield has 13% humidity (crops standard humidity values can be found in `units_mapping.py`).

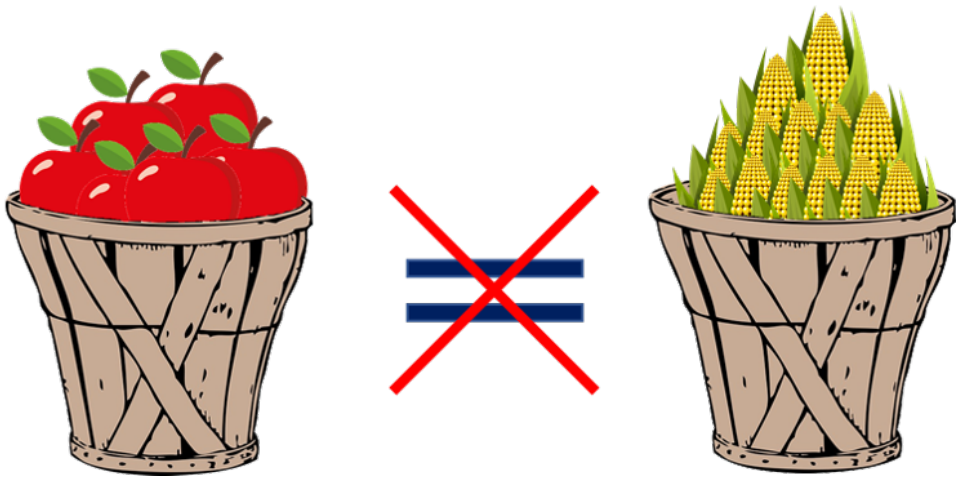

Fig. 1.2: A bushel is an empirical unit that mixes volume and mass. Therefore, a bushel of apple is not equal to a bushel of corn.

CHAPTER

TWO

USAGE

2.1 Functions

There are 3 functions for unit conversion:

1. `convert_unit()`:

This function converts all units except temperature and yield. Its has the following signature:

```
convert_unit(value, from_unit, to_unit)
```

where `value` is the value to be converted `from_unit` is the actual unit, and `to_unit` is the goal unit.

2. `convert_temperature_unit()`:

This function converts only temperature units. Its has the same signature as `convert_unit()`

```
convert_temperature_unit(value, from_unit, to_unit)
```

3. `convert_yield_unit()`:

This function converts yield units. It requires the name of the crop as an additional input compared to the two previous functions:

```
convert_yield_unit(value, from_unit, to_unit, crop)
```

2.2 Units format

Units format is given as a `str` object. The user is fairly free to use any desired unit format as long as the used format makes sense. For example, the unit velocity of one kilometer per hour could be $1\ km/h$ or $1\ km.h-1$:

```
convert_unit(1, 'km/h', 'm/s') == convert_unit(1, 'km/h', 'm.s-1')
```

CHAPTER

THREE

AVAILABLE UNITS (UNITS_MAPPING)

This package covers a large number of units coming from different metric systems (e.g. International System of Units, British system, United States customary system). A raw description of the available units follows in the block below.

```
length = {
    'm': ('m', 1),
    'mm': ('m', 1e-3),
    'cm': ('m', 1e-2),
    'dm': ('m', 1e-1),
    'dam': ('m', 10),
    'hm': ('m', 1e2),
    'km': ('m', 1e3),
    'inch': ('m', 2.54e-2),  # source https://en.wikipedia.org/wiki/Inch
    'in': ('m', 2.54e-2),  # source https://en.wikipedia.org/wiki/Inch
    'foot': ('m', 0.3048),  # https://en.wikipedia.org/wiki/Foot_(unit)
    'feet': ('m', 0.3048),  # https://en.wikipedia.org/wiki/Foot_(unit)
    'mile': ('m', 1609.344),  # source https://en.wikipedia.org/wiki/Mile
↪#International_mile [international mile]
    'yard': ('m', 0.9144),  # https://en.wikipedia.org/wiki/Yard
    'furlong': ('m', 201.1680),  # source https://en.wikipedia.org/wiki/Furlong␣
↪[international definition]
    'fur': ('m', 201.1680),  # source https://en.wikipedia.org/wiki/Furlong␣
↪[international definition]
    'link': ('m', 0.201168),  # source https://en.wikipedia.org/wiki/Link_(unit)␣
↪[international]
    'li': ('m', 0.201168),  # source https://en.wikipedia.org/wiki/Link_(unit)␣
↪[international]
    'rod': ('m', 5.0292),  # source https://en.wikipedia.org/wiki/Rod_(unit)␣
↪[international]
    'rd': ('m', 5.0292),  # see rod
    'chain': ('m', 20.1168),  # source https://en.wikipedia.org/wiki/Gunter#27s_chain␣
↪[international]
    'ch': ('m', 20.1168),  # source https://en.wikipedia.org/wiki/Gunter#27s_chain␣
↪[international]
    'league': ('m', 4828.042),  # source https://en.wikipedia.org/wiki/League_(unit)␣
↪[English land league]
    'lea': ('m', 4828.042),  # source https://en.wikipedia.org/wiki/League_(unit)␣
↪[English land league]
    'pole': ('m', 5.0292),  # see rod
    'perch': ('m', 5.0292)  # see rod
}

surface = {
    'm2': ('m2', 1),
    'mm2': ('m2', 1e-6),
```

(continues on next page)

(continued from previous page)

```
    'cm2': ('m2', 1e-4),
    'dm2': ('m2', 1e-2),
    'dam2': ('m2', 1e2),
    'hm2': ('m2', 1e4),
    'km2': ('m2', 1e6),
    'ha': ('m2', 1e4),
    'acre': ('m2', 4046.8564224),  # source https://en.wikipedia.org/wiki/Acre
↪#Differences_between_international_and_
    # US_survey_acres [international]
    'ac': ('m2', 4046.8564224),
    # source https://en.wikipedia.org/wiki/Acre#Differences_between_international_and_
↪US_survey_acres [international]
    'sq ft': ('m2', 0.3048 ** 2),  # see feet
    'sq yd': ('m2', 0.9144 ** 2),  # see yard
    'rood': ('m2', 1012),  # source https://en.wikipedia.org/wiki/Rood_(unit)
    'sq mile': ('m2', 1609.344 ** 2)  # see mile
}

volume = {
    'm3': ('m3', 1),
    'mm3': ('m3', 1e-9),
    'cm3': ('m3', 1e-6),
    'dm3': ('m3', 1e-3),
    'dam3': ('m3', 1e3),
    'hm3': ('m3', 1e6),
    'km3': ('m3', 1e9),
    'L': ('m3', 1e-3),
    'mL': ('m3', 1e-6),
    'cL': ('m3', 1e-5),
    'dL': ('m3', 1e-4),
    'daL': ('m3', 1e-2),
    'hL': ('m3', 1e-1),
    'kL': ('m3', 1),
    'cu in': ('m3', 2.54e-2 ** 3),  # see inch
    'cu ft': ('m3', 0.3048 ** 3),  # see feet
    'cu yd': ('m3', 0.9144 ** 3),  # see yard
    'acre ft': ('m3', 4046.8564224 * 0.3048),  # see acre and feet
    'pt': ('m3', 4.73176473e-04),
    # source https://en.wikipedia.org/wiki/Pint#United_States_liquid_pint [US liquid␣
↪pint]
    'qt': ('m3', 9.46352946e-04),
    # source https://en.wikipedia.org/wiki/Quart#United_States_liquid_quart [US␣
↪liquid quart]
    'pk': ('m3', 0.00880976754172),  # source https://global.britannica.com/science/
↪peck [US]
    'gal': ('m3', 0.003785411784),
    # source https://en.wikipedia.org/wiki/Gallon#The_US_liquid_gallon [US liquid␣
↪gallon]
    'bbl': ('m3', 0.119240471196),
    # source https://en.wikipedia.org/wiki/Barrel_(unit)#Fluid_barrel_in_the_US_and_
↪UK (US liquid gallon, not beer]
    'hogshead': ('m3', 0.238480942392),  # source https://en.wikipedia.org/wiki/
↪Hogshead [US wine]
}

mass = {
    'kg': ('kg', 1),
```

(continues on next page)

(continued from previous page)

```
    'mg': ('kg', 1e-6),
    'cg': ('kg', 1e-5),
    'dg': ('kg', 1e-4),
    'g': ('kg', 1e-3),
    'dag': ('kg', 1e-2),
    'hg': ('kg', 1e-1),
    'q': ('kg', 100),
    't': ('kg', 1e3),
    'lb': ('kg', 0.45359237),  # source https://en.wikipedia.org/wiki/Pound_(mass)␣
↪[international]
    'lbs': ('kg', 0.45359237),  # source https://en.wikipedia.org/wiki/Pound_(mass)␣
↪[international]
    'ton': ('kg', 907.18474),  # source https://en.wikipedia.org/wiki/Ton#Units_of_
↪mass.2Fweight [US]
    'long ton': ('kg', 1016.0469088),  # source https://en.wikipedia.org/wiki/Ton
↪#Units_of_mass.2Fweight [UK]
    'gr': ('kg', 6.479891000000001e-05),  # source https://en.wikipedia.org/wiki/
↪Grain_(unit) [Troy]
    'dr': ('kg', 0.0017718451953125),  # source https://en.wikipedia.org/wiki/Dram_
↪(unit)#British_unit_of_mass
    # [avoirdupois]
    'oz': ('kg', 0.028349523125),  # source https://en.wikipedia.org/wiki/Ounce␣
↪[avoirdupois]
    'cwt': ('kg', 45.359237),  # source https://en.wikipedia.org/wiki/Hundredweight␣
↪[US]
    'dwt': ('kg', 0.00155517384),
    'oz t': ('kg', 0.0311034768),  # source https://en.wikipedia.org/wiki/Troy_ounce
    'lb t': ('kg', 0.3732417216),  # source https://en.wikipedia.org/wiki/Pound_(mass)
↪#Troy_pound
}

time = {
    's': ('s', 1),
    'min': ('s', 60),
    'h': ('s', 3600),
    'day': ('s', 86400),
    'ms': ('s', 1e-3),
}

frequency = {
    'Hz': (['/s'], 1),
}

force = {
    'N': (['.kg', '.m', '/s2'], 1),
    'kN': (['.kg', '.m', '/s2'], 1e3),
    'pdl': (['.kg', '.m', '/s2'], 0.138254954376),  # source https://en.wikipedia.org/
↪wiki/Poundal
    'lbf': (['.kg', '.m', '/s2'], 4.4482216152605),  # source https://en.wikipedia.
↪org/wiki/Pound_(force)
}

pressure = {
    'Pa': (['.kg', '/m', '/s2'], 1),
    'mPa': (['.kg', '/m', '/s2'], 1e-3),
    'cPa': (['.kg', '/m', '/s2'], 1e-2),
    'dPa': (['.kg', '/m', '/s2'], 1e-1),
```

(continues on next page)

(continued from previous page)

```
    'daPa': (['.kg', '/m', '/s2'], 10),
    'hPa': (['.kg', '/m', '/s2'], 1e2),
    'kPa': (['.kg', '/m', '/s2'], 1e3),
    'MPa': (['.kg', '/m', '/s2'], 1e6),
    'bar': (['.kg', '/m', '/s2'], 100000),  # source https://en.wikipedia.org/wiki/
↪Bar_(unit)
    'atm': (['.kg', '/m', '/s2'], 101325),  # source https://en.wikipedia.org/wiki/
↪Atmosphere_(unit)
    # source https://en.wikipedia.org/wiki/Atmospheric_pressure#Standard_atmospheric
    'mmHg': (['.kg', '/m', '/s2'], 133.322387415),  # source https://en.wikipedia.org/
↪wiki/Millimeter_of_mercury
    'torr': (['.kg', '/m', '/s2'], 133.322387415),  # see mmHg, source https://en.
↪wikipedia.org/wiki/Torr
    'inHg': (['.kg', '/m', '/s2'], 3386.389),  # source https://en.wikipedia.org/wiki/
↪Inch_of_mercury [conventional]
    'ba': (['.kg', '/m', '/s2'], 0.1),  # source https://en.wikipedia.org/wiki/Barye
    'at': (['.kg', '/m', '/s2'], 98066.5),  # source https://en.wikipedia.org/wiki/
↪Technical_atmosphere
    'psi': ('kg/m/s2', 6894.757),  # source https://en.wikipedia.org/wiki/Pounds_per_
↪square_inch
}

energy = {
    'J': (['.kg', '.m2', '/s2'], 1),
    'TJ': (['.kg', '.m2', '/s2'], 1e12),
    'mJ': (['.kg', '.m2', '/s2'], 1e-3),
    'cJ': (['.kg', '.m2', '/s2'], 1e-2),
    'dJ': (['.kg', '.m2', '/s2'], 1e-1),
    'daJ': (['.kg', '.m2', '/s2'], 10),
    'hJ': (['.kg', '.m2', '/s2'], 1e2),
    'kJ': (['.kg', '.m2', '/s2'], 1e3),
    'MJ': (['.kg', '.m2', '/s2'], 1e6),
    'BTU': (['.kg', '.m2', '/s2'], 1054.3503),
    # source https://en.wikipedia.org/wiki/British_thermal_unit [thermochemical]
    'cal': (['.kg', '.m2', '/s2'], 4.184),  # source https://en.wikipedia.org/wiki/
↪Calorie [thermochemical calorie]
    'kWh': (['.kg', '.m2', '/s2'], 3.6e6),  # source https://en.wikipedia.org/wiki/
↪Kilowatt_hour
    'Ws': (['.kg', '.m2', '/s2'], 1),  # source https://en.wikipedia.org/wiki/Watt_
↪second
    'erg': (['.kg', '.m2', '/s2'], 1e-7),  # source https://en.wikipedia.org/wiki/Erg
    'eV': (['.kg', '.m2', '/s2'], 1.602176565e-19),  # source https://en.wikipedia.
↪org/wiki/Electronvolt
}

power = {
    'W': (['kg', '.m2', '/s3'], 1),
    'hp': (['kg', '.m2', '/s3'], 745.69987158227022),
    # source https://en.wikipedia.org/wiki/Horsepower#Mechanical_horsepower [imperial,
↪ mechanical]
}

angle = {
    'rad': ('rad', 1),
    'deg': ('rad', math.pi / 180),
}
```

(continues on next page)

(continued from previous page)

```
miscellaneous = {
    's2': ('s2', 1),
    'h2': ('s2', 3600 ** 2),
    's3': ('s3', 1),
    'h3': ('s3', 3600 ** 3),
}

temperature = {
    'K': ('K', (1, 0)),
    'degreeC': ('K', (1, 273.15)),  # source https://en.wikipedia.org/wiki/Kelvin
    'degreeF': ('K', (5 / 9, -32 * 5 / 9 + 273.15))  # % source https://en.wikipedia.
↪org/wiki/Kelvin
}

lb_to_kg = 0.45359237
yield_convert = {
    'barley': ('bu', 'kg', 48 * lb_to_kg),  # at 14.5%
    'corn': ('bu', 'kg', 56 * lb_to_kg),  # at 15.5%
    'cotton': ('bu', 'kg', 32 * lb_to_kg),  # not found!
    'oat': ('bu', 'kg', 32 * lb_to_kg),  # at 14%
    'soybean': ('bu', 'kg', 60 * lb_to_kg),  # at 13%
    'wheat': ('bu', 'kg', 60 * lb_to_kg),  # at 13.5%
    'sorghum': ('bu', 'kg', 56 * lb_to_kg),  # at 13.%
}

convert_table = {}
for table in [length, surface, volume, mass, time, frequency, force, pressure, energy,
↪ power, angle, miscellaneous]:
    convert_table.update(table)
```

CHAPTER

FOUR

SRC

4.1 convert_units package

4.1.1 Submodules

convert_units.bushel_converter module

convert_units.bushel_converter.**get_coeff_from_bushel_to_kg**(*crop*)
get coeff to convert bushel into kg

Parameters **crop** (*str*) – crop name

Returns coeff to convert bushel into kg

Return type (float)

Raises **(KeyError)** – if crop is not supported

convert_units.converter module

convert_units.converter.**convert_temperature_unit**(*value*, *from_unit*, *to_unit*)
Converts a temperature value from its actual unit to a goal unit.

Parameters

- **value** (*list or float*) – variable value to be converted
- **from_unit** (*str*) – the actual unit of the variable
- **to_unit** (*str*) – the goal unit of the variable

Returns the value converted to *to_unit*

Return type (list or float)

See also:

units_mapping for the covered units.

convert_units.converter.**convert_unit**(*value*, *from_unit*, *to_unit*)
Converts a value from its actual unit to a goal unit.

Parameters

- **value** (*list or float*) – variable value to be converted
- **from_unit** (*str*) – the actual unit of the variable

- **to_unit** (*str*) – the goal unit of the variable

Returns the value converted to *to_unit*

Return type (list or float)

Raises **(ValueError)** – if units dimensions differ

See also:

`units_mapping` for the covered units.

`convert_units.converter.`**convert_yield_unit**(*value*, *from_unit*, *to_unit*, *crop*)
Converts a value from its actual unit to a goal unit with a yield component.

Parameters

- **value** (*list or float*) – variable value to be converted
- **from_unit** (*str*) – the actual unit of the variable
- **to_unit** (*str*) – the goal unit of the variable
- **crop** (*str*) – crop name of the yield value

Returns the value converted to *to_unit*

Return type (list or float)

Warning: if it is not a yield conversion

See also:

`units_mapping` for the covered units.

convert_units.formatter module

`convert_units.formatter.`**clean_unit_format**(*unit*)
Function to format the unit string

Parameters **unit** (*str*) – [-] original format

Returns (list) formatted units

`convert_units.formatter.`**simple_unit_format**(*unit*)
Function to simplify unit format

Parameters **unit** (*str*) – unit to simplify

Returns simplified unit format

Return type (str)

convert_units.si_converter module

convert_units.si_converter.**apply_coef**(*value*, *coefficient*, *unit*)
Apply a coefficient to a value depending on the first character of value's unit (. or /)

Parameters

- **value** (*list or float*) – value on which the coefficient is applied
- **coefficient** (*float*) – coefficient to apply
- **unit** (*str*) – unit to define whether the coefficient must be multiplied or divided

Returns value with coefficient applied

Return type (list or float)

convert_units.si_converter.**get_coefficient_and_si_unit**(*units*)
Returns the SI unit and the corresponding conversion coefficient.

Parameters **units** (*list of str*) – split units to be converted

Returns (list): split SI units (float): conversion coefficient

Return type (tuple)

convert_units.si_converter.**get_simplified_si_units_and_coefficient**(*unit*)
Returns simplified SI unit and its corresponding coefficient.

Parameters **unit** (*str*) – non-formatted unit

Returns (dict): simplified SI units as keys, dimensions as values (float): a coefficient value to convert in SI

Return type (tuple)

convert_units.si_converter.**simplify_unit**(*units*)
Simplifies a list of units into an ordered lower-level list of units.

Parameters **units** (*list of str*) – split units to be simplified

Returns units as keys, dimensions as values

Return type (dict)

convert_units.si_converter.**split_unit**(*unit*)
Splits a unit into 3 parts: prefix, unit string and exponent.

Parameters **unit** (*str*) – formatted simple unit

Returns prefix, string unit, exponent

Return type (str, str, str)

convert_units.temperature module

convert_units.temperature.**get_temperature_si_unit_and_coefficient**(*unit*)
Returns SI unit and its corresponding coefficient.

Parameters **unit** (*str*) – non-formatted unit

Returns (float): a coefficient value to convert in SI (dict): SI units as key, dimensions as value

Return type (tuple)

Raises

- **(ValueError)** – if units are more complicated that just one simple unit
- **(KeyError)** – if one unit is not a temperature

convert_units.units_mapping module

Mapping of units supported by the convert units package

convert_units.version module

Maintain version for this package. Do not edit this file, use 'version' section of config.

convert_units.version.**MAJOR = 1**
(int) Version major component.

convert_units.version.**MINOR = 1**
(int) Version minor component.

convert_units.version.**POST = 1**
(int) Version post or bugfix component.

4.1.2 Module contents

A package of useful functions for units conversion

PYTHON MODULE INDEX

C