
steenzout.barcode documentation

Release 1.0

Thorsten Weimann, Alexander Shorin, Pedro Salgado

Apr 17, 2017

Contents

1	Contents	1
1.1	Introduction	1
1.2	Command-line interface	2
1.3	Provided barcodes	3
1.4	Writer	4
1.5	License	8
2	Indices and tables	9
3	Latest version	11
4	Related Links	13
5	API documentation	15
5.1	steenzout package	15
	Python Module Index	29

Introduction

This package was created to have barcodes available without having [PIL](#) (Python Imaging Library) installed.

As of version 0.4b1 PIL is also supported for creating barcodes.

All you need to create a barcode is to know the system (EAN, UPC, ...) and the code (e.g. for EAN-13: 123456789102).

As you see, you do not need the checksum, it will be calculated automatically.

In some systems (Code 39) the checksum is optional, there you can give the *add_checksum* keyword argument (default is True).

As of version 0.7beta3 Python 3 is supported, but not well tested.

Creating barcodes as SVG

To generate barcodes as SVG objects, you can use the default writer (simply not specify a writer).

Quick example:

```
>>> from steenzout import barcode
>>> ean = barcode.get('ean13', '123456789102')
# Now we look if the checksum was added
>>> ean.get_fullcode()
u'1234567891026'
>>> filename = ean.save('ean13')
>>> filename
u'ean13.svg'
>>> options = dict(compress=True)
>>> filename = ean.save('ean13', options)
>>> filename
u'ean13.svgz'
```

Now you have ean13.svg and the compressed ean13.svgz in your current working directory. Open it and see the result.

Creating barcodes as Image

New in version 0.4b1.

To generate barcodes as images, you must provide the ImageWriter to the *get* function.

Without any options, the images are rendered as PNG.

Quick example:

```
>>> from steenzout import barcode
>>> from steenzout.barcode.writer import ImageWriter
>>> ean = barcode.get('ean13', '123456789102', writer=ImageWriter())
>>> filename = ean.save('ean13')
>>> filename
u'ean13.png'
```

Command-line interface

The install script detects your Python version and adds the major and minor Python version number to the executable script.

For example, on Python 2.7, the command-line interface tool will be *py27-barcode*.

Commands

Usage:

```
$ py27-barcode --help
Usage: py27-barcode [OPTIONS] COMMAND [ARGS]...

Options:
  --version  Show the version and exit.
  --help     Show this message and exit.

Commands:
  encodings  List the available bar codes.
  formats    List the available image formats.
  generate    Generates the bar code.
```

Generate:

```
$ py27-barcode generate --help
Usage: py27-barcode generate [OPTIONS] INPUT OUTPUT

  Generates the bar code.

Options:
  -v, --verbose                  Enables verbosity.
  -e, --encoding ENCODING
  → [code128|code39|ean|ean13|ean8|gs1|gtin|isbn|isbn10|isbn13|issn|jan|pzn|upc|upca]
  -f, --format [BMP|EPS|GIF|JPEG|MSP|PCX|PNG|SVG|TIFF|XBM]
```

```
-u, --unit TEXT
--help                Show this message and exit.
```

The number of output formats available will depend if you installed *PIL*:

```
$ pip install steenzout.barcode[image]
```

Provided barcodes

Code 39

class Code39 (*code*, *writer=None*)
Initializes a new Code39 instance.

code
str

writer
writer.BaseWriter – writer class.

Parameters

- **code** (*str*) – Code 39 string without * and checksum (added automatically if *add_checksum* is True).
- **writer** (*writer* Instance) – instance of writer class to render the bar code.
- **add_checksum** (*bool*) – add the checksum to code or not.

Code 128

New in version 0.8beta1.

class Code128 (*code*, *writer=None*)
Initializes a new Code128 instance. The checksum is added automatically when building the bars.

Parameters

- **code** (*str*) – code 128 string without checksum (added automatically).
- **writer** (*writer.BaseWriter*) – instance of writer class to render the bar code.

PZN

class PZN (*code*, *writer=None*)
Initializes new German number for pharmaceutical products.

Parameters

- **pzn** (*str*) – code to render.
- **writer** (*writer.BaseWriter*) – instance of writer class to render the bar code.

EAN-13

EuropeanArticleNumber13
alias of EAN13

EAN-8

EuropeanArticleNumber8
alias of EAN8

JAN

JapanArticleNumber
alias of JAN

ISBN-13

InternationalStandardBookNumber13
alias of ISBN13

ISBN-10

InternationalStandardBookNumber10
alias of ISBN10

ISSN

InternationalStandardSerialNumber
alias of ISSN

UPC-A

UniversalProductCodeA
alias of UPCA

Writer

Common Writer Options

All writer take the following options (specified as keyword arguments to *Barcode.save(filename, option=value)* or set via *Writer.set_options(option=value)*).

Note: See the documentation of the specific writer for special options, only available for this writer.

Common Options:

module_width The width of one barcode module in mm as *float*. Defaults to **0.2**.

module_height The height of the barcode modules in mm as *float*. Defaults to **15.0**.

quiet_zone Distance on the left and on the right from the border to the first (last) barcode module in mm as *float*. Defaults to **6.5**.

font_size Font size of the text under the barcode in pt as *integer*. Defaults to **10**.

text_distance Distance between the barcode and the text under it in mm as *float*. Defaults to **5.0**.

background The background color of the created barcode as *string*. Defaults to **white**.

foreground The foreground and text color of the created barcode as *string*. Defaults to **black**.

New in version 0.6.

center_text If true (the default) the text is centered under the barcode else left aligned.

Note: Some barcode classes change the above defaults to fit in some kind of specification.

Writers

SVGWriter

Creates barcodes as (compressed) SVG objects.

Special Options

In addition to the common writer options you can give the following special option.

Special Option:

compress Boolean value to output a compressed SVG object (.svgz). Defaults to **False**.

ImageWriter

New in version 0.4b1.

Creates barcodes as image. All imagetypes supported by PIL are available.

Special Options

In addition to the common writer options you can give the following special options.

Special Options:

format The image file format as *string*. All formats supported by PIL are valid (e.g. PNG, JPEG, BMP, ...). Defaults to **PNG**.

dpi DPI as *integer* to calculate the image size in pixel. This value is used for all mm to px calculations. Defaults to **300**.

Create your own writer

To create your own writer, inherit from `steenzout.barcode.writer.BaseWriter`.

In your `__init__` method call `BaseWriter`'s `__init__` and give your callbacks for `initialize(raw_code)`, `paint_module(xpos, ypos, width, color)`, `paint_text(xpos, ypos)` and `finish()`.

Now instantiate a new barcode and give an instance of your new writer as argument.

If you now call `render` on the barcode instance your callbacks get called.

API (autogenerated)

class Base (*initialize=None, paint_module=None, paint_text=None, finish=None*)
Base class for all writers.

Initializes the basic writer options. Sub-classes can add more attributes and can set them directly or using `self.set_options(option=value)`.

Parameters

- **initialize** (*function*) – Callback for initializing the inheriting writer. Is called: `callback_initialize(raw_code)`
- **paint_module** (*function*) – Callback for painting one barcode module. Is called: `callback_paint_module(xpos, ypos, width, color)`
- **paint_text** (*function*) – Callback for painting the text under the barcode. Is called: `callback_paint_text(xpos, ypos)` using `self.text` as text.
- **finish** (*function*) – Callback for doing something with the completely rendered output. Is called: `return callback_finish()` and must return the rendered output.

calculate_size (*modules_per_line, number_of_lines, dpi=300*)
Calculates the size of the barcode in pixel.

Parameters

- **modules_per_line** (*int*) – number of modules in one line.
- **number_of_lines** (*int*) – number of lines of the barcode.
- **dpi** (*int*) – DPI to calculate.

Returns Width and height of the barcode in pixel.

Return type (tuple[int, int])

register_callback (*action, callback*)
Register one of the three callbacks if not given at instance creation.

Parameters

- **action** (*str*) – One of 'initialize', 'paint_module', 'paint_text', 'finish'.

- **callback** (*function*) – The callback function for the given action.

render (*code*)

Renders the barcode to whatever the inheriting writer provides, using the registered callbacks.

Parameters **code** (*list[str]*) – list of strings matching the writer spec (only contain 0 or 1).

save (*filename, output*)

See *Interface.save()*.

set_options (*options*)

Sets the given options as instance attributes (only if they are known).

Parameters **options** (*dict*) – All known instance attributes and more if the child class has defined them before this call.

class Interface

Writer interface.

save (*filename, content*)

Saves contents to *filename*.

Parameters

- **filename** (*str*) – filename without extension.
- **content** (*str*) – output of rendering process.

Returns the filename.

Return type (*str*)

create_svg_object ()

Returns a blank SVG document.

Returns XML document.

Return type (*xml.dom.minidom.DocumentType*)

mm2px (*value, dpi=300*)

Converts the given value in millimeters into dots per inch.

Parameters

- **value** (*float*) – value, in millimeters.
- **dpi** (*int*) – resolution, in dots per inch.

Returns value, in dots per inch.

Return type (*float*)

pt2mm (*value*)

Converts given value in points to millimeters.

Parameters **value** (*int*) – value in points.

Returns value, in millimeters.

Return type (*float*)

License

This package is distributed under the MIT license:

```
Copyright (c) 2010-2013 Thorsten Weimann  
Copyright (c) 2014 Alexander Shorin  
Copyright (c) 2016 Pedro Salgado
```

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.

CHAPTER 2

Indices and tables

- search

CHAPTER 3

Latest version

You can always download the latest release [here](#).

CHAPTER 4

Related Links

Project page <https://github.com/steenzout/python-barcode/>

Issues <https://github.com/steenzout/python-barcode/issues/>

This section contains information on a specific function, class, or method.

steenzout package

Subpackages

steenzout.barcode package

Subpackages

steenzout.barcode.charsets package

Submodules

steenzout.barcode.charsets.code128 module

Code 128 character sets.

steenzout.barcode.charsets.code39 module

Code 39 character sets.

steenzout.barcode.charsets.ean module

EAN character sets.

steenzout.barcode.charsets.upc module

UPC character sets.

Module contents

Character sets package.

steenzout.barcode.cli package

Module contents

Command-line tool package.

Submodules

steenzout.barcode.base module

Base bar code module.

```
class Barcode (code, writer=None)
    Bases: steenzout.object.Object
    Base bar code class.

    build()

        Returns

        Return type (str)

    classmethod calculate_checksum (code)
        Calculates a bar code checksum.

        Parameters code (str) – the code.

        Returns the checksum.

        Return type (integer)

    checksum
        (int) – bar code checksum.

    code
        (str) – bar code.

    digits = 0

    get_fullcode()
        Returns the full code, encoded in the barcode.

        Returns Full human readable code.

        Return type (str)

    name = ‘
```

```
raw = None
```

render (*writer_options=None*)

Renders the barcode using *self.writer*.

Parameters **writer_options** (*dict*) – options for *self.writer*, see writer docs for details.

Returns output of the writer's render method.

save (*filename, options=None*)

Renders the barcode and saves it in *filename*.

Parameters

- **filename** (*str*) – filename to save the barcode in (without filename extension).
- **options** (*dict*) – the same as in `:py:func:'self.render'`.

Returns Filename with extension.

Return type (*str*)

to_ascii ()

Returns ASCII representation of the bar code.

Returns ASCII representation of the bar code.

Return type (*str*)

classmethod validate (*code*)

Validates the given bar code.

write (*fp, options=None*)

Renders the barcode and writes it to the file like object *fp*.

Parameters

- **fp** – File like object object to write the raw data in.
- **options** (*dict*) – the same as in `:py:func:'self.render'`.

writer

(*steenzout.barcode.writer.Interface*) – writer instance.

steenzout.barcode.codex module

Module: barcode.codex

Provided barcodes Code 39, Code 128, PZN

class Code128 (*code, writer=None*)

Bases: *steenzout.barcode.base.Barcode*

Initializes a new Code128 instance. The checksum is added automatically when building the bars.

Parameters

- **code** (*str*) – code 128 string without checksum (added automatically).
- **writer** (*writer.BaseWriter*) – instance of writer class to render the bar code.

build ()

static calculate_checksum (*code*)

encoded

get_fullcode ()

```
name = 'Code 128'
```

```
render (writer_options=None)
```

```
static validate (code)
```

```
class Code39 (code, writer=None)
```

Bases: `steenzout.barcode.base.Barcode`

Initializes a new Code39 instance.

```
code
```

str

```
writer
```

`writer.BaseWriter` – writer class.

Parameters

- **code** (*str*) – Code 39 string without * and checksum (added automatically if *add_checksum* is True).
- **writer** (`writer` Instance) – instance of writer class to render the bar code.
- **add_checksum** (*bool*) – add the checksum to code or not.

```
build ()
```

```
static calculate_checksum (code)
```

```
get_fullcode ()
```

```
name = 'Code 39'
```

```
render (writer_options=None)
```

```
static validate (code)
```

```
class PZN (code, writer=None)
```

Bases: `steenzout.barcode.codex.Code39`

Initializes new German number for pharmaceutical products.

Parameters

- **pzn** (*str*) – code to render.
- **writer** (`writer.BaseWriter`) – instance of writer class to render the bar code.

```
build ()
```

```
static calculate_checksum (code)
```

```
code39 ()
```

```
digits = 6
```

```
get_fullcode ()
```

```
name = 'Pharmazentralnummer'
```

```
static validate (code)
```

```
check_code (code, name, allowed)
```

steenzout.barcode.ean module

Module: barcode.ean

Provided barcodes EAN-13, EAN-8, JAN

class EAN13 (*code*, *writer=None*)

Bases: *steenzout.barcode.base.Barcode*

Class for EAN13 bar codes.

checksum

int – EAN checksum.

Parameters

- **ean** (*str*) – the EAN number.
- **writer** (*writer.BaseWriter*) – instance of writer class to render the bar code.

build()

Builds the barcode pattern from *self.ean*.

Returns The pattern as string.

Return type (*str*)

static calculate_checksum (*code*)

Calculates a EAN-13 code checksum.

Parameters **code** (*str*) – EAN-13 code.

Returns the checksum for *self.ean*.

Return type (*integer*)

digits = 13

get_fullcode()

name = 'EAN-13'

render (*writer_options=None*)

to_ascii()

Returns an ascii representation of the barcode.

Returns ascii representation of the barcode.

Return type (*str*)

static validate (*code*)

Calculates a EAN-13 code checksum.

Parameters **code** (*str*) – EAN-13 code.

Raises

- *IllegalCharacterError* in case the bar code contains illegal characters.
- *ValueError* in case the bar code exceeds its maximum length or – if the checksum digit doesn't match.

class EAN8 (*code*, *writer=None*)

Bases: *steenzout.barcode.ean.EAN13*

Class for EAN-8 bar codes.

See EAN-13 for details.

Parameters `code` (str): EAN-8 number. `writer` (`writer.BaseWriter`): instance of writer class to render the bar code.

build()

Builds the barcode pattern from *self.ean*.

Returns string representation of the pattern.

Return type (str)

static calculate_checksum (*code*)

Calculates an EAN-8 code checksum.

Parameters `code` (*str*) – EAN-8 code.

Returns EAN checksum.

Return type (int)

digits = 8

name = 'EAN-8'

static validate (*code*)

Calculates a EAN-8 code checksum.

Parameters `code` (*str*) – EAN-8 code.

Raises

- `IllegalCharacterError` in case the bar code contains illegal characters.
- `ValueError` in case the bar code exceeds its maximum length or – if the checksum digit doesn't match..

EuropeanArticleNumber13

alias of [EAN13](#)

EuropeanArticleNumber8

alias of [EAN8](#)

class JAN (*code*, *writer=None*)

Bases: [steenzout.barcode.ean.EAN13](#)

Class for JAN bar codes.

Parameters

- `code` (*str*) – the jan number.
- `writer` (`writer.BaseWriter`) – instance of writer class to render the bar code.

name = 'JAN'

valid_country_codes = [450, 451, 452, 453, 454, 455, 456, 457, 458, 459, 490, 491, 492, 493, 494, 495, 496, 497, 498, 499]

JapanArticleNumber

alias of [JAN](#)

steenzout.barcode.errors module

Exceptions module.

exception BarcodeError (*msg*)

Bases: `exceptions.Exception`

Base class for steenzout.barcode exceptions.

exception BarcodeNotFoundError (*name*)

Bases: `steenzout.barcode.errors.BarcodeError`

Raised when an unknown barcode is requested.

exception IllegalCharacterError (*allowed*)

Bases: `steenzout.barcode.errors.BarcodeError`

Raised when a bar code contains illegal characters.

exception NumberOfDigitsError (*msg*)

Bases: `steenzout.barcode.errors.BarcodeError`

Raised when the number of digits do not match.

exception WrongCountryCodeError (*country*)

Bases: `steenzout.barcode.errors.BarcodeError`

Raised when a JAN (Japan Article Number) doesn't start with 450-459 or 490-499.

steenzout.barcode.factory module

Factory module.

create_instance (*name, code, writer=None*)

Return bar code instance.

Parameters

- **name** (*str*) – bar code name.
- **code** (*str*) – code text.
- **writer** (`steenzout.barcode.writer.Interface`) – writer instance.

Raises (`BarcodeNotFoundError`) – when the bar code encoding name is invalid or the encoding is not implemented.

Returns bar code instance.

Return type (`steenzout.barcode.base.Base`)

steenzout.barcode.helpers module

Helpers module.

sum_chars (*char, other*)

Sum the value of two string characters.

Parameters

- **char** (*char*) – string of the integer.
- **other** (*char*) – string of the integer.

Returns sum of the integer value of *x* and integer value of *y*.

Return type (`int`)

steenzout.barcode.isxn module

ISXN module.

Provided barcodes ISBN-13, ISBN-10, ISSN

This module provides some special codes, which are no standalone bar codes.

All codes where transformed to EAN-13 barcodes.

In every case, the checksum is new calculated.

Example:

```
>>> from steenzout.barcode import get_barcode
>>> ISBN = get_barcode('isbn10')
>>> isbn = ISBN('0132354187')
>>> unicode(isbn)
u'0132354187'
>>> isbn.get_fullcode()
u'9780132354189'
>>> # Test with wrong checksum
>>> isbn = ISBN('0132354180')
>>> unicode(isbn)
u'0132354187'
```

class ISBN10 (*code*, *writer=None*)

Bases: *steenzout.barcode.isxn.ISBN13*

Class for ISBN-10 bar codes.

Parameters

- **code** (*str*) – ISBN number.
- **writer** (*writer.BaseWriter*) – instance of writer class to render the bar code.

static calculate_checksum (*code*)

digits = 10

ean13 ()

Returns the EAN-13 representation of the ISBN-10 bar code.

Returns EAN-13 representation of the bar code.

Return type (EAN13)

isbn13 ()

Returns the ISBN-13 representation of the ISBN-10 bar code.

Returns ISBN-13 representation of the bar code.

Return type (*ISBN13*)

name = 'ISBN-10'

static validate (*code*)

class ISBN13 (*isbn*, *writer=None*)

Bases: *steenzout.barcode.ean.EAN13*

Class for ISBN-13 bar codes.

Parameters

- **isbn** (*str*) – ISBN number.
- **writer** (*writer.BaseWriter*) – instance of writer class to render the bar code.

static calculate_checksum (*code*)

digits = 13

name = 'ISBN-13'

static validate (*code*)

class ISSN (*issn, writer=None*)

Bases: *steenzout.barcode.ean.EAN13*

Class for ISSN bar codes.

This code is rendered as EAN-13 by prefixing it with 977 and adding 00 between code and checksum.

Parameters

- **issn** (*str*) – issn number.
- **writer** (*writer.BaseWriter*) – instance of writer class to render the bar code.

build ()

Builds the barcode pattern from *self.ean*.

Returns The pattern as string.

Return type (*str*)

static calculate_checksum (*code*)

digits = 8

ean13 ()

Returns the EAN-13 representation of the ISSN bar code.

Returns EAN-13 representation of the bar code.

Return type (*EAN13*)

name = 'ISSN'

static validate (*code*)

InternationalStandardBookNumber10

alias of *ISBN10*

InternationalStandardBookNumber13

alias of *ISBN13*

InternationalStandardSerialNumber

alias of *ISSN*

steenzout.barcode.metadata module

Metadata module.

steenzout.barcode.upc module

UPC module.

Provided barcodes UPC-A

class **UPCA** (*code*, *writer=None*)

Bases: *steenzout.barcode.base.Barcode*

Class for UPC-A bar codes.

Parameters

- **code** (*str*) – UPC-A bar code.
- **writer** (*writer.BaseWriter*) – instance of writer class to render the bar code.

build ()

Builds the bar code pattern.

Returns the bar code pattern.

Return type (*str*)

static **calculate_checksum** (*code*)

Calculates the UPC-A checksum.

Parameters **code** (*str*) – UPC-A code.

Returns UPC-A checksum.

Return type (*int*)

digits = 12

get_fullcode ()

name = 'UPC-A'

render (*writer_options=None*)

to_ascii ()

Returns an ASCII representation of the bar code.

Returns ASCII representation of the bar code.

Return type (*str*)

static **validate** (*code*)

Calculates a UPC-A code checksum.

Parameters **code** (*str*) – UPC-A code.

Raises

- *IllegalCharacterError* in case the bar code contains illegal characters.
- *ValueError* in case the bar code exceeds its maximum length or – if the checksum digit doesn't match.

UniversalProductCodeA

alias of *UPCA*

steenzout.barcode.writer module

class Base (*initialize=None, paint_module=None, paint_text=None, finish=None*)

Bases: `steenzout.object.Object`, `steenzout.barcode.writer.Interface`

Base class for all writers.

Initializes the basic writer options. Sub-classes can add more attributes and can set them directly or using `self.set_options(option=value)`.

Parameters

- **initialize** (*function*) – Callback for initializing the inheriting writer. Is called: `callback_initialize(raw_code)`
- **paint_module** (*function*) – Callback for painting one barcode module. Is called: `callback_paint_module(xpos, ypos, width, color)`
- **paint_text** (*function*) – Callback for painting the text under the barcode. Is called: `callback_paint_text(xpos, ypos)` using `self.text` as text.
- **finish** (*function*) – Callback for doing something with the completely rendered output. Is called: `return callback_finish()` and must return the rendered output.

calculate_size (*modules_per_line, number_of_lines, dpi=300*)

Calculates the size of the barcode in pixel.

Parameters

- **modules_per_line** (*int*) – number of modules in one line.
- **number_of_lines** (*int*) – number of lines of the barcode.
- **dpi** (*int*) – DPI to calculate.

Returns Width and height of the barcode in pixel.

Return type (tuple[int, int])

register_callback (*action, callback*)

Register one of the three callbacks if not given at instance creation.

Parameters

- **action** (*str*) – One of 'initialize', 'paint_module', 'paint_text', 'finish'.
- **callback** (*function*) – The callback function for the given action.

render (*code*)

Renders the barcode to whatever the inheriting writer provides, using the registered callbacks.

Parameters **code** (*list[str]*) – list of strings matching the writer spec (only contain 0 or 1).

save (*filename, output*)

See `Interface.save()`.

set_options (*options*)

Sets the given options as instance attributes (only if they are known).

Parameters **options** (*dict*) – All known instance attributes and more if the child class has defined them before this call.

DEFAULT_WRITER

alias of `SVG`

class ImageWriterBases: `steenzout.barcode.writer.Base`**save** (*filename*, *output*)See `Interface.save()`.**class Interface**Bases: `object`

Writer interface.

save (*filename*, *content*)Saves contents to *filename*.**Parameters**

- **filename** (*str*) – filename without extension.
- **content** (*str*) – output of rendering process.

Returns the filename.**Return type** (`str`)**class SVG**Bases: `steenzout.barcode.writer.Base`**save** (*filename*, *content*)See `Interface.save()`.**create_svg_object** ()

Returns a blank SVG document.

Returns XML document.**Return type** (`xml.dom.minidom.DocumentType`)**mm2px** (*value*, *dpi=300*)

Converts the given value in millimeters into dots per inch.

Parameters

- **value** (*float*) – value, in millimeters.
- **dpi** (*int*) – resolution, in dots per inch.

Returns value, in dots per inch.**Return type** (`float`)**pt2mm** (*value*)

Converts given value in points to millimeters.

Parameters **value** (*int*) – value in points.**Returns** value, in millimeters.**Return type** (`float`)**Module contents**

This package provides a simple way to create standard bar codes.

It needs no external packages to be installed, the bar codes are created as SVG objects.

If PIL (Python Imaging Library) is installed, the bar codes can also be rendered as images (all formats supported by PIL).

encodings()

Return bar code encodings available.

Returns available bar code encodings.

Return type (list[str])

formats()

Return image formats available.

Returns available image formats.

Return type (list['str'])

generate(name, code, writer=None, output=None, writer_options=None)

Generates a file containing an image of the bar code.

Parameters

- **name** (*str*) – bar code name.
- **code** (*str*) – bar code.
- **writer** (*steenzout.barcode.writer.Interface*) – writer instance.
- **output** (*str*) – filename of output.
- **writer_options** (*dict*) – options for the writer class.

Raises (BarcodeNotFoundError) – when the bar code encoding name is invalid or the encoding is not implemented.

version()

Returns package version.

Returns package version.

Return type (str)

Module contents

steenzout namespace package.

S

- `steenzout`, [27](#)
- `steenzout.barcode`, [26](#)
- `steenzout.barcode.base`, [16](#)
- `steenzout.barcode.charsets`, [16](#)
- `steenzout.barcode.charsets.code128`, [15](#)
- `steenzout.barcode.charsets.code39`, [15](#)
- `steenzout.barcode.charsets.ean`, [15](#)
- `steenzout.barcode.charsets.upc`, [16](#)
- `steenzout.barcode.cli`, [16](#)
- `steenzout.barcode.codex`, [17](#)
- `steenzout.barcode.ean`, [19](#)
- `steenzout.barcode.errors`, [20](#)
- `steenzout.barcode.factory`, [21](#)
- `steenzout.barcode.helpers`, [21](#)
- `steenzout.barcode.isxn`, [22](#)
- `steenzout.barcode.metadata`, [23](#)
- `steenzout.barcode.upc`, [24](#)
- `steenzout.barcode.writer`, [6](#)

B

Barcode (class in steenzout.barcode.base), 16
BarcodeError, 20
BarcodeNotFoundError, 21
Base (class in steenzout.barcode.writer), 6, 25
build() (Barcode method), 16
build() (Code128 method), 17
build() (Code39 method), 18
build() (EAN13 method), 19
build() (EAN8 method), 20
build() (ISSN method), 23
build() (PZN method), 18
build() (UPCA method), 24

C

calculate_checksum() (Code128 static method), 17
calculate_checksum() (Code39 static method), 18
calculate_checksum() (EAN13 static method), 19
calculate_checksum() (EAN8 static method), 20
calculate_checksum() (ISBN10 static method), 22
calculate_checksum() (ISBN13 static method), 23
calculate_checksum() (ISSN static method), 23
calculate_checksum() (PZN static method), 18
calculate_checksum() (steenzout.barcode.base.Barcode class method), 16
calculate_checksum() (UPCA static method), 24
calculate_size() (Base method), 6, 25
check_code() (in module steenzout.barcode.codex), 18
checksum (Barcode attribute), 16
checksum (EAN13 attribute), 19
code (Barcode attribute), 16
code (Code39 attribute), 3, 18
Code128 (class in steenzout.barcode.codex), 3, 17
Code39 (class in steenzout.barcode.codex), 3, 18
code39() (PZN method), 18
create_instance() (in module steenzout.barcode.factory), 21
create_svg_object() (in module steenzout.barcode.writer), 7, 26

D

DEFAULT_WRITER (in module steenzout.barcode.writer), 25
digits (Barcode attribute), 16
digits (EAN13 attribute), 19
digits (EAN8 attribute), 20
digits (ISBN10 attribute), 22
digits (ISBN13 attribute), 23
digits (ISSN attribute), 23
digits (PZN attribute), 18
digits (UPCA attribute), 24

E

EAN13 (class in steenzout.barcode.ean), 19
ean13() (ISBN10 method), 22
ean13() (ISSN method), 23
EAN8 (class in steenzout.barcode.ean), 19
encoded (Code128 attribute), 17
encodings() (in module steenzout.barcode), 27
EuropeanArticleNumber13 (in module steenzout.barcode.ean), 4, 20
EuropeanArticleNumber8 (in module steenzout.barcode.ean), 4, 20

F

formats() (in module steenzout.barcode), 27

G

generate() (in module steenzout.barcode), 27
get_fullcode() (Barcode method), 16
get_fullcode() (Code128 method), 17
get_fullcode() (Code39 method), 18
get_fullcode() (EAN13 method), 19
get_fullcode() (PZN method), 18
get_fullcode() (UPCA method), 24

I

IllegalCharacterError, 21
ImageWriter (class in steenzout.barcode.writer), 25

Interface (class in steenzout.barcode.writer), 7, 26

InternationalStandardBookNumber10 (in module steenzout.barcode.isxn), 4, 23

InternationalStandardBookNumber13 (in module steenzout.barcode.isxn), 4, 23

InternationalStandardSerialNumber (in module steenzout.barcode.isxn), 4, 23

ISBN10 (class in steenzout.barcode.isxn), 22

ISBN13 (class in steenzout.barcode.isxn), 22

isbn13() (ISBN10 method), 22

ISSN (class in steenzout.barcode.isxn), 23

J

JAN (class in steenzout.barcode.ean), 20

JapanArticleNumber (in module steenzout.barcode.ean), 4, 20

M

mm2px() (in module steenzout.barcode.writer), 7, 26

N

name (Barcode attribute), 16

name (Code128 attribute), 17

name (Code39 attribute), 18

name (EAN13 attribute), 19

name (EAN8 attribute), 20

name (ISBN10 attribute), 22

name (ISBN13 attribute), 23

name (ISSN attribute), 23

name (JAN attribute), 20

name (PZN attribute), 18

name (UPCA attribute), 24

NumberOfDigitsError, 21

P

pt2mm() (in module steenzout.barcode.writer), 7, 26

PZN (class in steenzout.barcode.codex), 3, 18

R

raw (Barcode attribute), 16

register_callback() (Base method), 6, 25

render() (Barcode method), 16

render() (Base method), 7, 25

render() (Code128 method), 18

render() (Code39 method), 18

render() (EAN13 method), 19

render() (UPCA method), 24

S

save() (Barcode method), 17

save() (Base method), 7, 25

save() (ImageWriter method), 26

save() (Interface method), 7, 26

save() (SVG method), 26

set_options() (Base method), 7, 25

steenzout (module), 27

steenzout.barcode (module), 26

steenzout.barcode.base (module), 16

steenzout.barcode.charsets (module), 16

steenzout.barcode.charsets.code128 (module), 15

steenzout.barcode.charsets.code39 (module), 15

steenzout.barcode.charsets.ean (module), 15

steenzout.barcode.charsets.upc (module), 16

steenzout.barcode.cli (module), 16

steenzout.barcode.codex (module), 17

steenzout.barcode.ean (module), 19

steenzout.barcode.errors (module), 20

steenzout.barcode.factory (module), 21

steenzout.barcode.helpers (module), 21

steenzout.barcode.isxn (module), 22

steenzout.barcode.metadata (module), 23

steenzout.barcode.upc (module), 24

steenzout.barcode.writer (module), 6, 25

sum_chars() (in module steenzout.barcode.helpers), 21

SVG (class in steenzout.barcode.writer), 26

T

to_ascii() (Barcode method), 17

to_ascii() (EAN13 method), 19

to_ascii() (UPCA method), 24

U

UniversalProductCodeA (in module steenzout.barcode.upc), 4, 24

UPCA (class in steenzout.barcode.upc), 24

V

valid_country_codes (JAN attribute), 20

validate() (Code128 static method), 18

validate() (Code39 static method), 18

validate() (EAN13 static method), 19

validate() (EAN8 static method), 20

validate() (ISBN10 static method), 22

validate() (ISBN13 static method), 23

validate() (ISSN static method), 23

validate() (PZN static method), 18

validate() (steenzout.barcode.base.Barcode class method), 17

validate() (UPCA static method), 24

version() (in module steenzout.barcode), 27

W

write() (Barcode method), 17

writer, 4

writer (Barcode attribute), 17

writer (Code39 attribute), 3, 18

`writer_options`, [4](#)

`WrongCountryCodeError`, [21](#)