papersize Documentation

Release 1.5.0

Louis Paternault

Feb 18, 2024

CONTENTS

Paper size related data and functions.

This module provides tools to manipulate paper sizes, that is:

- a dictionary of several named standard names (e.g. A4, letter) , with their respective sizes (with and height);
- functions to convert sizes between units;
- functions to manipulate paper orientation (portrait or landscape);
- tools to parse paper sizes, so that you do not have to worry about the format of paper sizes provided by your user, it being *a4* or *21cm x 29.7cm*.

CHAPTER

ONE

DOWNLOAD AND INSTALL

See the main project page for instructions, and changelog.

CHAPTER

TWO

MODULE DOCUMENTATION

2.1 Constants

papersize.**UNITS**

Dictionary of units.

Keys are unit abbreviation (e.g. `pt` or `cm`), and values are their value in points (e.g. `UNITS['pt']` is 1, `UNITS['pc']` is 12), as `decimal.Decimal` objects.

```
UNITS = {
    "": Decimal("1"),  # Default is point (pt)
    "pt": Decimal("1"),  # point
    "mm": Decimal("7227") / Decimal("2540"),  # millimeter
    "cm": Decimal("7227") / Decimal("254"),  # centimeter
    "in": Decimal("72.27"),  # inch
    "bp": Decimal("803") / Decimal("800"),  # big point
    "pc": Decimal("12"),  # pica
    "dd": Decimal("1238") / Decimal("1157"),  # didot
    "cc": Decimal("14856") / Decimal("1157"),  # cicero
    "nd": Decimal("685") / Decimal("642"),  # new didot
    "nc": Decimal("1370") / Decimal("107"),  # new cicero
    "sp": Decimal("1") / Decimal("65536"),  # scaled point
}
```

papersize.**UNITS_HELP**

Human description of each unit.

Keys are unit abbreviation (e.g. `pt` or `cm`), and values are strings explaining the meaning of this unit. You can use it to list and explain to your users the available units.

Note that the descriptions are *translated*.

papersize.**SIZES**

Dictionary of named sizes.

Keys are names (e.g. `a4`, `letter`) and values are strings, human-readable, and parsable by *parse_papersize()* (e.g. `21cm x 29.7cm`).

```
SIZES = {
    # http://www.printernational.org/iso-paper-sizes.php
    "4a0": "1682mm x 2378mm",
    "2a0": "1189mm x 1682mm",
```

(continues on next page)

(continued from previous page)

```
    "a0": "841mm x 1189mm",
    "a1": "594mm x 841mm",
    "a2": "420mm x 594mm",
    "a3": "297mm x 420mm",
    "a4": "210mm x 297mm",
    "a5": "148mm x 210mm",
    "a6": "105mm x 148mm",
    "a7": "74mm x 105mm",
    "a8": "52mm x 74mm",
    "a9": "37mm x 52mm",
    "a10": "26mm x 37mm",
    "b0": "1000mm x 1414mm",
    "b1": "707mm x 1000mm",
    "b2": "500mm x 707mm",
    "b3": "353mm x 500mm",
    "b4": "250mm x 352mm",
    "b5": "176mm x 250mm",
    "b6": "125mm x 176mm",
    "b7": "88mm x 125mm",
    "b8": "62mm x 88mm",
    "b9": "44mm x 62mm",
    "b10": "31mm x 44mm",
    "a2extra": "445mm x 619mm",
    "a3extra": "322mm x 445mm",
    "a3super": "305mm x 508mm",
    "supera3": "305mm x 487mm",
    "a4extra": "235mm x 322mm",
    "a4super": "229mm x 322mm",
    "supera4": "227mm x 356mm",
    "a4long": "210mm x 348mm",
    "a5extra": "173mm x 235mm",
    "sob5extra": "202mm x 276mm",
    # http://www.engineeringtoolbox.com/office-paper-sizes-d_213.html
    "letter": "8.5in x 11in",
    "legal": "8.5in x 14in",
    "executive": "7in x 10in",
    "tabloid": "11in x 17in",
    "statement": "5.5in x 8.5in",
    "halfletter": "5.5in x 8.5in",
    "folio": "8in x 13in",
    # http://hplipopensource.com/hplip-web/tech_docs/page_sizes.html
    "flsa": "8.5in x 13in",
    # http://www.coding-guidelines.com/numbers/ndb/units/area.txt
    "flse": "8.5in x 13in",
    # http://jexcelapi.sourceforge.net/resources/javadocs/2_6_10/docs/jxl/format/
↪PaperSize.html
    "note": "8.5in x 11in",
    "11x17": "11in x 17in",
    "10x14": "10in x 14in",
    # https://en.wikipedia.org/w/index.php?title=Paper_size&oldid=814180250
    "c0": "917mm × 1297mm",
    "c1": "648mm × 917mm",
```

(continues on next page)

(continued from previous page)

```
    "c2": "458mm × 648mm",
    "c3": "324mm × 458mm",
    "c4": "229mm × 324mm",
    "c5": "162mm × 229mm",
    "c6": "114mm × 162mm",
    "c7": "81mm × 114mm",
    "c8": "57mm × 81mm",
    "c9": "40mm × 57mm",
    "c10": "28mm × 40mm",
    "juniorlegal": "5in × 8in",
    "memo": "halfletter",
    "governmentletter": "8in × 10in",
    "governmentlegal": "8.5in × 13in",
    "ledger": "17in x 11in",
    "arch1": "9in x 12in",
    "arch2": "12in x 18in",
    "arch3": "18in x 24in",
    "arch4": "24in x 36in",
    "arch5": "30in x 42in",
    "arch6": "36in x 48in",
    "archa": "arch1",
    "archb": "arch2",
    "archc": "arch3",
    "archd": "arch4",
    "arche1": "arch5",
    "arche": "arch6",
    "arche2": "26in x 38in",
    "arche3": "27in x 39in",
}
```

papersize.**SIZES_HELP**

Human description of each paper size.

Keys are size abbreviation (e.g. `A4` or `letter`), and values are strings explaining the meaning of this size. You can use it to list and explain to your users the available paper sizes.

For historical reasons, keys of `SIZES` are lower cases, while keys of `SIZES_HELP` are not. But case aside, those dictionaries contain exactly the same set of keys.

Note that the descriptions are *translated*.

papersize.**PORTRAIT**

Constant corresponding to the portrait orientation

That is, height greater than width.

papersize.**LANDSCAPE**

Constant corresponding to the landscape orientation

That is, width greater than height.

2.2 Unit conversion

papersize.**convert_length**(*length*, *orig*, *dest*)

Convert length from one unit to another.

Parameters

- **length** (*decimal.Decimal*) – Length to convert, as any object convertible to a decimal.Decimal.
- **orig** (*str*) – Unit of length, as a string which is a key of *UNITS*.
- **dest** (*str*) – Unit in which length will be converted, as a string which is a key of *UNITS*.

Due to floating point arithmetic, there can be small rounding errors.

```
>>> convert_length(0.1, "cm", "mm")
Decimal('1.000000000000000055511151231')
```

2.3 Parsers

papersize.**parse_length**(*string*, *unit='pt'*)

Return a length corresponding to the string.

Parameters

- **string** (*str*) – The string to parse, as a length and a unit, for instance 10.2cm.
- **unit** (*str*) – The unit of the return value, as a key of *UNITS*.

Returns

The length, in an unit given by the unit argument.

Return type

decimal.Decimal

```
>>> parse_length("1cm", "mm")
Decimal('1E+1')
>>> parse_length("1cm", "cm")
Decimal('1')
>>> parse_length("10cm")
Decimal('284.52755905511811023622047 24')
```

papersize.**parse_couple**(*string*, *unit='pt'*)

Return a tuple of dimensions.

Parameters

string (*str*) – The string to parse, as "LENGTHxLENGTH" (where LENGTH are length, parsable by *parse_length()*). Example: 21cm x 29.7cm. The separator can be x, × or empty, surrounded by an arbitrary number of spaces. For instance: 2cmx3cm, 2cm x 3cm, 2cm×3cm, 2cm 3cm.

Return type

tuple

Returns

A tuple of decimal.Decimal, representing the dimensions.

```
>>> parse_couple("1cm 10cm", "mm")
(Decimal('1E+1'), Decimal('1E+2'))
>>> parse_couple("1mm 10mm", "cm")
(Decimal('0.1'), Decimal('1'))
```

papersize.**parse_papersize**(*string*, *unit='pt'*)

Return the papersize corresponding to string.

Parameters

- **string** (*str*) – The string to parse. It can be either a named size (as keys of constant *SIZES*), or a couple of lengths (that will be processed by *parse_couple()*). The named paper sizes are case insensitive. The following strings return the same size: `a4`, `A4`, `21cm 29.7cm`, `210mmx297mm`, `21cm × 297mm`...
- **unit** (*str*) – The unit of the return values.

Returns

The paper size, as a couple of `decimal.Decimal`.

Return type

`tuple`

```
>>> parse_papersize("A4", "cm")
(Decimal('21.000000000000000000000000000'), Decimal('29.700000000000000000000000000'))
>>> parse_papersize("21cm x 29.7cm", "mm")
(Decimal('210.000000000000000000000000000'), Decimal('297.000000000000000000000000000'))
>>> parse_papersize("10 100")
(Decimal('10'), Decimal('100'))
```

2.4 Paper orientation

papersize.**is_portrait**(*width*, *height*, *, *strict=False*, *fuzzy=False*, *ndigits=7*)

Return whether paper orientation is portrait

That is, height greater or equal to width.

Parameters

- **width** – Width of paper, as any sortable object.
- **height** – Height of paper, as any sortable object.
- **strict** (*bool*) – If `False`, square format (width equals height) is considered portrait; if `True` square format is not considered portrait.
- **fuzzy** (*bool*) – If `True`, comparison is done up to `ndigits` digits.
- **ndigits** (*int*) – Number of digits when using fuzzy comparison.

```
>>> is_portrait(11, 10)
False
>>> is_portrait(10, 10)
True
>>> is_portrait(10, 11)
True
```

papersize.**is_landscape**(*width*, *height*, *, *strict=False*, *fuzzy=False*, *ndigits=7*)

Return whether paper orientation is landscape

That is, width greater or equal to height.

Parameters

- **width** – Width of paper, as any sortable object.
- **height** – Height of paper, as any sortable object.
- **strict** – If `False`, square format (width equals height) is considered landscape; if `True` square format is not considered landscape.
- **fuzzy** (*bool*) – If `True`, comparison is done up to `ndigits` digits.
- **ndigits** (*int*) – Number of digits when using fuzzy comparison.

```
>>> is_landscape(11, 10)
True
>>> is_landscape(10, 10)
True
>>> is_landscape(10, 11)
False
```

papersize.**is_square**(*width*, *height*, *, *fuzzy=False*, *ndigits=7*)

Return whether paper is a square (width equals height).

Parameters

- **width** – Width of paper, as any sortable object.
- **height** – Height of paper, as any sortable object.
- **fuzzy** (*bool*) – If `True`, comparison is done up to `ndigits` digits.
- **ndigits** (*int*) – Number of digits when using fuzzy comparison.

```
>>> is_square(11, 10)
False
>>> is_square(10, 10)
True
>>> is_square(10, 10.00000001, fuzzy=False)
False
>>> is_square(10, 10.00000001, fuzzy=True)
True
>>> is_square(10, 10.00000001, fuzzy=True, ndigits=10)
False
```

papersize.**rotate**(*size*, *orientation*)

Return the size, rotated if necessary to make it portrait or landscape.

Parameters

- **size** (*tuple*) – Couple paper of dimension, as sortable objects (`int`, `float`, `decimal.Decimal`...).
- **orientation** – Return format, one of `PORTRAIT` or `LANDSCAPE`.

Returns

The size, as a couple of dimensions, of the same type of the `size` parameter.

Return type
tuple

```
>>> rotate((21, 29.7), PORTRAIT)
(21, 29.7)
>>> rotate((21, 29.7), LANDSCAPE)
(29.7, 21)
```

2.5 Exceptions

class papersize.**PapersizeException**

All exceptions of this module inherit from this one.

class papersize.**CouldNotParse**(*string*)

Raised when a string could not be parsed.

Parameters
string (*str*) – String that could not be parsed.

class papersize.**UnknownOrientation**(*string*)

Raised when type of argument Orientation is wrong.

Parameters
string (*obj*) – Object wrongly provided as an orientation.

CHAPTER

THREE

INTERNATIONALISATION

Constants *SIZES_HELP* and *UNITS_HELP* are translated. If your application is not translated, just ignore it. If it is translated (using `gettext` or babel for instance), translations are provided.

3.1 How to use it?

This module provides *translation_directory()*:

papersize.**translation_directory**()

Return an context manager proiding a directory in which translation files are located.

New in version 1.5.0.

3.1.1 Example with `gettext`

```
with papersize.translation_directory() as directory:
    gettext.bindtextdomain("papersize", localedir=directory)
    gettext.textdomain("papersize")
    _ = gettext.gettext
    print(_("centimeter"))
```

```
centimètre
```

3.1.2 Everlasting translation directory

Function *translation_directory()* is a context manager, so the directory it returns is only guaranteed to last until its end. If you need the (maybe temporary) directory to last until your application exists, you can use the following example (source).

```
import contextlib
import atexit

def papersizetranslations():
    file_manager = contextlib.ExitStack()
    atexit.register(file_manager.close)
    return file_manager.enter_context(papersize.translation_directory())
```

3.2 Languages

Right now, only French translations are provided. Translations in other languages are gladly accepted.

CHAPTER

FOUR

CONTRIBUTING

Bla bla bla

4.1 Translation

Install babel, and cd to the root of the `papersize` repository. Then:

- Extract strings to translate:

```
pybabel extract -F babel.cfg -o papersize.pot .
```

- Update French translations catalog (replace `update` with `init` for first translation of a new language):

```
pybabel update -i papersize.pot -d papersize/translations --domain papersize -l fr
```

- Manually update translations:

```
$EDITOR papersize/translations/fr/LC_MESSAGES/papersize.po
```

- Compile translations:

```
pybabel compile -d papersize/translations --domain papersize
```

CHAPTER

FIVE

INDICES AND TABLES

- genindex
- modindex
- search

INDEX