

- Update `wand.display` to use Python3 compatible `print()` function. [#395 by Tero Vuotila]

Version 0.5.0

Released on January 1, 2019.

- Support for ImageMagick-7.
- Improved support for 32-bit systems.
- Improved support for non-Q16 libraries.
- Removed `README.rst` from `setup.py`'s `data_files`. [#336]
- Improved `EXIF:ORIENTATION` handling. [#364 by M. Skrzypek]
- Tolerate failures while accessing `wand.api`. [#220 by Utkarsh Upadhyay]
- Added support for Image Artifacts through `Image.artifacts`. [#369]
- Added optional stroke color/width parameters for `Font`.
- Image layers support (#22)
 - Added `Image.coalesce()` method.
 - Added `Image.deconstruct` method.
 - Added `Image.dispose` property.
 - Added `Image.optimize_layers()` method.
 - Added `Image.optimize_transparency()` method.
- Implemented `__array_interface__()` for NumPy [#65]
- Migrated the following methods & attributes from `Image` to `BaseImage` for a more uniformed code-base.
 - `Image.compression`
 - `Image.format`
 - `Image.auto_orient()`
 - `Image.border()`
 - `Image.contrast_stretch()`
 - `Image.gamma()`
 - `Image.level()`
 - `Image.linear_stretch()`
 - `Image.normalize()`
 - `Image.strip()`
 - `Image.transpose()`
 - `Image.transverse()`
 - `Image.trim()`
- Added `Image.clut()` method.
- Added `Image.concat()` method. [#177]
- Added `Image.deskew()` method.

- Added `Image.despeckle()` method.
- Added `Image.edge()` method.
- Added `Image.emboss()` method. [#196]
- Added `Image.enhance()` method. [#132]
- Added `Image.export_pixels()` method.
- Added `Image.import_pixels()` method.
- Added `Image.morphology()` method. [#132]
- Added `Image.posterize()` method.
- Added `Image.shade()` method.
- Added `Image.shadow()` method.
- Added `Image.sharpen()` method. [#132]
- Added `Image.shave()` method.
- Added `Image.unique_colors()` method.
- Method `Drawing.draw()` now accepts `BaseImage` for folks extended classes.
- Added `Image.loop` property. [#227]
- Fixed `SingleImage.delay` property. [#153]
- Attribute `Image.font_antialias` has been deprecated in favor of `Image.antialias`. [#218]
- Fixed ordering of `COMPRESSION_TYPES` based on ImageMagick version. [#309]
- Fixed drawing on `SingleImage`. [#289]
- Fixed wrapping issue for larger offsets when using `gravity` kwarg in `Image.crop()` method. [#367]

3.16.2 0.4 series

Version 0.4.5

Released on November 12, 2018.

- Improve library searching when `MAGICK_HOME` environment variable is set. [#320 by Chase Anderson]
- Fixed misleading `TypeError: object of type 'NoneType' has no len()` during destroy routines. [#346 by Carey Metcalfe]
- Added `Image.blur()` method (`MagickBlurImage()`). [#311 by Alexander Karpinsky]
- Added `Image.extent()` method (`MagickExtentImage()`). [#233 by Jae-Myoung Yu]
- Added `Image.resample()` method (`MagickResampleImage()`). [#244 by Zio Tibia]

Version 0.4.4

Released on October 22, 2016.

- Added `BaseError`, `BaseWarning`, and `BaseFatalError`, base classes for domains. [#292]
- Fixed `TypeError` during parsing version caused by format change of ImageMagick version string (introduced by 6.9.6.2). [#310, Debian bug report #841548]

- Properly fixed again memory-leak when accessing images constructed in `Image.sequence[]`. It had still leaked memory in the case an image is not closed using `with` but manual `wand.resource.Resource.destroy()/wand.image.Image.close()` method call. [#237]

Version 0.4.3

Released on June 1, 2016.

- Fixed `repr()` for empty `Image` objects. [#265]
- Added `Image.compare()` method (`MagickCompareImages()`). [#238, #268 by Gyusun Yeom]
- Added `Image.page` and related properties for virtual canvas handling. [#284 by Dan Harrison]
- Added `Image.merge_layers()` method (`MagickMergeImageLayers()`). [#281 by Dan Harrison]
- Fixed `OSError` during import `libc.dylib` due to El Capitan's SIP protection. [#275 by Ramesh Dharan]

Version 0.4.2

Released on November 30, 2015.

- Fixed `ImportError` on MSYS2. [#257 by Eon Jeong]
- Added `Image.quantize()` method (`MagickQuantizeImage()`). [#152 by Kang Hyojun, #262 by Jeong YunWon]
- Added `Image.transform_colorspace()` `quantize` (`MagickTransformImageColorspace()`). [#152 by Adrian Jung, #262 by Jeong YunWon]
- Now ImageMagick DLL can be loaded on Windows even if its location is stored in the registry. [#261 by Roeland Schoukens]
- Added depth parameter to `Image` constructor. The depth, width and height parameters can be used with the filename, file and blob parameters to load raw pixel data. [#261 by Roeland Schoukens]

Version 0.4.1

Released on August 3, 2015.

- Added `Image.auto_orient()` that fixes orientation by checking EXIF tags.
- Added `Image.transverse()` method (`MagickTransverseImage()`).
- Added `Image.transpose()` method (`MagickTransposeImage()`).
- Added `Image.evaluate()` method.
- Added `Image.frame()` method.
- Added `Image.function()` method.
- Added `Image.fx()` expression method.
- Added gravity options in `Image.crop()` method. [#222 by Eric McConville]
- Added `Image.matte_color` property.
- Added `Image.virtual_pixel` property.
- Added `Image.distort()` method.
- Added `Image.contrast_stretch()` method.

- Added `Image.gamma()` method.
- Added `Image.linear_stretch()` method.
- Additional support for `Image.alpha_channel`.
- Additional query functions have been added to `wand.version` API. [#120]
 - Added `configure_options()` function.
 - Added `fonts()` function.
 - Added `formats()` function.
- Additional IPython support. [#117]
 - Render RGB `Color` preview.
 - Display each frame in image `Sequence`.
- Fixed memory-leak when accessing images constructed in `Image.sequence[]`. [#237 by Eric McConville]
- Fixed Windows memory-deallocate errors on `wand.drawing` API. [#226 by Eric McConville]
- Fixed `ImportError` on FreeBSD. [#252 by Pellaeon Lin]

Version 0.4.0

Released on February 20, 2015.

See also:

whatsnew/0.4 This guide introduces what's new in Wand 0.4.

- Complete `wand.drawing` API. The whole work was done by Eric McConville. Huge thanks for his effort! [#194 by Eric McConville]
 - Added `Drawing.arc()` method (*Arc*).
 - Added `Drawing.bezier()` method (*Bezier*).
 - Added `Drawing.circle()` method (*Circle*).
 - *Color & Matte*
 - * Added `wand.drawing.PAINT_METHOD_TYPES` constant.
 - * Added `Drawing.color()` method.
 - * Added `Drawing.matte()` method.
 - Added `Drawing.composite()` method (*Composite*).
 - Added `Drawing.ellipse()` method (*Ellipse*).
 - *Paths*
 - * Added `path_start()` method.
 - * Added `path_finish()` method.
 - * Added `path_close()` method.
 - * Added `path_curve()` method.
 - * Added `path_curve_to_quadratic_bezier()` method.
 - * Added `path_elliptic_arc()` method.

- * Added `path_horizontal_line()` method.
 - * Added `path_line()` method.
 - * Added `path_move()` method.
 - * Added `path_vertical_line()` method.
 - Added `Drawing.point()` method (*Point*).
 - Added `Drawing.polygon()` method (*Polygon*).
 - Added `Drawing.polyline()` method (*Polyline*).
 - *Push & Pop*
 - * Added `push()` method.
 - * Added `push_clip_path()` method.
 - * Added `push_defs()` method.
 - * Added `push_pattern()` method.
 - * Added `clip_path` property.
 - * Added `set_fill_pattern_url()` method.
 - * Added `set_stroke_pattern_url()` method.
 - * Added `pop()` method.
 - Added `Drawing.rectangle()` method (*Rectangles*).
 - Added `stroke_dash_array` property.
 - Added `stroke_dash_offset` property.
 - Added `stroke_line_cap` property.
 - Added `stroke_line_join` property.
 - Added `stroke_miter_limit` property.
 - Added `stroke_opacity` property.
 - Added `stroke_width` property.
 - Added `fill_opacity` property.
 - Added `fill_rule` property.
- Error message of `MissingDelegateError` raised by `Image.liquid_rescale()` became nicer.

3.16.3 0.3 series

Version 0.3.9

Released on December 20, 2014.

- Added 'pdf:use-cropbox' option to `Image.options` dictionary (and `OPTIONS` constant). [#185 by Christoph Neuroth]
- Fixed a bug that exception message was `bytes` instead of `str` on Python 3.
- The `size` parameter of `Font` class becomes optional. Its default value is 0, which means *autosized*. [#191 by Cha, Hojeong]

- Fixed a bug that `Image.read()` had tried using `MagickReadImageFile()` even when the given file object has no `mode` attribute. [#205 by Stephen J. Fuhry]

Version 0.3.8

Released on August 3, 2014.

- Fixed a bug that transparent background becomes filled with white when SVG is converted to other bitmap image format like PNG. [#184]
- Added `Image.negate()` method. [#174 by Park Joon-Kyu]
- Fixed a segmentation fault on `Image.modulate()` method. [#173 by Ted Fung, #158]
- Added suggestion to install freetype also if Homebrew is used. [#141]
- Now `image/x-gif` also is determined as `animation`. [#181 by Juan-Pablo Scaletti]

Version 0.3.7

Released on March 25, 2014.

- A hotfix of debug prints made at 0.3.6.

Version 0.3.6

Released on March 23, 2014.

- Added `Drawing.rectangle()` method. *Now you can draw rectangles.* [#159]
- Added `Image.compression` property. [#171]
- Added `contextlib.nested()` function to `wand.compat` module.
- Fixed `UnicodeEncodeError` when `Drawing.text()` method gives Unicode text argument in Python 2. [#163]
- Now it now allows to use Wand when Python is invoked with the `-OO` flag. [#169 by Samuel Maudo]

Version 0.3.5

Released on September 13, 2013.

- Fix segmentation fault on `Image.save()` method. [#150]

Version 0.3.4

Released on September 9, 2013.

- Added `Image.modulate()` method. [#134 by Dan P. Smith]
- Added `Image.colorspace` property. [#135 by Volodymyr Kuznetsov]
- Added `Image.unsharp_mask()` method. [#136 by Volodymyr Kuznetsov]
- Added 'jpeg:sampling-factor' option to `Image.options` dictionary (and `OPTIONS` constant). [#137 by Volodymyr Kuznetsov]
- Fixed ImageMagick shared library resolution on Arch Linux. [#139, #140 by Sergey Tereschenko]

- Added `Image.sample()` method. [#142 by Michael Allen]
- Fixed a bug that `Image.save()` preserves only one frame of the given animation when file-like object is passed. [#143, #145 by Michael Allen]
- Fixed searching of ImageMagick shared library with HDR support enabled. [#148, #149 by Lipin Dmitriy]

Version 0.3.3

Released on August 4, 2013. It's author's birthday.

- Added `Image.gaussian_blur()` method.
- Added `Drawing.stroke_color` property. [#129 by Zeray Rice]
- Added `Drawing.stroke_width` property. [#130 by Zeray Rice]
- Fixed a memory leak of `Color` class. [#127 by Wieland Morgenstern]
- Fixed a bug that `Image.save()` to stream truncates data. [#128 by Michael Allen]
- Fixed broken `display()` on Python 3. [#126]

Version 0.3.2

Released on July 11, 2013.

- Fixed incorrect encoding of filenames. [#122]
- Fixed key type of `Image.metadata` dictionary to `str` from `bytes` in Python 3.
- Fixed CentOS compatibility [#116, #124 by Pierre Vanliefland]
 - Made `DrawSetTextInterlineSpacing()` and `DrawGetTextInterlineSpacing()` optional.
 - Added exception in drawing API when trying to use `DrawSetTextInterlineSpacing()` and `DrawGetTextInterlineSpacing()` functions when they are not available.
 - Added `WandLibraryVersionError` class for library versions issues.

Version 0.3.1

Released on June 23, 2013.

- Fixed `ImportError` on Windows.

Version 0.3.0

Released on June 17, 2013.

See also:

whatsnew/0.3 This guide introduces what's new in Wand 0.3.

- Now also works on Python 2.6, 2.7, and 3.2 or higher.
- Added `wand.drawing` module. [#64 by Adrian Jung]
- Added `Drawing.get_font_metrics()` method. [#69, #71 by Cha, Hojeong]
- Added `Image.caption()` method. [#74 by Cha, Hojeong]

- Added optional `color` parameter to `Image.trim()` method.
- Added `Image.border()` method. [2496d37f75d75e9425f95dde07033217dc8afefc by Jae-Myoung Yu]
- Added resolution parameter to `Image.read()` method and the constructor of `Image`. [#75 by Andrey Antukh]
- Added `Image.liquid_rescale()` method which does *seam carving*. See also *Seam carving (also known as content-aware resizing)*.
- Added `Image.metadata` immutable mapping attribute and `Metadata` mapping type for it. [#56 by Michael Elovsikh]
- Added `Image.channel_images` immutable mapping attribute and `ChannelImageDict` mapping for it.
- Added `Image.channel_depths` immutable mapping attribute and `ChannelDepthDict` mapping for it.
- Added `Image.composite_channel()` method.
- Added `Image.read()` method. [#58 by Piotr Florczyk]
- Added `Image.resolution` property. [#58 by Piotr Florczyk]
- Added `Image.blank()` method. [#60 by Piotr Florczyk]
- Fixed several memory leaks. [#62 by Mitch Lindgren]
- Added `ImageProperty` mixin class to maintain a weak reference to the parent image.
- Renamed `wand.image.COMPOSITE_OPS` to `COMPOSITE_OPERATORS`.
- Now it shows helpful error message when ImageMagick library cannot be found.
- Added IPython-specialized formatter.
- Added `QUANTUM_DEPTH` constant.
- Added these properties to `Color` class:
 - `red_quantum`
 - `green_quantum`
 - `blue_quantum`
 - `alpha_quantum`
 - `red_int8`
 - `green_int8`
 - `blue_int8`
 - `alpha_int8`
- Added `Image.normalize()` method. [#95 by Michael Curry]
- Added `Image.transparent_color()` method. [#98 by Lionel Koenig]
- Started supporting resizing and cropping of GIF images. [#88 by Bear Dong, #112 by Taeho Kim]
- Added `Image.flip()` method.
- Added `Image.flop()` method.
- Added `Image.orientation` property. [88574468a38015669dae903185fb328abdd717c0 by Taeho Kim]
- `wand.resource.DestroyedResourceError` becomes a subtype of `wand.exceptions.WandException`.

- `Color` is now hashable, so can be used as a key of dictionaries, or an element of sets. [#114 by klutzy]
- `Color` has `normalized_string` property.
- `Image` has `histogram` dictionary.
- Added optional `fuzz` parameter to `Image.trim()` method. [#113 by Evaldo Junior]

3.16.4 0.2 series

Version 0.2.4

Released on May 28, 2013.

- Fix `NameError` in `Resource.resource` setter. [#89 forwarded from Debian bug report #699064 by Jakub Wilk]
- Fix the problem of library loading for Mac with Homebrew and Arch Linux. [#102 by Roel Gerrits, #44]

Version 0.2.3

Released on January 25, 2013.

- Fixed a bug that `Image.transparentize()` method (and `Image.watermark()` method which internally uses it) didn't work.
- Fixed segmentation fault occurred when `Color.red`, `Color.green`, or `Color.blue` is accessed.
- Added `Color.alpha` property.
- Fixed a bug that format converting using `Image.format` property or `Image.convert()` method doesn't correctly work to save blob.

Version 0.2.2

Released on September 24, 2012.

- A compatibility fix for FreeBSD. [Patch by Olivier Duchateau]
- Now `Image` can be instantiated without any opening. Instead, it can take `width/height` and `background`. [#53 by Michael Elovskikh]
- Added `Image.transform()` method which is a convenience method accepting geometry strings to perform cropping and resizing. [#50 by Mitch Lindgren]
- Added `Image.units` property. [#45 by Piotr Florczyk]
- Now `Image.resize()` method raises a proper error when it fails for any reason. [#41 by Piotr Florczyk]
- Added `Image.type` property. [#33 by Yauhen Yakimovich, #42 by Piotr Florczyk]

Version 0.2.1

Released on August 19, 2012. Beta version.

- Added `Image.trim()` method. [#26 by Jökull Sólberg Auðunsson]
- Added `Image.depth` property. [#31 by Piotr Florczyk]
- Now `Image` can take an optional `format` hint. [#32 by Michael Elovskikh]

- Added `Image.alpha_channel` property. [#35 by Piotr Florczyk]
- The default value of `Image.resize()`'s filter option has changed from 'triangle' to 'undefined'. [#37 by Piotr Florczyk]
- Added version data of the linked ImageMagick library into `wand.version` module:
 - `MAGICK_VERSION` (`GetMagickVersion()`)
 - `MAGICK_VERSION_INFO` (`GetMagickVersion()`)
 - `MAGICK_VERSION_NUMBER` (`GetMagickVersion()`)
 - `MAGICK_RELEASE_DATE` (`GetMagickReleaseDate()`)
 - `MAGICK_RELEASE_DATE_STRING` (`GetMagickReleaseDate()`)

Version 0.2.0

Released on June 20, 2012. Alpha version.

- Added `Image.transparentize()` method. [#19 by Jeremy Axmacher]
- Added `Image.composite()` method. [#19 by Jeremy Axmacher]
- Added `Image.watermark()` method. [#19 by Jeremy Axmacher]
- Added `Image.quantum_range` property. [#19 by Jeremy Axmacher]
- Added `Image.reset_coords()` method and `reset_coords` option to `Image.rotate()` method. [#20 by Juan Pablo Scaletti]
- Added `Image.strip()` method. [#23 by Dmitry Vukolov]
- Added `Image.compression_quality` property. [#23 by Dmitry Vukolov]
- Now the current version can be found from the command line interface: `python -m wand.version`.

3.16.5 0.1 series

Version 0.1.10

Released on May 8, 2012. Still alpha version.

- So many Windows compatibility issues are fixed. [#14 by John Simon]
- Added `wand.api.libmagick`.
- Fixed a bug that raises `AttributeError` when it's trying to warn. [#16 by Tim Dettrick]
- Now it throws `ImportError` instead of `AttributeError` when the shared library fails to load. [#17 by Kieran Spear]
- Fixed the example usage on index page of the documentation. [#18 by Jeremy Axmacher]

Version 0.1.9

Released on December 23, 2011. Still alpha version.

- Now `wand.version.VERSION_INFO` becomes `tuple` and `wand.version.VERSION` becomes a string.

- Added `Image.background_color` property.
- Added `==` operator for `Image` type.
- Added `hash()` support of `Image` type.
- Added `Image.signature` property.
- Added `wand.display` module.
- Changed the theme of Sphinx documentation.
- Changed the start example of the documentation.

Version 0.1.8

Released on December 2, 2011. Still alpha version.

- Wrote some guide documentations: *Reading images*, *Writing images* and *Resizing and cropping*.
- Added `Image.rotate()` method for in-place rotation.
- Made `Image.crop()` to raise proper `ValueError` instead of `IndexError` for invalid width/height arguments.
- Changed the type of `Image.resize()` method's `blur` parameter from `numbers.Rational` to `numbers.Real`.
- Fixed a bug of raising `ValueError` when invalid `filter` has passed to `Image.resize()` method.

Version 0.1.7

Released on November 10, 2011. Still alpha version.

- Added `Image.mimetype` property.
- Added `Image.crop()` method for in-place crop.

Version 0.1.6

Released on October 31, 2011. Still alpha version.

- Removed a side effect of `Image.make_blob()` method that changes the image format silently.
- Added `Image.format` property.
- Added `Image.convert()` method.
- Fixed a bug about Python 2.6 compatibility.
- Use the internal representation of `PixelWand` instead of the string representation for `Color` type.

Version 0.1.5

Released on October 28, 2011. Slightly mature alpha version.

- Now `Image` can read Python file objects by `file` keyword argument.
- Now `Image.save()` method can write into Python file objects by `file` keyword argument.
- `Image.make_blob()`'s `format` argument becomes omissible.

Version 0.1.4

Released on October 27, 2011. Hotfix of the malformed Python package.

Version 0.1.3

Released on October 27, 2011. Slightly mature alpha version.

- Pixel getter for *Image*.
- Row getter for *Image*.
- Mac compatibility.
- Windows compatibility.
- 64-bit processor compatibility.

Version 0.1.2

Released on October 16, 2011. Still alpha version.

- *Image* implements iterable interface.
- Added *wand.color* module.
- Added the abstract base class of all Wand resource objects: *wand.resource.Resource*.
- *Image* implements slicing.
- Cropping *Image* using its slicing operator.

Version 0.1.1

Released on October 4, 2011. Still alpha version.

- Now it handles errors and warnings properly and in natural way of Python.
- Added *Image.make_blob()* method.
- Added *blob* parameter into *Image* constructor.
- Added *Image.resize()* method.
- Added *Image.save()* method.
- Added *Image.clone()* method.
- Drawed the pretty logo picture (thanks to Hyojin Choi).

Version 0.1.0

Released on October 1, 2011. Very alpha version.

3.17 Talks and Presentations

3.17.1 Talks in 2012

- Lightning talk at Python Korea November 2012

4.1 wand — Simple MagickWand API binding for Python

4.1.1 wand.image — Image objects

Opens and manipulates images. Image objects can be used in `with` statement, and these resources will be automatically managed (even if any error happened):

```
with Image(filename='pikachu.png') as i:
    print('width =', i.width)
    print('height =', i.height)
```

`wand.image.ALPHA_CHANNEL_TYPES = ('undefined', 'activate', 'background', 'copy', 'deactivate', 'discrete', 'extract', 'off', 'on', 'opaque', 'reset', 'set')` The list of alpha channel types

- 'undefined'
- 'activate'
- 'background'
- 'copy'
- 'deactivate'
- 'discrete' - Only available in ImageMagick-7
- 'extract'
- 'off' - Only available in ImageMagick-7
- 'on' - Only available in ImageMagick-7
- 'opaque'
- 'reset' - Only available in ImageMagick-6
- 'set'

- 'shape'
- 'transparent'
- 'flatten' - Only available in ImageMagick-6
- 'remove'

See also:

ImageMagick Image Channel Describes the SetImageAlphaChannel method which can be used to modify alpha channel. Also describes AlphaChannelType

wand.image.CHANNELS = {'all_channels': 134217727, 'alpha': 8, 'black': 32, 'blue': 4,
(dict) The dictionary of channel types.

- 'undefined'
- 'red'
- 'gray'
- 'cyan'
- 'green'
- 'magenta'
- 'blue'
- 'yellow'
- 'alpha'
- 'opacity'
- 'black'
- 'index'
- 'composite_channels'
- 'all_channels'
- 'true_alpha'
- 'rgb_channels'
- 'gray_channels'
- 'sync_channels'
- 'default_channels'

See also:

ImageMagick Color Channels Lists the various channel types with descriptions of each

wand.image.COLORSPACE_TYPES = ('undefined', 'rgb', 'gray', 'transparent', 'hsla', 'lab', 'x',
(tuple) The list of colorspaces.

- 'undefined'
- 'rgb'
- 'gray'
- 'transparent'

- 'ohta'
- 'lab'
- 'xyz'
- 'ycbcr'
- 'ycc'
- 'yiq'
- 'ypbpr'
- 'yuv'
- 'cmyk'
- 'srgb'
- 'hsb'
- 'hsl'
- 'hwb'
- 'rec601luma' - Only available with ImageMagick-6
- 'rec601ycbcr'
- 'rec709luma' - Only available with ImageMagick-6
- 'rec709ycbcr'
- 'log'
- 'cmy'
- 'luv'
- 'hcl'
- 'lch'
- 'lms'
- 'lchab'
- 'lchuv'
- 'scrgb'
- 'hsi'
- 'hsv'
- 'hclp'
- 'xyy' - Only available with ImageMagick-7
- 'ydbdr'

See also:

ImageMagick Color Management Describes the ImageMagick color management operations

New in version 0.3.4.

`wand.image.COMPARE_METRICS = ('undefined', 'absolute', 'mean_absolute', 'mean_error_per_pixel')`
([tuple](#)) The list of compare metric types

- 'copy_opacity' - Only available with ImageMagick-6
- 'copy_red'
- 'copy_yellow'
- 'darken'
- 'darken_intensity' - Only available with ImageMagick-7
- 'dst_atop'
- 'dst'
- 'dst_in'
- 'dst_out'
- 'dst_over'
- 'difference'
- 'displace'
- 'dissolve'
- 'distort' - Only available with ImageMagick-7
- 'divide' - Only available with ImageMagick-6
- 'exclusion'
- 'hard_light'
- 'head_mix' - Only available with ImageMagick-7
- 'hue'
- 'in'
- 'intensity' - Only available with ImageMagick-7
- 'lighten'
- 'lighten_intensity' - Only available with ImageMagick-7
- 'linear_burn' - Only available with ImageMagick-7
- 'linear_dodge' - Only available with ImageMagick-7
- 'linear_light'
- 'luminize'
- 'mathematics' - Only available with ImageMagick-7
- 'minus' - Only available with ImageMagick-6
- 'minus_dst' - Only available with ImageMagick-7
- 'minus_src' - Only available with ImageMagick-7
- 'modulate'
- 'modulate_add' - Only available with ImageMagick-7
- 'modulate_subtract' - Only available with ImageMagick-7
- 'multiply'
- 'out'

- 'over'
- 'overlay'
- 'pegtop_light' - Only available with ImageMagick-7
- 'plus'
- 'replace'
- 'saturate'
- 'screen'
- 'soft_light'
- 'src_atop'
- 'src'
- 'src_in'
- 'src_out'
- 'src_over'
- 'subtract' - Only available with ImageMagick-6
- 'threshold'
- 'vivid_light' - Only available with ImageMagick-7
- 'xor'

Changed in version 0.3.0: Renamed from `COMPOSITE_OPS` to `COMPOSITE_OPERATORS`.

See also:

Compositing Images ImageMagick v6 Examples Image composition is the technique of combining images that have, or do not have, transparency or an alpha channel. This is usually performed using the IM **composite** command. It may also be performed as either part of a larger sequence of operations or internally by other image operators.

ImageMagick Composition Operators Demonstrates the results of applying the various composition composition operators.

`wand.image.COMPRESSION_TYPES = ('undefined', 'no', 'bzip', 'dxt1', 'dxt3', 'dxt5', 'fax',`
(tuple) The list of `Image.compression` types.

New in version 0.3.6.

Changed in version 0.5.0: Support for ImageMagick-6 & ImageMagick-7

`wand.image.DISPOSE_TYPES = ('undefined', 'none', 'background', 'previous')`
(tuple) The list of `BaseImage.dispose` types.

New in version 0.5.0.

`wand.image.DISTORTION_METHODS = ('undefined', 'affine', 'affine_projection', 'scale_rotate'`
(tuple) The list of `BaseImage.distort()` methods.

- 'undefined'
- 'affine'
- 'affine_projection'
- 'scale_rotate_translate'

- 'perspective'
- 'perspective_projection'
- 'bilinear_forward'
- 'bilinear_reverse'
- 'polynomial'
- 'arc'
- 'polar'
- 'depolar'
- 'cylinder_2_plane'
- 'plane_2_cylinder'
- 'barrel'
- 'barrel_inverse'
- 'shepards'
- 'resize'
- 'sentinel'

New in version 0.4.1.

`wand.image.DITHER_METHODS = ('undefined', 'no', 'riemersma', 'floyd_steinberg')`
(tuple) The list of Dither methods.

- 'undefined'
- 'no'
- 'riemersma'
- 'floyd_steinberg'

New in version 0.5.0.

`wand.image.EVALUATE_OPS = ('undefined', 'add', 'and', 'divide', 'leftshift', 'max', 'min',`
(tuple) The list of evaluation operators

- 'undefined'
- 'abs'
- 'add'
- 'addmodulus'
- 'and'
- 'cosine'
- 'divide'
- 'exponential'
- 'gaussiannoise'
- 'impulsenoise'
- 'laplaciannoise'
- 'leftshift'

- 'log'
- 'max'
- 'mean'
- 'median'
- 'min'
- 'multiplicativenoise'
- 'multiply'
- 'or'
- 'poissonnoise'
- 'pow'
- 'rightshift'
- 'set'
- 'sine'
- 'subtract'
- 'sum'
- 'threshold'
- 'thresholdblack'
- 'thresholdwhite'
- 'uniformnoise'
- 'xor'

See also:

ImageMagick Image Evaluation Operators Describes the `MagickEvaluateImageChannel` method and lists the various evaluations operators

`wand.image.FILTER_TYPES = ('undefined', 'point', 'box', 'triangle', 'hermite', 'hanning', 'hamming', 'blackman', 'gaussian', 'quadratic', 'cubic')` The list of filter types.

- 'undefined'
- 'point'
- 'box'
- 'triangle'
- 'hermite'
- 'hanning'
- 'hamming'
- 'blackman'
- 'gaussian'
- 'quadratic'
- 'cubic'

- 'catrom'
- 'mitchell'
- 'jinc'
- 'sinc'
- 'sincfast'
- 'kaiser'
- 'welsh'
- 'parzen'
- 'bohman'
- 'bartlett'
- 'lagrange'
- 'lanczos'
- 'lanczossharp'
- 'lanczos2'
- 'lanczos2sharp'
- 'robidoux'
- 'robidouxsharp'
- 'cosine'
- 'spline'
- 'sentinel'

See also:

ImageMagick Resize Filters Demonstrates the results of resampling images using the various resize filters and blur settings available in ImageMagick.

`wand.image.FUNCTION_TYPES = ('undefined', 'polynomial', 'sinusoid', 'arcsin', 'arctan')`
(tuple) The list of `Image.function` types.

- 'undefined'
- 'arcsin'
- 'arctan'
- 'polynomial'
- 'sinusoid'

`wand.image.GRAVITY_TYPES = ('forget', 'north_west', 'north', 'north_east', 'west', 'center')`
(tuple) The list of `gravity` types.

- 'forget'
- 'north_west'
- 'north'
- 'north_east'

- 'west'
- 'center'
- 'east'
- 'south_west'
- 'south'
- 'south_east'

New in version 0.3.0.

`wand.image.IMAGE_LAYER_METHOD = ('undefined', 'coalesce', 'compareany', 'compareclear', 'compareoverlay')`
(tuple) The list of methods for `merge_layers()` and `compare_layers()`.

- 'undefined'
- 'coalesce'
- 'compareany' - Only used for `compare_layers()`.
- 'compareclear' - Only used for `compare_layers()`.
- 'compareoverlay' - Only used for `compare_layers()`.
- 'dispose'
- 'optimize'
- 'optimizeimage'
- 'optimizeplus'
- 'optimizetrans'
- 'removedups'
- 'removezero'
- 'composite'
- 'merge' - Only used for `merge_layers()`.
- 'flatten' - Only used for `merge_layers()`.
- 'mosaic' - Only used for `merge_layers()`.
- 'trimbounds' - Only used for `merge_layers()`.

New in version 0.4.3.

`wand.image.IMAGE_TYPES = ('undefined', 'bilevel', 'grayscale', 'grayscalematte', 'palette')`
(tuple) The list of image types

- 'undefined'
- 'bilevel'
- 'grayscale'
- 'grayscalealpha' - Only available with ImageMagick-7
- 'grayscalematte' - Only available with ImageMagick-6
- 'palette'
- 'palettealpha' - Only available with ImageMagick-7
- 'palettematte' - Only available with ImageMagick-6

- 'truecolor'
- 'truecoloralpha' - Only available with ImageMagick-7
- 'truecolormatte' - Only available with ImageMagick-6
- 'colorseparation'
- 'colorseparationalpha' - Only available with ImageMagick-7
- 'colorseparationmatte' - Only available with ImageMagick-6
- 'optimize'
- 'palettembilevelalpha' - Only available with ImageMagick-7
- 'palettembilevelmatte' - Only available with ImageMagick-6

See also:

ImageMagick Image Types Describes the `MagickSetImageType` method which can be used to set the type of an image

`wand.image.INTERLACE_TYPES = ('undefined', 'no', 'line', 'plane', 'partition', 'gif', 'jpeg')`
(*tuple*) The list of interlace schemes.

- 'undefined'
- 'no'
- 'line'
- 'plane'
- 'partition'
- 'gif'
- 'jpeg'
- 'png'

New in version 0.5.2.

`wand.image.KERNEL_INFO_TYPES = ('undefined', 'unity', 'gaussian', 'dog', 'log', 'blur', 'comet')`
(*tuple*) The list of builtin kernels.

- 'undefined'
- 'unity'
- 'gaussian'
- 'dog'
- 'log'
- 'blur'
- 'comet'
- 'laplacian'
- 'sobel'
- 'frei_chen'
- 'roberts'

- 'prewitt'
- 'compass'
- 'kirsch'
- 'diamond'
- 'square'
- 'rectangle'
- 'octagon'
- 'disk'
- 'plus'
- 'cross'
- 'ring'
- 'peaks'
- 'edges'
- 'corners'
- 'diagonals'
- 'line_ends'
- 'line_junctions'
- 'ridges'
- 'convex_hull'
- 'thin_se'
- 'skeleton'
- 'chebyshev'
- 'manhattan'
- 'octagonal'
- 'euclidean'
- 'user_defined'
- 'binomial'

wand.image.**MORPHOLOGY_METHODS** = ('undefined', 'convolve', 'correlate', 'erode', 'dilate',
(tuple) The list of morphology methods.

- 'undefined'
- 'convolve'
- 'correlate'
- 'erode'
- 'dilate'
- 'erode_intensity'
- 'dilate_intensity'

- 'distance'
- 'open'
- 'close'
- 'open_intensity'
- 'close_intensity'
- 'smooth'
- 'edgein'
- 'edgeout'
- 'edge'
- 'tophat'
- 'bottom_hat'
- 'hit_and_miss'
- 'thinning'
- 'thicken'
- 'voronoi'
- 'iterative_distance'

wand.image. **ORIENTATION_TYPES** = ('undefined', 'top_left', 'top_right', 'bottom_right', 'bottom_left')
(tuple) The list of *orientation* types.

New in version 0.3.0.

wand.image. **PIXEL_INTERPOLATE_METHODS** = ('undefined', 'average', 'average9', 'average16', 'background', 'bilinear', 'blend', 'catrom', 'integer', 'mesh', 'nearest', 'spline')
(tuple) List of interpolate pixel methods (ImageMagick-7 only.)

- 'undefined'
- 'average'
- 'average9'
- 'average16'
- 'background'
- 'bilinear'
- 'blend'
- 'catrom'
- 'integer'
- 'mesh'
- 'nearest'
- 'spline'

New in version 0.5.0.

wand.image. **STORAGE_TYPES** = ('undefined', 'char', 'double', 'float', 'integer', 'long', 'longlong', 'short', 'unsigned_char', 'unsigned_int', 'unsigned_long', 'unsigned_longlong', 'unsigned_short')
(tuple) The list of pixel storage types.

- 'undefined'

- 'char'
- 'double'
- 'float'
- 'integer'
- 'long'
- 'quantum'
- 'short'
- New in version 0.5.0.

`wand.image.VIRTUAL_PIXEL_METHOD = ('undefined', 'background', 'constant', 'dither', 'edge', 'mirror', 'random', 'tile', 'transparent', 'mask', 'black', 'gray', 'white', 'horizontal_tile', 'vertical_tile', 'horizontal_tile_edge', 'vertical_tile_edge', 'checker_tile')`
(tuple) The list of *virtual_pixel* types. - 'undefined' - 'background' - 'constant' - 'dither' - 'edge' - 'mirror' - 'random' - 'tile' - 'transparent' - 'mask' - 'black' - 'gray' - 'white' - 'horizontal_tile' - 'vertical_tile' - 'horizontal_tile_edge' - 'vertical_tile_edge' - 'checker_tile'

New in version 0.4.1.

`wand.image.UNIT_TYPES = ('undefined', 'pixelsperinch', 'pixelspercentimeter')`
(tuple) The list of resolution unit types.

- 'undefined'
- 'pixelsperinch'
- 'pixelspercentimeter'

See also:

ImageMagick Image Units Describes the `MagickSetImageUnits` method which can be used to set image units of resolution

class `wand.image.BaseImage` (*wand*)

The abstract base of *Image* (container) and *SingleImage*. That means the most of operations, defined in this abstract class, are possible for both *Image* and *SingleImage*.

New in version 0.3.0.

alpha_channel

(bool) Get state of image alpha channel. It can also be used to enable/disable alpha channel, but with different behavior new, copied, or existing.

Behavior of setting *alpha_channel* is defined with the following values:

- **'activate', 'on', or True will enable an images** alpha channel. Existing alpha data is preserved.
- **'deactivate', 'off', or False will disable an images** alpha channel. Any data on the alpha will be preserved.
- **'associate' & 'disassociate' toggle alpha channel flag in** certain image-file specifications.
- **'set'** enables and resets any data in an images alpha channel.
- **'opaque'** enables alpha/matte channel, and forces full opaque image.
- **'transparent'** enables alpha/matte channel, and forces full transparent image.

- **'extract'** copies data in alpha channel across all other channels, and disables alpha channel.
- **'copy'** calculates the gray-scale of RGB channels, and applies it to alpha channel.
- **'shape'** is identical to **'copy'**, but will color the resulting image with the value defined with `background_color`.
- **'remove'** will composite `background_color` value.
- **'background'** replaces full-transparent color with background color.

New in version 0.2.1.

Changed in version 0.4.1: Support for additional setting values. However `Image.alpha_channel` will continue to return `bool` if the current alpha/matte state is enabled.

animation

(`bool`) Whether the image is animation or not. It doesn't only mean that the image has two or more images (frames), but all frames are even the same size. It's about image format, not content. It's `False` even if `image/ico` consists of two or more images of the same size.

For example, it's `False` for `image/jpeg`, `image/gif`, `image/ico`.

If `image/gif` has two or more frames, it's `True`. If `image/gif` has only one frame, it's `False`.

New in version 0.3.0.

Changed in version 0.3.8: Became to accept `image/x-gif` as well.

antialias

(`bool`) If vectors & fonts will use anti-aliasing.

Changed in version 0.5.0: Previously named `font_antialias`.

auto_orient (*args, **kwargs)

Adjusts an image so that its orientation is suitable for viewing (i.e. top-left orientation). If available it uses `MagickAutoOrientImage()` (was added in ImageMagick 6.8.9+) if you have an older magick library, it will use `_auto_orient()` method for fallback.

New in version 0.4.1.

background_color

(`wand.color.Color`) The image background color. It can also be set to change the background color.

New in version 0.1.9.

blue_primary

(`tuple`) The chromatic blue primary point for the image. With ImageMagick-6 the primary value is (x, y) coordinates; however, ImageMagick-7 has (x, y, z).

New in version 0.5.2.

blur (*args, **kwargs)

Blurs the image. We convolve the image with a gaussian operator of the given radius and standard deviation (sigma). For reasonable results, the radius should be larger than sigma. Use a radius of 0 and `blur()` selects a suitable radius for you.

Parameters

- **radius** (`numbers.Real`) – the radius of the, in pixels, not counting the center pixel
- **sigma** (`numbers.Real`) – the standard deviation of the, in pixels

New in version 0.4.5.

border (*color, width, height, compose='copy'*)

Surrounds the image with a border.

Parameters

- **bordercolor** – the border color pixel wand
- **width** (*numbers.Integral*) – the border width
- **height** (*numbers.Integral*) – the border height
- **compose** (*basestring*) – Use composite operator when applying frame. Only used if called with ImageMagick 7+.

New in version 0.3.0.

Changed in version 0.5.0: Added *compose* paramater, and ImageMagick 7 support.

caption (**args, **kwargs*)

Writes a caption *text* into the position.

Parameters

- **text** (*basestring*) – text to write
- **left** (*numbers.Integral*) – x offset in pixels
- **top** (*numbers.Integral*) – y offset in pixels
- **width** (*numbers.Integral*) – width of caption in pixels. default is *width* of the image
- **height** (*numbers.Integral*) – height of caption in pixels. default is *height* of the image
- **font** (*wand.font.Font*) – font to use. default is *font* of the image
- **gravity** (*basestring*) – text placement gravity. uses the current *gravity* setting of the image by default

New in version 0.3.0.

clamp ()

Restrict color values between 0 and quantum range. This is useful when applying arithmetic operations that could result in color values over/underflowing.

New in version 0.5.0.

clone ()

Clones the image. It is equivalent to call *Image* with *image* parameter.

```
with img.clone() as cloned:
    # manipulate the cloned image
    pass
```

Returns the cloned new image

Return type *Image*

New in version 0.1.1.

clut (**args, **kwargs*)

Replace color values by referencing another image as a Color Look Up Table.

Parameters

- **image** (*wand.image.BaseImage*) – Color Look Up Table image.
- **method** (basestring) – Pixel Interpolate method. Only available with ImageMagick-7. See [PIXEL_INTERPOLATE_METHODS](#)

New in version 0.5.0.

coalesce (*args, **kwargs)

Rebuilds image sequence with each frame size the same as first frame, and composites each frame atop of previous.

Note: Only affects GIF, and other formats with multiple pages/layers.

New in version 0.5.0.

colorspace

(basestring) The image colorspace.

Defines image colorspace as in [COLORSPACE_TYPES](#) enumeration.

It may raise `ValueError` when the colorspace is unknown.

New in version 0.3.4.

compare (*args, **kwargs)

Compares an image to a reconstructed image.

Parameters

- **image** (*wand.image.Image*) – The reference image
- **metric** (basestring) – The metric type to use for comparing.

Returns The difference image (*wand.image.Image*), the computed distortion between the images (*numbers.Integral*)

Return type `tuple`

..versionadded:: 0.4.3

compose

(basestring) The type of image compose. It's a string from [COMPOSITE_OPERATORS](#) list. It also can be set.

New in version 0.5.1.

composite (image, left, top)

Places the supplied image over the current image, with the top left corner of image at coordinates left, top of the current image. The dimensions of the current image are not changed.

Parameters

- **image** (*wand.image.Image*) – the image placed over the current image
- **left** (*numbers.Integral*) – the x-coordinate where image will be placed
- **top** (*numbers.Integral*) – the y-coordinate where image will be placed

New in version 0.2.0.

composite_channel (*args, **kwargs)

Composite two images using the particular channel.

Parameters

- **channel** – the channel type. available values can be found in the [CHANNELS](#) mapping
- **image** ([Image](#)) – the composited source image. (the receiver image becomes the destination)
- **operator** – the operator that affects how the composite is applied to the image. available values can be found in the [COMPOSITE_OPERATORS](#) list
- **left** ([numbers.Integral](#)) – the column offset of the composited source image
- **top** ([numbers.Integral](#)) – the row offset of the composited source image

Raises [ValueError](#) – when the given channel or operator is invalid

New in version 0.3.0.

compression

([basestring](#)) The type of image compression. It's a string from [COMPRESSION_TYPES](#) list. It also can be set.

New in version 0.3.6.

Changed in version 0.5.2: Setting [compression](#) now sets both [image_info](#) and [images](#) in the internal image stack.

compression_quality

([numbers.Integral](#)) Compression quality of this image.

New in version 0.2.0.

Changed in version 0.5.2: Setting [compression_quality](#) now sets both [image_info](#) and [images](#) in the internal image stack.

concat (*args, **kwargs)

Concatenates images in stack into a single image. Left-to-right by default, top-to-bottom if [stacked](#) is True.

Parameters [stacked](#) ([bool](#)) – stack images in a column, or in a row (default)

New in version 0.5.0.

contrast_stretch (*args, **kwargs)

Enhance contrast of image by adjusting the span of the available colors.

If only [black_point](#) is given, match the CLI behavior by assuming the [white_point](#) has the same delta percentage off the top e.g. contrast stretch of 15% is calculated as [black_point](#) = 0.15 and [white_point](#) = 0.85.

Parameters

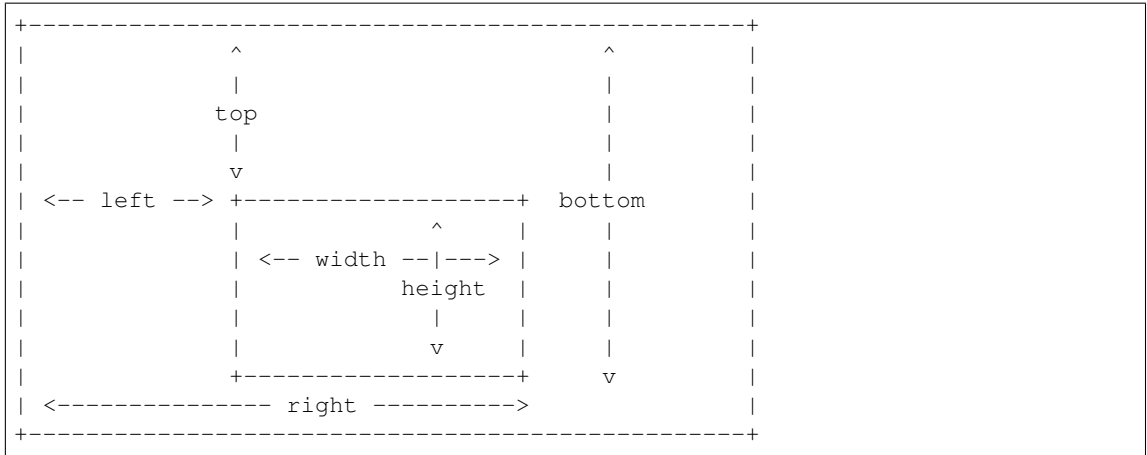
- **black_point** ([numbers.Real](#)) – black point between 0.0 and 1.0. default is 0.0
- **white_point** ([numbers.Real](#)) – white point between 0.0 and 1.0. default value of 1.0 minus [black_point](#)
- **channel** ([CHANNELS](#)) – optional color channel to apply contrast stretch

Raises [ValueError](#) – if channel is not in [CHANNELS](#)

New in version 0.4.1.

crop (*args, **kwargs)

Crops the image in-place.



Parameters

- **left** (`numbers.Integral`) – x-offset of the cropped image. default is 0
- **top** (`numbers.Integral`) – y-offset of the cropped image. default is 0
- **right** (`numbers.Integral`) – second x-offset of the cropped image. default is the `width` of the image. this parameter and `width` parameter are exclusive each other
- **bottom** (`numbers.Integral`) – second y-offset of the cropped image. default is the `height` of the image. this parameter and `height` parameter are exclusive each other
- **width** (`numbers.Integral`) – the `width` of the cropped image. default is the `width` of the image. this parameter and `right` parameter are exclusive each other
- **height** (`numbers.Integral`) – the `height` of the cropped image. default is the `height` of the image. this parameter and `bottom` parameter are exclusive each other
- **reset_coords** (`bool`) – optional flag. If set, after the rotation, the coordinate frame will be relocated to the upper-left corner of the new image. By default is `True`.
- **gravity** (`GRAVITY_TYPES`) – optional flag. If set, will calculate the `top` and `left` attributes. This requires both `width` and `height` parameters to be included.

Raises **ValueError** – when one or more arguments are invalid

Note: If you want to crop the image but not in-place, use slicing operator.

Changed in version 0.4.1: Added `gravity` option. Using `gravity` along with `width` & `height` to auto-adjust `left` & `top` attributes.

Changed in version 0.1.8: Made to raise `ValueError` instead of `IndexError` for invalid `width/height` arguments.

New in version 0.1.7.

deconstruct (`*args, **kwargs`)

Iterates over internal image stack, and adjust each frame size to minimum bounding region of any changes from the previous frame.

New in version 0.5.0.

depth

(`numbers.Integral`) The depth of this image.

New in version 0.2.1.

deskew (*args, **kwargs)

Attempts to remove skew artifacts common with most scanning & optical import devices.

Params threshold limit between foreground & background.

New in version 0.5.0.

despeckle (*args, **kwargs)

Applies filter to reduce noise in image.

New in version 0.5.0.

dirty = None

(bool) Whether the image is changed or not.

dispose

(basestring) Controls how the image data is handled during animations. Values are from [DISPOSE_TYPES](#) list, and can also be set.

See also:

[Dispose Images](#) section in [Animation Basics](#) article.

New in version 0.5.0.

distort (*args, **kwargs)

Distorts an image using various distorting methods.

Parameters

- **method** (basestring) – Distortion method name from [DISTORTION_METHODS](#)
- **arguments** ([collections.abc.Sequence](#)) – List of distorting float arguments unique to distortion method
- **best_fit** (bool) – Attempt to resize resulting image fit distortion. Defaults False

New in version 0.4.1.

edge (*args, **kwargs)

Applies convolution filter to detect edges.

Parameters radius ([numbers.Real](#)) – aperture of detection filter.

New in version 0.5.0.

emboss (*args, **kwargs)

Applies convolution filter against gaussians filter.

Note: The *radius* value should be larger than *sigma* for best results.

Parameters

- **radius** ([numbers.Real](#)) – filter aperture size.
- **sigma** ([numbers.Real](#)) – standard deviation.

New in version 0.5.0.

enhance (*args, **kwargs)

Applies digital filter to reduce noise.

New in version 0.5.0.

equalize (*args, **kwargs)
Equalizes the image histogram

New in version 0.3.10.

evaluate (*args, **kwargs)
Apply arithmetic, relational, or logical expression to an image.

Percent values must be calculated against the quantum range of the image:

```
fifty_percent = img.quantum_range * 0.5
img.evaluate(operator='set', value=fifty_percent)
```

Parameters

- **operator** (*EVALUATE_OPS*) – Type of operation to calculate
- **value** (*numbers.Real*) – Number to calculate with operator
- **channel** (*CHANNELS*) – Optional channel to apply operation on.

Raises

- **TypeError** – When value is not numeric.
- **ValueError** – When operator, or channel are not defined in constants.

New in version 0.4.1.

export_pixels (x=0, y=0, width=None, height=None, channel_map='RGBA', storage='char')
Export pixel data from a raster image to a list of values.

The `channel_map` tells ImageMagick which color channels to export, and what order they should be written as – per pixel. Valid entries for `channel_map` are:

- 'R' - Red channel
- 'G' - Green channel
- 'B' - Blue channel
- 'A' - Alpha channel (0 is transparent)
- 'O' - Alpha channel (0 is opaque)
- 'C' - Cyan channel
- 'Y' - Yellow channel
- 'M' - Magenta channel
- 'K' - Black channel
- 'I' - Intensity channel (only for grayscale)
- 'P' - Padding

See [STORAGE_TYPES](#) for a list of valid `storage` options. This tells ImageMagick what type of data it should calculate & write to. For example; a storage type of 'char' will write a 8-bit value between 0 ~ 255, a storage type of 'short' will write a 16-bit value between 0 ~ 65535, and a 'integer' will write a 32-bit value between 0 ~ 4294967295.

Note: By default, the entire image will be exported as 'char' storage with each pixel mapping Red, Green, Blue, & Alpha channels.

Parameters

- **x** (`numbers.Integral`) – horizontal starting coordinate of raster.
- **y** (`numbers.Integral`) – vertical starting coordinate of raster.
- **width** (`numbers.Integral`) – horizontal length of raster.
- **height** (`numbers.Integral`) – vertical length of raster.
- **channel_map** (`basestring`) – a string listing the channel data format for each pixel.
- **storage** (`basestring`) – what data type each value should be calculated as.

Returns list of values.

Return type `collections.abc.Sequence`

New in version 0.5.0.

extent (`*args, **kwargs`)

extends the image as defined by the geometry, gravity, and wand background color. Set the (x,y) offset of the geometry to move the original wand relative to the extended wand.

Parameters

- **width** (`numbers.Integral`) – the *width* of the extended image. default is the *width* of the image.
- **height** (`numbers.Integral`) – the *height* of the extended image. default is the *height* of the image.
- **x** (`numbers.Integral`) – the x offset of the extended image. default is 0
- **y** (`numbers.Integral`) – the y offset of the extended image. default is 0

New in version 0.4.5.

flip (`*args, **kwargs`)

Creates a vertical mirror image by reflecting the pixels around the central x-axis. It manipulates the image in place.

New in version 0.3.0.

flop (`*args, **kwargs`)

Creates a horizontal mirror image by reflecting the pixels around the central y-axis. It manipulates the image in place.

New in version 0.3.0.

font

(`wand.font.Font`) The current font options.

font_antialias

Deprecated since version 0.5.0: Use *antialias* instead.

font_path

(`basestring`) The path of the current font. It also can be set.

font_size

([numbers.Real](#)) The font size. It also can be set.

format

([basestring](#)) The image format.

If you want to convert the image format, just reset this property:

```
assert isinstance(img, wand.image.Image)
img.format = 'png'
```

It may raise [ValueError](#) when the format is unsupported.

See also:

ImageMagick Image Formats ImageMagick uses an ASCII string known as *magick* (e.g. GIF) to identify file formats, algorithms acting as formats, built-in patterns, and embedded profile types.

New in version 0.1.6.

frame (*args, **kwargs)

Creates a bordered frame around image. Inner & outer bevel can simulate a 3D effect.

Parameters

- **matte** ([wand.color.Color](#)) – color of the frame
- **width** ([numbers.Integral](#)) – total size of frame on x-axis
- **height** ([numbers.Integral](#)) – total size of frame on y-axis
- **inner_bevel** ([numbers.Real](#)) – inset shadow length
- **outer_bevel** ([numbers.Real](#)) – outset highlight length

New in version 0.4.1.

function (*args, **kwargs)

Apply an arithmetic, relational, or logical expression to an image.

Defaults entire image, but can isolate affects to single color channel by passing [CHANNELS](#) value to channel parameter.

Note: Support for function methods added in the following versions of ImageMagick.

- 'polynomial' >= 6.4.8-8
 - 'sinusoid' >= 6.4.8-8
 - 'arcsin' >= 6.5.3-1
 - 'arctan' >= 6.5.3-1
-

Parameters

- **function** ([basestring](#)) – a string listed in [FUNCTION_TYPES](#)
- **arguments** ([collections.abc.Sequence](#)) – a sequence of doubles to apply against function
- **channel** ([basestring](#)) – optional [CHANNELS](#), defaults all

Raises

- **ValueError** – when a function, or channel is not defined in there respected constant
- **TypeError** – if arguments is not a sequence

New in version 0.4.1.

fx (*args, **kwargs)

Manipulate each pixel of an image by given expression.

FX will preserve current wand instance, and return a new instance of *Image* containing affected pixels.

Defaults entire image, but can isolate affects to single color channel by passing *CHANNELS* value to channel parameter.

See also:

The anatomy of FX expressions can be found at <http://www.imagemagick.org/script/fx.php>

Parameters

- **expression** (basestring) – The entire FX expression to apply
- **channel** (*CHANNELS*) – Optional channel to target.

Returns A new instance of an image with expression applied

Return type *Image*

New in version 0.4.1.

gamma (*args, **kwargs)

Gamma correct image.

Specific color channels can be correct individual. Typical values range between 0.8 and 2.3.

Parameters

- **adjustment_value** (*numbers.Real*) – value to adjust gamma level
- **channel** (basestring) – optional channel to apply gamma correction

Raises

- **TypeError** – if gamma_point is not a *numbers.Real*
- **ValueError** – if channel is not in *CHANNELS*

New in version 0.4.1.

gaussian_blur (*args, **kwargs)

Blurs the image. We convolve the image with a gaussian operator of the given radius and standard deviation (sigma). For reasonable results, the radius should be larger than sigma. Use a radius of 0 and *blur()* selects a suitable radius for you.

Parameters

- **radius** (*numbers.Real*) – the radius of the, in pixels, not counting the center pixel
- **sigma** (*numbers.Real*) – the standard deviation of the, in pixels

New in version 0.3.3.

gravity

(basestring) The text placement gravity used when annotating with text. It's a string from *GRAVITY_TYPES* list. It also can be set.

green_primary

([tuple](#)) The chromatic green primary point for the image. With ImageMagick-6 the primary value is (x, y) coordinates; however, ImageMagick-7 has (x, y, z).

New in version 0.5.2.

hald_clut (*args, **kwargs)

Replace color values by referencing a Higher And Lower Dimension (HALD) Color Look Up Table (CLUT). You can generate a HALD image by using ImageMagick's *hald:* protocol.

```
with Image(filename='rose:') as img:
    with Image(filename='hald:3') as hald:
        hald.gamma(1.367)
        img.hald_clut(hald)
```

Parameters **image** ([wand.image.BaseImage](#)) – The HALD color matrix.

New in version 0.5.0.

height

([numbers.Integral](#)) The height of this image.

histogram

([HistogramDict](#)) The mapping that represents the histogram. Keys are [Color](#) objects, and values are the number of pixels.

Tip: True-color photos can have millions of color values. If performance is more valuable than accuracy, remember to [quantize\(\)](#) the image before generating a [HistogramDict](#).

```
with Image(filename='hd_photo.jpg') as img:
    img.quantize(255, 'RGB', 0, False, False)
    hist = img.histogram
```

New in version 0.3.0.

implode (amount=0.0, method='undefined')

Creates a “imploding” effect by pulling pixels towards the center of the image.

Parameters

- **amount** ([numbers.Real](#)) – Normalized degree of effect between 0.0 & 1.0.
- **method** (basestring) – Which interpolate method to apply to effected pixels. See [PIXEL_INTERPOLATE_METHODS](#) for a list of options. Only available with ImageMagick-7.

New in version 0.5.2.

import_pixels (x=0, y=0, width=None, height=None, channel_map='RGB', storage='char', data=None)

Import pixel data from a byte-string to the image. The instance of [Image](#) must already be allocated with the correct size.

The `channel_map` tells ImageMagick which color channels to export, and what order they should be written as – per pixel. Valid entries for `channel_map` are:

- 'R' - Red channel
- 'G' - Green channel
- 'B' - Blue channel

- 'A' - Alpha channel (0 is transparent)
- 'O' - Alpha channel (0 is opaque)
- 'C' - Cyan channel
- 'Y' - Yellow channel
- 'M' - Magenta channel
- 'K' - Black channel
- 'I' - Intensity channel (only for grayscale)
- 'P' - Padding

See [STORAGE_TYPES](#) for a list of valid `storage` options. This tells ImageMagick what type of data it should calculate & write to. For example; a storage type of 'char' will write a 8-bit value between 0 ~ 255, a storage type of 'short' will write a 16-bit value between 0 ~ 65535, and a 'integer' will write a 32-bit value between 0 ~ 4294967295.

Note: By default, the entire image will be exported as 'char' storage with each pixel mapping Red, Green, Blue, & Alpha channels.

Parameters

- **x** (`numbers.Integral`) – horizontal starting coordinate of raster.
- **y** (`numbers.Integral`) – vertical starting coordinate of raster.
- **width** (`numbers.Integral`) – horizontal length of raster.
- **height** (`numbers.Integral`) – vertical length of raster.
- **channel_map** (`basestring`) – a string listing the channel data format for each pixel.
- **storage** (`basestring`) – what data type each value should be calculated as.

New in version 0.5.0.

interlace_scheme

(`basestring`) The interlace used by the image. See [INTERLACE_TYPES](#).

New in version 0.5.2.

interpolate_method

(`basestring`) The interpolation method of the image. See [PIXEL_INTERPOLATE_METHODS](#).

New in version 0.5.2.

level (*black=0.0, white=None, gamma=1.0, channel=None*)

Adjusts the levels of an image by scaling the colors falling between specified black and white points to the full available quantum range.

If only `black` is given, `white` will be adjusted inward.

Parameters

- **black** (`numbers.Real`) – Black point, as a percentage of the system's quantum range. Defaults to 0.
- **white** (`numbers.Real`) – White point, as a percentage of the system's quantum range. Defaults to 1.0.

- **gamma** (`numbers.Real`) – Optional gamma adjustment. Values > 1.0 lighten the image’s midtones while values < 1.0 darken them.
- **channel** (`CHANNELS`) – The channel type. Available values can be found in the [CHANNELS](#) mapping. If `None`, normalize all channels.

Note: Images may not be affected if the `white` value is equal, or less than, the `black` value.

New in version 0.4.1.

linear_stretch (`*args, **kwargs`)

Enhance saturation intensity of an image.

Parameters

- **black_point** (`numbers.Real`) – Black point between 0.0 and 1.0. Default 0.0
- **white_point** (`numbers.Real`) – White point between 0.0 and 1.0. Default 1.0

New in version 0.4.1.

liquid_rescale (`*args, **kwargs`)

Rescales the image with [seam carving](#), also known as image retargeting, content-aware resizing, or liquid rescaling.

Parameters

- **width** (`numbers.Integral`) – the width in the scaled image
- **height** (`numbers.Integral`) – the height in the scaled image
- **delta_x** (`numbers.Real`) – maximum seam transversal step. 0 means straight seams. default is 0
- **rigidity** (`numbers.Real`) – introduce a bias for non-straight seams. default is 0

Raises [wand.exceptions.MissingDelegateError](#) – when ImageMagick isn’t configured `--with-lqr` option.

Note: This feature requires ImageMagick to be configured `--with-lqr` option. Or it will raise [MissingDelegateError](#):

See also:

[Seam carving — Wikipedia](#) The article which explains what seam carving is on Wikipedia.

loop

(`numbers.Integral`) Number of frame iterations. A value of 0 will loop forever.

matte_color

(`wand.color.Color`) The color value of the matte channel. This can also be set.

..versionadded:: 0.4.1

merge_layers (`*args, **kwargs`)

Composes all the image layers from the current given image onward to produce a single image of the merged layers.

The initial canvas’s size depends on the given `ImageLayerMethod`, and is initialized using the first images background color. The images are then composited onto that image in sequence using the given

composition that has been assigned to each individual image. The method must be set with a value from `IMAGE_LAYER_METHOD` that is acceptable to this operation. (See ImageMagick documentation for more details.)

Parameters `method` (basestring) – the method of selecting the size of the initial canvas.

New in version 0.4.3.

modulate (*args, **kwargs)

Changes the brightness, saturation and hue of an image. We modulate the image with the given brightness, saturation and hue.

Parameters

- **brightness** (`numbers.Real`) – percentage of brightness
- **saturation** (`numbers.Real`) – percentage of saturation
- **hue** (`numbers.Real`) – percentage of hue rotation

Raises `ValueError` – when one or more arguments are invalid

New in version 0.3.4.

morphology (*args, **kwargs)

Manipulate pixels based on the shape of neighboring pixels.

The method determines what type of effect to apply to matching kernel shapes. Common methods can be add/remove, or lighten/darken pixel values.

The kernel describes the shape of the matching neighbors. Common shapes are provided as “built-in” kernels. See `:const‘KERNEL_INFO_TYPES‘` for examples. The format for built-in kernels is:

```
label:geometry
```

Where *label* is the kernel name defined in `KERNEL_INFO_TYPES`, and *geometry* is an optional geometry size. For example:

```
with Image(filename='rose:') as img:
    img.morphology(method='dilate', kernel='octagon:3x3')
    # or simply
    img.morphology(method='edgein', kernel='octagon')
```

Custom kernels can be applied by following a similar format:

```
geometry:args
```

Where *geometry* is the size of the custom kernel, and *args* list a comma separated list of values. For example:

```
custom_kernel='5x3:nan,1,1,1,nan 1,1,1,1,1 nan,1,1,1,nan'
with Image(filename='rose:') as img:
    img.morphology(method='dilate', kernel=custom_kernel)
```

Parameters

- **method** (basestring) – effect function to apply. See `MORPHOLOGY_METHODS` for a list of methods.
- **kernel** (basestring) – shape to evaluate surrounding pixels. See `KERNEL_INFO_TYPES` for a list of built-in shapes.

- **iterations** (`numbers.Integral`) – Number of times a morphology method should be applied to the image. Default 1. Use -1 for unlimited iterations until the image is unchanged by the method operator.

New in version 0.5.0.

negate (`grayscale=False, channel=None`)

Negate the colors in the reference image.

Parameters

- **grayscale** (`bool`) – if set, only negate grayscale pixels in the image.
- **channel** (`basestring`) – the channel type. available values can be found in the [CHANNELS](#) mapping. If None, negate all channels.

New in version 0.3.8.

normalize (`*args, **kwargs`)

Normalize color channels.

Parameters **channel** (`basestring`) – the channel type. available values can be found in the [CHANNELS](#) mapping. If None, normalize all channels.

optimize_layers (`*args, **kwargs`)

Attempts to crop each frame to the smallest image without altering the animation.

Note: This will only affect GIF image formats.

New in version 0.5.0.

optimize_transparency (`*args, **kwargs`)

Iterates over frames, and sets transparent values for each pixel unchanged by previous frame.

Note: This will only affect GIF image formats.

New in version 0.5.0.

options = None

(`OptionDict`) The mapping of internal option settings.

New in version 0.3.0.

Changed in version 0.3.4: Added 'jpeg:sampling-factor' option.

Changed in version 0.3.9: Added 'pdf:use-cropbox' option.

orientation

(`basestring`) The image orientation. It's a string from [ORIENTATION_TYPES](#) list. It also can be set.

New in version 0.3.0.

page

The dimensions and offset of this Wand's page as a 4-tuple: (`width, height, x, y`).

Note that since it is based on the virtual canvas, it may not equal the dimensions of an image. See the ImageMagick documentation on the virtual canvas for more information.

New in version 0.4.3.

page_height

(`numbers.Integral`) The height of the page for this wand.

New in version 0.4.3.

page_width

(`numbers.Integral`) The width of the page for this wand.

New in version 0.4.3.

page_x

(`numbers.Integral`) The X-offset of the page for this wand.

New in version 0.4.3.

page_y

(`numbers.Integral`) The Y-offset of the page for this wand.

New in version 0.4.3.

posterize (*args, **kwargs)

Reduce color levels per channel.

Parameters

- **levels** (`numbers.Integral`) – Number of levels per channel.
- **dither** (*basestring*) – Dither method to apply. See [DITHER_METHODS](#).

New in version 0.5.0.

quantize (*args, **kwargs)

quantize analyzes the colors within a sequence of images and chooses a fixed number of colors to represent the image. The goal of the algorithm is to minimize the color difference between the input and output image while minimizing the processing time.

Parameters

- **number_colors** (`numbers.Integral`) – the number of colors.
- **colorspace_type** (*basestring*) – `colorspace_type`. available value can be found in the [COLORSPACE_TYPES](#)
- **treedepth** (`numbers.Integral`) – normally, this integer value is zero or one. a zero or one tells *quantize()* to choose a optimal tree depth of $\log_4(\text{number_colors})$. a tree of this depth generally allows the best representation of the reference image with the least amount of memory and the fastest computational speed. in some cases, such as an image with low color dispersion (a few number of colors), a value other than $\log_4(\text{number_colors})$ is required. to expand the color tree completely, use a value of 8
- **dither** (`bool`) – a value other than zero distributes the difference between an original image and the corresponding color reduced algorithm to neighboring pixels along a Hilbert curve
- **measure_error** (`bool`) – a value other than zero measures the difference between the original and quantized images. this difference is the total quantization error. The error is computed by summing over all pixels in an image the distance squared in RGB space between each reference pixel value and its quantized value

New in version 0.4.2.

quantum_range

(`int`) The maxumim value of a color channel that is supported by the imagemagick library.

New in version 0.2.0.

red_primary

([tuple](#)) The chromatic red primary point for the image. With ImageMagick-6 the primary value is (x, y) coordinates; however, ImageMagick-7 has (x, y, z).

New in version 0.5.2.

resample (*args, **kwargs)

Adjust the number of pixels in an image so that when displayed at the given Resolution or Density the image will still look the same size in real world terms.

Parameters

- **x_res** ([numbers.Real](#)) – the X resolution (density) in the scaled image. default is the original resolution.
- **y_res** ([numbers.Real](#)) – the Y resolution (density) in the scaled image. default is the original resolution.
- **filter** (basestring, [numbers.Integral](#)) – a filter type to use for resizing. choose one in [FILTER_TYPES](#). default is 'undefined' which means IM will try to guess best one to use.
- **blur** ([numbers.Real](#)) – the blur factor where > 1 is blurry, < 1 is sharp. default is 1

New in version 0.4.5.

reset_coords ()

Reset the coordinate frame of the image so to the upper-left corner is (0, 0) again (crop and rotate operations change it).

New in version 0.2.0.

resize (*args, **kwargs)

Resizes the image.

Parameters

- **width** ([numbers.Integral](#)) – the width in the scaled image. default is the original width
- **height** ([numbers.Integral](#)) – the height in the scaled image. default is the original height
- **filter** (basestring, [numbers.Integral](#)) – a filter type to use for resizing. choose one in [FILTER_TYPES](#). default is 'undefined' which means IM will try to guess best one to use
- **blur** ([numbers.Real](#)) – the blur factor where > 1 is blurry, < 1 is sharp. default is 1

Changed in version 0.2.1: The default value of filter has changed from 'triangle' to 'undefined' instead.

Changed in version 0.1.8: The blur parameter changed to take [numbers.Real](#) instead of [numbers.Rational](#).

New in version 0.1.1.

resolution

([tuple](#)) Resolution of this image.

New in version 0.3.0.

rotate (*args, **kwargs)

Rotates the image right. It takes a background color for degree that isn't a multiple of 90.

Parameters

- **degree** (`numbers.Real`) – a degree to rotate. multiples of 360 affect nothing
- **background** (`wand.color.Color`) – an optional background color. default is transparent
- **reset_coords** (`bool`) – optional flag. If set, after the rotation, the coordinate frame will be relocated to the upper-left corner of the new image. By default is *True*.

New in version 0.2.0: The `reset_coords` parameter.

New in version 0.1.8.

sample (*args, **kwargs)

Resizes the image by sampling the pixels. It's basically quicker than `resize()` except less quality as a tradeoff.

Parameters

- **width** (`numbers.Integral`) – the width in the scaled image. default is the original width
- **height** (`numbers.Integral`) – the height in the scaled image. default is the original height

New in version 0.3.4.

sequence = None

(`collections.abc.Sequence`) The list of *SingleImages* that the image contains.

New in version 0.3.0.

shade (*args, **kwargs)

Creates a 3D effect by simulating a light from an elevated angle.

Parameters

- **gray** (`bool`) – Isolate the effect on pixel intensity. Default is False.
- **azimuth** (`numbers.Real`) – Angle from x-axis.
- **elevation** (`numbers.Real`) – Amount of pixels from the z-axis.

New in version 0.5.0.

shadow (*args, **kwargs)

Generates an image shadow.

Parameters

- **alpha** (`numbers.Real`) – Ratio of transparency.
- **sigma** (`numbers.Real`) – Standard deviation of the gaussian filter.
- **x** (`numbers.Integral`) – x-offset.
- **y** (`numbers.Integral`) – y-offset.

New in version 0.5.0.

sharpen (*args, **kwargs)

Applies a gaussian effect to enhance the sharpness of an image.

Note: For best results, ensure `radius` is larger than `sigma`.

Defaults values of zero will have ImageMagick attempt to auto-select suitable values.

Parameters

- **radius** (`numbers.Real`) – size of gaussian aperture.
- **sigma** (`numbers.Real`) – Standard deviation of the gaussian filter.

New in version 0.5.0.

shave (**args, **kwargs*)

Remove pixels from the edges.

Parameters

- **columns** (`numbers.Integer`) – amount to shave off the x-axis.
- **rows** (`numbers.Integer`) – amount to shave off the y-axis.

New in version 0.5.0.

signature

(`str`) The SHA-256 message digest for the image pixel stream.

New in version 0.1.9.

size

(`tuple`) The pair of (*width*, *height*).

strip ()

Strips an image of all profiles and comments.

New in version 0.2.0.

threshold (**args, **kwargs*)

Changes the value of individual pixels based on the intensity of each pixel compared to threshold. The result is a high-contrast, two color image. It manipulates the image in place.

Parameters

- **threshold** (`numbers.Real`) – threshold as a factor of quantum
- **channel** (`basestring`) – the channel type. available values can be found in the [CHANNELS](#) mapping. If `None`, threshold all channels.

New in version 0.3.10.

transform (**args, **kwargs*)

Transforms the image using `MagickTransformImage()`, which is a convenience function accepting geometry strings to perform cropping and resizing. Cropping is performed first, followed by resizing. Either or both arguments may be omitted or given an empty string, in which case the corresponding action will not be performed. Geometry specification strings are defined as follows:

A geometry string consists of a size followed by an optional offset. The size is specified by one of the options below, where **bold** terms are replaced with appropriate integer values:

scale% Height and width both scaled by specified percentage

scale-x%*x*scale-y% Height and width individually scaled by specified percentages. Only one % symbol is needed.

width Width given, height automatically selected to preserve aspect ratio.

xheight Height given, width automatically selected to preserve aspect ratio.

widthxheight Maximum values of width and height given; aspect ratio preserved.

widthxheight! Width and height emphatically given; original aspect ratio ignored.

widthxheight> Shrinks images with dimension(s) larger than the corresponding width and/or height dimension(s).

widthxheight< Enlarges images with dimensions smaller than the corresponding width and/or height dimension(s).

area@ Resize image to have the specified area in pixels. Aspect ratio is preserved.

The offset, which only applies to the cropping geometry string, is given by `{+-}x{+-}y`, that is, one plus or minus sign followed by an `x` offset, followed by another plus or minus sign, followed by a `y` offset. Offsets are in pixels from the upper left corner of the image. Negative offsets will cause the corresponding number of pixels to be removed from the right or bottom edge of the image, meaning the cropped size will be the computed size minus the absolute value of the offset.

For example, if you want to crop your image to 300x300 pixels and then scale it by 2x for a final size of 600x600 pixels, you can call:

```
image.transform('300x300', '200%')
```

This method is a fairly thin wrapper for the C API, and does not perform any additional checking of the parameters except insofar as verifying that they are of the correct type. Thus, like the C API function, the method is very permissive in terms of what it accepts for geometry strings; unrecognized strings and trailing characters will be ignored rather than raising an error.

Parameters

- **crop** (basestring) – A geometry string defining a subregion of the image to crop to
- **resize** (basestring) – A geometry string defining the final size of the image

See also:

ImageMagick Geometry Specifications Cropping and resizing geometry for the `transform` method are specified according to ImageMagick's geometry string format. The ImageMagick documentation provides more information about geometry strings.

New in version 0.2.2.

Changed in version 0.5.0: Will call `crop()` followed by `resize()` in the event that `MagickTransformImage()` is not available.

transform_colorspace (*args, **kwargs)

Transform image's colorspace.

Parameters **colorspace_type** (basestring) – colorspace_type. available value can be found in the [COLORSPACE_TYPES](#)

New in version 0.4.2.

transparent_color (*args, **kwargs)

Makes the color `color` a transparent color with a tolerance of fuzz. The `alpha` parameter specifies the transparency level and the parameter `fuzz` specifies the tolerance.

Parameters

- **color** (`wand.color.Color`) – The color that should be made transparent on the image, color object
- **alpha** (`numbers.Real`) – the level of transparency: 1.0 is fully opaque and 0.0 is fully transparent.
- **fuzz** (`numbers.Integral`) – By default target must match a particular pixel color exactly. However, in many cases two colors may differ by a small amount. The fuzz member of image defines how much tolerance is acceptable to consider two colors as the same. For example, set fuzz to 10 and the color red at intensities of 100 and 102 respectively are now interpreted as the same color for the color.
- **invert** (`bool`) – Boolean to tell to paint the inverse selection.

New in version 0.3.0.

transparentize (`*args, **kwargs`)

Makes the image transparent by subtracting some percentage of the black color channel. The transparency parameter specifies the percentage.

Parameters **transparency** (`numbers.Real`) – the percentage fade that should be performed on the image, from 0.0 to 1.0

New in version 0.2.0.

transpose (`*args, **kwargs`)

Creates a vertical mirror image by reflecting the pixels around the central x-axis while rotating them 90-degrees.

New in version 0.4.1.

transverse (`*args, **kwargs`)

Creates a horizontal mirror image by reflecting the pixels around the central y-axis while rotating them 270-degrees.

New in version 0.4.1.

trim (`*args, **kwargs`)

Remove solid border from image. Uses top left pixel as a guide by default, or you can also specify the color to remove.

Parameters

- **color** (`Color`) – the border color to remove. if it's omitted top left pixel is used by default
- **fuzz** (`numbers.Integral`) – Defines how much tolerance is acceptable to consider two colors as the same.

Changed in version 0.5.2: The `color` parameter may except color-compliant strings.

Changed in version 0.3.0: Optional `color` and `fuzz` parameters.

New in version 0.2.1.

type

(`basestring`) The image type.

Defines image type as in `IMAGE_TYPES` enumeration.

It may raise `ValueError` when the type is unknown.

New in version 0.2.2.

unique_colors (*args, **kwargs)

Discards all duplicate pixels, and rebuilds the image as a single row.

New in version 0.5.0.

units

(basestring) The resolution units of this image.

unsharp_mask (*args, **kwargs)

Sharpens the image using unsharp mask filter. We convolve the image with a Gaussian operator of the given radius and standard deviation (sigma). For reasonable results, radius should be larger than sigma. Use a radius of 0 and `unsharp_mask()` selects a suitable radius for you.

Parameters

- **radius** (`numbers.Real`) – the radius of the Gaussian, in pixels, not counting the center pixel
- **sigma** (`numbers.Real`) – the standard deviation of the Gaussian, in pixels
- **amount** (`numbers.Real`) – the percentage of the difference between the original and the blur image that is added back into the original
- **threshold** (`numbers.Real`) – the threshold in pixels needed to apply the difference amount

New in version 0.3.4.

vignette (*args, **kwargs)

Creates a soft vignette style effect on the image.

Parameters

- **radius** (`numbers.Real`) – the radius of the Gaussian blur effect.
- **sigma** (`numbers.Real`) – the standard deviation of the Gaussian effect.
- **x** (`numbers.Integral`) – Width of the oval effect.
- **y** (`numbers.Integral`) – Height of the oval effect.

New in version 0.5.2.

virtual_pixel

(basestring) The virtual pixel of image. This can also be set with a value from `VIRTUAL_PIXEL_METHOD` ... versionadded:: 0.4.1

wand

Internal pointer to the MagickWand instance. It may raise `ClosedImageError` when the instance has destroyed already.

watermark (*args, **kwargs)

Transparentized the supplied image and places it over the current image, with the top left corner of image at coordinates left, top of the current image. The dimensions of the current image are not changed.

Parameters

- **image** (`wand.image.Image`) – the image placed over the current image
- **transparency** (`numbers.Real`) – the percentage fade that should be performed on the image, from 0.0 to 1.0
- **left** (`numbers.Integral`) – the x-coordinate where *image* will be placed
- **top** (`numbers.Integral`) – the y-coordinate where *image* will be placed

New in version 0.2.0.

wave (*args, **kwargs)

Creates a ripple effect within the image.

Parameters

- **amplitude** (`numbers.Real`) – height of wave form.
- **wave_length** (`numbers.Real`) – width of wave form.
- **method** (`basestring`) – pixel interpolation method. Only available with ImageMagick-7. See [PIXEL_INTERPOLATE_METHODS](#)

New in version 0.5.2.

white_point

(`tuple`) The chromatic white point for the image. With ImageMagick-6 the primary value is (x, y) coordinates; however, ImageMagick-7 has (x, y, z).

New in version 0.5.2.

white_threshold (*args, **kwargs)

Forces all pixels above a given color as white. Leaves pixels below threshold unaltered.

Parameters **threshold** (`Color`) – Color to be referenced as a threshold.

New in version 0.5.2.

width

(`numbers.Integral`) The width of this image.

class `wand.image.ChannelDepthDict` (*image*)

The mapping table of channels to their depth.

Parameters **image** (*Image*) – an image instance

Note: You don't have to use this by yourself. Use `Image.channel_depths` property instead.

New in version 0.3.0.

class `wand.image.ChannelImageDict` (*image*)

The mapping table of separated images of the particular channel from the image.

Parameters **image** (*Image*) – an image instance

Note: You don't have to use this by yourself. Use `Image.channel_images` property instead.

New in version 0.3.0.

exception `wand.image.ClosedImageError`

An error that rises when some code tries access to an already closed image.

class `wand.image.HistogramDict` (*image*)

Specialized mapping object to represent color histogram. Keys are colors, and values are the number of pixels.

Parameters **image** (*BaseImage*) – the image to get its histogram

New in version 0.3.0.

```
class wand.image.Image (image=None, blob=None, file=None, filename=None, format=None,
                        width=None, height=None, depth=None, background=None, resolution=None, pseudo=None)
```

An image object.

Parameters

- **image** (*Image*) – makes an exact copy of the image
- **blob** (*bytes*) – opens an image of the blob byte array
- **file** (*file object*) – opens an image of the file object
- **filename** (*basestring*) – opens an image of the filename string
- **format** (*basestring*) – forces filename to buffer. format to help ImageMagick detect the file format. Used only in blob or file cases
- **width** (*numbers.Integral*) – the width of new blank image or an image loaded from raw data.
- **height** (*numbers.Integral*) – the height of new blank image or an image loaded from raw data.
- **depth** (*numbers.Integral*) – the depth used when loading raw data.
- **background** (*wand.color.Color*) – an optional background color. default is transparent
- **resolution** (*collections.abc.Sequence, numbers.Integral*) – set a resolution value (dpi), useful for vectorial formats (like pdf)

New in version 0.1.5: The file parameter.

New in version 0.1.1: The blob parameter.

New in version 0.2.1: The format parameter.

New in version 0.2.2: The width, height, background parameters.

New in version 0.3.0: The resolution parameter.

New in version 0.4.2: The depth parameter.

Changed in version 0.4.2: The depth, width and height parameters can be used with the filename, file and blob parameters to load raw pixel data.

New in version 0.5.0: The pseudo parameter.

[left:right, top:bottom]

Crops the image by its left, right, top and bottom, and then returns the cropped one.

```
with img[100:200, 150:300] as cropped:
    # manipulated the cropped image
    pass
```

Like other subscriptable objects, default is 0 or its width/height:

```
img[:, :]          #--> just clone
img[100, 200:]     #--> equivalent to img[0:100, 200:img.height]
```

Negative integers count from the end (width/height):

```
img[-70:-50, -20:-10]
#--> equivalent to img[width-70:width-50, height-20:height-10]
```

Returns the cropped image

Rtype *Image*

New in version 0.1.2.

artifacts = None

(*ArtifactTree*) A dict mapping to image artifacts. Similar to *metadata*, but used to alter behavior of various internal operations.

New in version 0.5.0.

blank (*width, height, background=None*)

Creates blank image.

Parameters

- **width** (*numbers.Integral*) – the width of new blank image.
- **height** (*numbers.Integral*) – the height of new blank image.
- **background** (*wand.color.Color*) – an optional background color. default is transparent

Returns blank image

Return type *Image*

New in version 0.3.0.

channel_depths = None

(*ChannelDepthDict*) The mapping of channels to their depth. Read only.

New in version 0.3.0.

channel_images = None

(*ChannelImageDict*) The mapping of separated channels from the image.

```
with image.channel_images['red'] as red_image:
    display(red_image)
```

clear()

Clears resources associated with the image, leaving the image blank, and ready to be used with new image.

New in version 0.3.0.

close()

Closes the image explicitly. If you use the image object in *with* statement, it was called implicitly so don't have to call it.

Note: It has the same functionality of *destroy()* method.

compare_layers (*method*)

Generates new images showing the delta pixels between layers. Similar pixels are converted to transparent. Useful for debugging complex animations.

```
with img.compare_layers('compareany') as delta:
    delta.save(filename='framediff_%02d.png')
```

Note: May not work as expected if animations are already optimized.

Parameters `method` (basestring) – Can be 'compareany', 'compareclear', or 'compareoverlay'

Returns new image stack.

Return type *Image*

New in version 0.5.0.

convert (*format*)

Converts the image format with the original image maintained. It returns a converted image instance which is new.

```
with img.convert('png') as converted:
    converted.save(filename='converted.png')
```

Parameters `format` (basestring) – image format to convert to

Returns a converted image

Return type *Image*

Raises **ValueError** – when the given `format` is unsupported

New in version 0.1.6.

destroy ()

Manually remove *SingleImage*'s in the *Sequence*, allowing it to be properly garbage collected after using a `with Image()` context manager.

make_blob (*format=None*)

Makes the binary string of the image.

Parameters `format` (basestring) – the image format to write e.g. 'png', 'jpeg'. it is omissible

Returns a blob (bytes) string

Return type *bytes*

Raises **ValueError** – when `format` is invalid

Changed in version 0.1.6: Removed a side effect that changes the image `format` silently.

New in version 0.1.5: The `format` parameter became optional.

New in version 0.1.1.

metadata = None

(*Metadata*) The metadata mapping of the image. Read only.

New in version 0.3.0.

mimetype

(basestring) The MIME type of the image e.g. 'image/jpeg', 'image/png'.

New in version 0.1.7.

profiles = None

(*ProfileDict*) The mapping of image profiles.

New in version 0.5.1.

pseudo (*width, height, pseudo='xc:'*)

Creates a new image from ImageMagick's internal protocol coders.

Parameters

- **width** (`numbers.Integral`) – Total columns of the new image.
- **height** (`numbers.Integral`) – Total rows of the new image.
- **pseudo** (`basestring`) – The protocol & arguments for the pseudo image.

New in version 0.5.0.

read (`file=None, filename=None, blob=None, resolution=None`)

Read new image into `Image()` object.

Parameters

- **blob** (`bytes`) – reads an image from the `blob` byte array
- **file** (`file object`) – reads an image from the `file` object
- **filename** (`basestring`) – reads an image from the `filename` string
- **resolution** (`collections.abc.Sequence, numbers.Integral`) – set a resolution value (DPI), useful for vectorial formats (like PDF)

New in version 0.3.0.

save (`file=None, filename=None`)

Saves the image into the `file` or `filename`. It takes only one argument at a time.

Parameters

- **file** (`file object`) – a `file` object to write to
- **filename** (`basestring`) – a `filename` string to write to

New in version 0.1.5: The `file` parameter.

New in version 0.1.1.

class `wand.image.ImageProperty` (`image`)

The mixin class to maintain a weak reference to the parent `Image` object.

New in version 0.3.0.

image

(`Image`) The parent image.

It ensures that the parent `Image`, which is held in a weak reference, still exists. Returns the dereferenced `Image` if it does exist, or raises a `ClosedImageError` otherwise.

Exc `ClosedImageError` when the parent `Image` has been destroyed

class `wand.image.Iterator` (`image=None, iterator=None`)

Row iterator for `Image`. It shouldn't be instantiated directly; instead, it can be acquired through `Image` instance:

```
assert isinstance(image, wand.image.Image)
iterator = iter(image)
```

It doesn't iterate every pixel, but rows. For example:

```
for row in image:
    for col in row:
        assert isinstance(col, wand.color.Color)
        print(col)
```

Every row is a `collections.abc.Sequence` which consists of one or more `wand.color.Color` values.

Parameters `image` (*Image*) – the image to get an iterator

New in version 0.1.3.

clone()

Clones the same iterator.

class `wand.image.Metadata` (*image*)

Class that implements dict-like read-only access to image metadata like EXIF or IPTC headers. Most WRITE encoders will ignore properties assigned here.

Parameters `image` (*Image*) – an image instance

Note: You don't have to use this by yourself. Use `Image.metadata` property instead.

New in version 0.3.0.

class `wand.image.OptionDict` (*image*)

Free-form mutable mapping of global internal settings.

New in version 0.3.0.

Changed in version 0.5.0: Remove key check to `OPTIONS`. Image properties are specific to vendor, and this library should not attempt to manage the 100+ options in a whitelist.

`wand.image.manipulative` (*function*)

Mark the operation manipulating itself instead of returning new one.

class `wand.image.ArtifactTree` (*image*)

Splay tree to map image artifacts. Values defined here are intended to be used elsewhere, and will not be written to the encoded image.

For example:

```
# Omit timestamp from PNG file headers.
with Image(filename='input.png') as img:
    img.artifacts['png:exclude-chunks'] = 'tIME'
    img.save(filename='output.png')
```

Parameters `image` (*Image*) – an image instance

Note: You don't have to use this by yourself. Use `Image.artifacts` property instead.

New in version 0.5.0.

class `wand.image.ProfileDict` (*image*)

The mapping table of embedded image profiles.

Use this to get, set, and delete whole profile payloads on an image. Each payload is a raw binary string.

For example:

```
with Image(filename='photo.jpg') as img:
    # Extract EXIF
    with open('exif.bin', 'wb') as payload:
```

(continues on next page)

(continued from previous page)

```

        payload.write(img.profiles['exif'])
    # Import ICC
    with open('color_profile.icc', 'rb') as payload:
        img.profiles['icc'] = payload.read()
    # Remove XMP
    del img.profiles['xmp']

```

See also:[Embedded Image Profiles](#) for a list of supported profiles.

New in version 0.5.1.

4.1.2 wand.color — Colors

New in version 0.1.2.

class wand.color.Color (string=None, raw=None)

Color value.

Unlike any other objects in Wand, its resource management can be implicit when it used outside of `with` block. In these case, its resource are allocated for every operation which requires a resource and destroyed immediately. Of course it is inefficient when the operations are much, so to avoid it, you should use color objects inside of `with` block explicitly e.g.:

```

red_count = 0
with Color('#f00') as red:
    with Image(filename='image.png') as img:
        for row in img:
            for col in row:
                if col == red:
                    red_count += 1

```

Parameters **string** (basestring) – a color name string e.g. 'rgb(255, 255, 255)', '#fff', 'white'. see [ImageMagick Color Names](#) doc also

Changed in version 0.3.0: `Color` objects become hashable.

Changed in version 0.5.1: Color channel properties can now be set.

Changed in version 0.5.1: Added `cyan`, `magenta`, `yellow`, & `black` properties for CMYK `Color` instances.Changed in version 0.5.1: Method `Color.from_hsl()` can create a RGB color from hue, saturation, & lightness values.**See also:**

ImageMagick Color Names The color can then be given as a color name (there is a limited but large set of these; see below) or it can be given as a set of numbers (in decimal or hexadecimal), each corresponding to a channel in an RGB or RGBA color model. HSL, HSLA, HSB, HSBA, CMYK, or CMYKA color models may also be specified. These topics are briefly described in the sections below.

== (other)

Equality operator.

Param other a color another one

Type `color` *Color*

Returns True only if two images equal.

Rtype `bool`

alpha

(`numbers.Real`) Alpha value, from 0.0 to 1.0.

alpha_int8

(`numbers.Integral`) Alpha value as 8bit integer which is a common style. From 0 to 255.

New in version 0.3.0.

alpha_quantum

(`numbers.Integral`) Alpha value. Scale depends on `QUANTUM_DEPTH`.

New in version 0.3.0.

black

(`numbers.Real`) Black, or 'K', color channel in CMYK colorspace. Unused by RGB colorspace.

New in version 0.5.1.

black_int8

(`numbers.Integral`) Black value as 8bit integer which is a common style. From 0 to 255.

New in version 0.5.1.

black_quantum

(`numbers.Integral`) Black. Scale depends on `QUANTUM_DEPTH`.

New in version 0.5.1.

blue

(`numbers.Real`) Blue, from 0.0 to 1.0.

blue_int8

(`numbers.Integral`) Blue as 8bit integer which is a common style. From 0 to 255.

New in version 0.3.0.

blue_quantum

(`numbers.Integral`) Blue. Scale depends on `QUANTUM_DEPTH`.

New in version 0.3.0.

static c_equals (*a*, *b*)

Raw level version of equality test function for two pixels.

Parameters

- **a** (`ctypes.c_void_p`) – a pointer to PixelWand to compare
- **b** (`ctypes.c_void_p`) – a pointer to PixelWand to compare

Returns True only if two pixels equal

Return type `bool`

Note: It's only for internal use. Don't use it directly. Use `==` operator of *Color* instead.

cyan

(`numbers.Real`) Cyan color channel in CMYK colorspace. Unused by RGB colorspace.

New in version 0.5.1.

cyan_int8

(`numbers.Integral`) Cyan value as 8bit integer which is a common style. From 0 to 255.

New in version 0.5.1.

cyan_quantum

(`numbers.Integral`) Cyan. Scale depends on `QUANTUM_DEPTH`.

New in version 0.5.1.

dirty = None

(`bool`) Whether the color has changed or not.

classmethod from_hsl (*hue=0.0, saturation=0.0, lightness=0.0*)

Creates a RGB color from HSL values. The hue, saturation, and lightness must be normalized between 0.0 & 1.0.

```
h=0.75 # 270 Degrees
s=1.0  # 100 Percent
l=0.5  # 50 Percent
with Color.from_hsl(hue=h, saturation=s, lightness=l) as color:
    print(color)  #=> srgb(128,0,255)
```

Parameters

- **hue** (`numbers.Real`) – a normalized double between 0.0 & 1.0.
- **saturation** (`numbers.Real`) – a normalized double between 0.0 & 1.0.
- **lightness** (`numbers.Real`) – a normalized double between 0.0 & 1.0.

Return type `Color`

New in version 0.5.1.

green

(`numbers.Real`) Green, from 0.0 to 1.0.

green_int8

(`numbers.Integral`) Green as 8bit integer which is a common style. From 0 to 255.

New in version 0.3.0.

green_quantum

(`numbers.Integral`) Green. Scale depends on `QUANTUM_DEPTH`.

New in version 0.3.0.

hsl ()

Calculate the HSL color values from the RGB color.

Returns Tuple containing three normalized doubles, between 0.0 & 1.0, representing hue, saturation, and lightness.

Return type `collections.Sequence`

New in version 0.5.1.

magenta

(`numbers.Real`) Magenta color channel in CMYK colorspace. Unused by RGB colorspace.

New in version 0.5.1.

magenta_int8

(`numbers.Integral`) Magenta value as 8bit integer which is a common style. From 0 to 255.

New in version 0.5.1.

normalized_string

(`basestring`) The normalized string representation of the color. The same color is always represented to the same string.

New in version 0.3.0.

red

(`numbers.Real`) Red, from 0.0 to 1.0.

red_int8

(`numbers.Integral`) Red as 8bit integer which is a common style. From 0 to 255.

New in version 0.3.0.

red_quantum

(`numbers.Integral`) Red. Scale depends on `QUANTUM_DEPTH`.

New in version 0.3.0.

string

(`basestring`) The string representation of the color.

yellow

(`numbers.Real`) Yellow color channel in CMYK colorspace. Unused by RGB colorspace.

New in version 0.5.1.

yellow_int8

(`numbers.Integral`) Yellow as 8bit integer which is a common style. From 0 to 255.

New in version 0.5.1.

yellow_quantum

(`numbers.Integral`) Yellow. Scale depends on `QUANTUM_DEPTH`.

New in version 0.5.1.

`wand.color.scale_quantum_to_int8(quantum)`

Straightforward port of `ScaleQuantumToChar()` inline function.

Parameters `quantum` (`numbers.Integral`) – quantum value

Returns 8bit integer of the given quantum value

Return type `numbers.Integral`

New in version 0.3.0.

Changed in version 0.5.0: Added HDRI support

4.1.3 `wand.font` — Fonts

New in version 0.3.0.

`Font` is an object which takes the `path` of font file, `size`, `color`, and whether to use `antialiasing`. If you want to use font by its name rather than the file path, use `TTFQuery` package. The font path resolution by its name is a very complicated problem to achieve.

See also:

TTFQuery — Find and Extract Information from TTF Files TTFQuery builds on the [FontTools-TTX](#) package to allow the Python programmer to accomplish a number of tasks:

- query the system to find installed fonts
- retrieve metadata about any TTF font file
 - this includes the glyph outlines (shape) of individual code-points, which allows for rendering the glyphs in 3D (such as is done in OpenGLContext)
- lookup/find fonts by:
 - abstract family type
 - proper font name
- build simple metadata registries for run-time font matching

class `wand.font.Font`

Font struct which is a subtype of `tuple`.

Parameters

- **path** (`str`, `basestring`) – the path of the font file
- **size** (`numbers.Real`) – the size of typeface. 0 by default which means *autosized*
- **color** (`Color`) – the color of typeface. black by default
- **antialias** (`bool`) – whether to use antialiasing. True by default
- **stroke_color** (`Color`) – optional color to outline typeface.
- **stroke_width** (`numbers.Real`) – optional thickness of typeface outline.

Changed in version 0.3.9: The `size` parameter becomes optional. Its default value is 0, which means *autosized*.

Changed in version 0.5.0: Added `stroke_color` & `stroke_width` paramaters.

antialias

(`bool`) Whether to apply antialiasing (True) or not (False).

color

(`wand.color.Color`) The font color.

path

(`basestring`) The path of font file.

size

(`numbers.Real`) The font size in pixels.

stroke_color

(`wand.color.Color`) The stroke color.

stroke_width

(`numbers.Real`) The width of the stroke line.

4.1.4 `wand.drawing` — Drawings

The module provides some vector drawing functions.

New in version 0.3.0.

`wand.drawing.CLIP_PATH_UNITS = ('undefined_path_units', 'user_space', 'user_space_on_use',`
(`collections.abc.Sequence`) The list of clip path units

- 'undefined_path_units'
- 'user_space'
- 'user_space_on_use'
- 'object_bounding_box'

wand.drawing.**FILL_RULE_TYPES** = ('undefined', 'evenodd', 'nonzero')
(collections.abc.Sequence) The list of fill-rule types.

- 'undefined'
- 'evenodd'
- 'nonzero'

wand.drawing.**FONT_METRICS_ATTRIBUTES** = ('character_width', 'character_height', 'ascender',
(collections.abc.Sequence) The attribute names of font metrics.

wand.drawing.**GRAVITY_TYPES** = ('forget', 'north_west', 'north', 'north_east', 'west', 'center',
(collections.abc.Sequence) The list of text gravity types.

- 'forget'
- 'north_west'
- 'north'
- 'north_east'
- 'west'
- 'center'
- 'east'
- 'south_west'
- 'south'
- 'south_east'
- 'static'

wand.drawing.**LINE_CAP_TYPES** = ('undefined', 'butt', 'round', 'square')
(collections.abc.Sequence) The list of LineCap types

- 'undefined';
- 'butt'
- 'round'
- 'square'

wand.drawing.**LINE_JOIN_TYPES** = ('undefined', 'miter', 'round', 'bevel')
(collections.abc.Sequence) The list of LineJoin types

- 'undefined'
- 'miter'
- 'round'
- 'bevel'

wand.drawing.**PAINT_METHOD_TYPES** = ('undefined', 'point', 'replace', 'floodfill', 'filltoborder',
(collections.abc.Sequence) The list of paint method types.

- 'undefined'
- 'point'
- 'replace'
- 'floodfill'
- 'filltoborder'
- 'reset'

wand.drawing.**STRETCH_TYPES** = ('undefined', 'normal', 'ultra_condensed', 'extra_condensed',
(collections.abc.Sequence) The list of stretch types for fonts

- 'undefined;
- 'normal'
- 'ultra_condensed'
- 'extra_condensed'
- 'condensed'
- 'semi_condensed'
- 'semi_expanded'
- 'expanded'
- 'extra_expanded'
- 'ultra_expanded'
- 'any'

wand.drawing.**STYLE_TYPES** = ('undefined', 'normal', 'italic', 'oblique', 'any')
(collections.abc.Sequence) The list of style types for fonts

- 'undefined;
- 'normal'
- 'italic'
- 'oblique'
- 'any'

wand.drawing.**TEXT_ALIGN_TYPES** = ('undefined', 'left', 'center', 'right')
(collections.abc.Sequence) The list of text align types.

- 'undefined'
- 'left'
- 'center'
- 'right'

wand.drawing.**TEXT_DECORATION_TYPES** = ('undefined', 'no', 'underline', 'overline', 'line_th
(collections.abc.Sequence) The list of text decoration types.

- 'undefined'
- 'no'
- 'underline'

- 'overline'
- 'line_through'

wand.drawing.TEXT_DIRECTION_TYPES = ('undefined', 'right_to_left', 'left_to_right')
(`collections.abc.Sequence`) The list of text direction types.

- 'undefined'
- 'right_to_left'
- 'left_to_right'

class wand.drawing.Drawing (*drawing=None*)

Drawing object. It maintains several vector drawing instructions and can get drawn into zero or more *Image* objects by calling it.

For example, the following code draws a diagonal line to the image:

```
with Drawing() as draw:
    draw.line((0, 0), image.size)
    draw(image)
```

Parameters *drawing* (*Drawing*) – an optional drawing object to clone. use *clone()* method rather than this parameter

New in version 0.3.0.

affine (*matrix*)

Adjusts the current affine transformation matrix with the specified affine transformation matrix. Note that the current affine transform is adjusted rather than replaced.

x', y', 1	=	x, y, 1	*	sx rx 0
				ry sy 0
				tx ty 1

Parameters *matrix* (`collections.abc.Sequence`) – a list of *Real* to define affine matrix [sx, rx, ry, sy, tx, ty]

New in version 0.4.0.

alpha (*x=None, y=None, paint_method='undefined'*)

Paints on the image's opacity channel in order to set effected pixels to transparent.

To influence the opacity of pixels. The available methods are:

- 'undefined'
- 'point'
- 'replace'
- 'floodfill'
- 'filltoborder'
- 'reset'

Note: This method replaces `matte()` in ImageMagick version 7. An `AttributeError` will be raised if attempting to call on a library without DrawAlpha support.

New in version 0.5.0.

arc (*start, end, degree*)

Draws a arc using the current `stroke_color`, `stroke_width`, and `fill_color`.

Parameters

- **start** (`Sequence`) – (`Real`, `numbers.Real`) pair which represents starting x and y of the arc
- **end** (`Sequence`) – (`Real`, `numbers.Real`) pair which represents ending x and y of the arc
- **degree** (`Sequence`) – (`Real`, `numbers.Real`) pair which represents starting degree, and ending degree

New in version 0.4.0.

bezier (*points=None*)

Draws a bezier curve through a set of points on the image, using the specified array of coordinates.

At least four points should be given to complete a bezier path. The first & forth point being the start & end point, and the second & third point controlling the direction & curve.

Example bezier on image

```
with Drawing() as draw:
    points = [(40,10), # Start point
              (20,50), # First control
              (90,10), # Second control
              (70,40)] # End point
    draw.stroke_color = Color('#000')
    draw.fill_color = Color('#fff')
    draw.bezier(points)
    draw.draw(image)
```

Parameters **points** (`list`) – list of x,y tuples

New in version 0.4.0.

border_color

(`Color`) the current border color. It also can be set. This attribute controls the behavior of `color()` during 'filltoborder' operation.

New in version 0.4.0.

circle (*origin, perimeter*)

Draws a circle from origin to perimeter

Parameters

- **origin** (`collections.abc.Sequence`) – (`Real`, `numbers.Real`) pair which represents origin x and y of circle
- **perimeter** (`collections.abc.Sequence`) – (`Real`, `numbers.Real`) pair which represents perimeter x and y of circle

New in version 0.4.0.

clip_path

(basestring) The current clip path. It also can be set.

New in version 0.4.0.

clip_rule

(basestring) The current clip rule. It also can be set. It's a string value from `FILL_RULE_TYPES` list.

New in version 0.4.0.

clip_units

(basestring) The current clip units. It also can be set. It's a string value from `CLIP_PATH_UNITS` list.

New in version 0.4.0.

clone()

Copies a drawing object.

Returns a duplication

Return type `Drawing`

color (*x=None, y=None, paint_method='undefined'*)

Draws a color on the image using current fill color, starting at specified position & method.

Available methods in `wand.drawing.PAINT_METHOD_TYPES`:

- 'undefined'
- 'point'
- 'replace'
- 'floodfill'
- 'filltoborder'
- 'reset'

New in version 0.4.0.

comment (*message=None*)

Adds a comment to the vector stream.

Parameters **message** (basestring) – the comment to set.

New in version 0.4.0.

composite (*operator, left, top, width, height, image*)

Composites an image onto the current image, using the specified composition operator, specified position, and at the specified size.

Parameters

- **operator** – the operator that affects how the composite is applied to the image. available values can be found in the `COMPOSITE_OPERATORS` list
- **type** – `COMPOSITE_OPERATORS`
- **left** (`numbers.Real`) – the column offset of the composited drawing source
- **top** (`numbers.Real`) – the row offset of the composited drawing source
- **width** (`numbers.Real`) – the total columns to include in the composited source
- **height** (`numbers.Real`) – the total rows to include in the composited source

New in version 0.4.0.

draw (*image*)

Renders the current drawing into the *image*. You can simply call *Drawing* instance rather than calling this method. That means the following code which calls *Drawing* object itself:

```
drawing(image)
```

is equivalent to the following code which calls *draw()* method:

```
drawing.draw(image)
```

Parameters *image* (*BaseImage*) – the image to be drawn

ellipse (*origin*, *radius*, *rotation*=(0, 360))

Draws a ellipse at *origin* with independent x & y radius. Ellipse can be partial by setting start & end rotation.

Parameters

- **origin** (*collections.abc.Sequence*) – (*Real*, *numbers.Real*) pair which represents origin x and y of circle
- **radius** (*collections.abc.Sequence*) – (*Real*, *numbers.Real*) pair which represents radius x and radius y of circle
- **rotation** (*collections.abc.Sequence*) – (*Real*, *numbers.Real*) pair which represents start and end of ellipse. Default (0,360)

New in version 0.4.0.

fill_color

(*Color*) The current color to fill. It also can be set.

fill_opacity

(*Real*) The current fill opacity. It also can be set.

New in version 0.4.0.

fill_rule

(*basestring*) The current fill rule. It can also be set. It's a string value from *FILL_RULE_TYPES* list.

New in version 0.4.0.

font

(*basestring*) The current font name. It also can be set.

font_family

(*basestring*) The current font family. It also can be set.

New in version 0.4.0.

font_resolution

(*Sequence*) The current font resolution. It also can be set.

New in version 0.4.0.

font_size

(*numbers.Real*) The font size. It also can be set.

font_stretch

(*basestring*) The current font stretch variation. It also can be set, but will only apply if the font-family or encoder supports the stretch type.

New in version 0.4.0.

font_style

(*basestring*) The current font style. It also can be set, but will only apply if the font-family supports the style.

New in version 0.4.0.

font_weight

(*Integral*) The current font weight. It also can be set.

New in version 0.4.0.

get_font_metrics (*image, text, multiline=False*)

Queries font metrics from the given *text*.

Parameters

- **image** (*BaseImage*) – the image to be drawn
- **text** (*basestring*) – the text string for get font metrics.
- **multiline** (*boolean*) – text is multiline or not

gravity

(*basestring*) The text placement gravity used when annotating with text. It's a string from *GRAVITY_TYPES* list. It also can be set.

line (*start, end*)

Draws a line *start* to *end*.

Parameters

- **start** (*collections.abc.Sequence*) – (*Integral, numbers.Integral*) pair which represents starting *x* and *y* of the line
- **end** (*collections.abc.Sequence*) – (*Integral, numbers.Integral*) pair which represents ending *x* and *y* of the line

matte (*x=None, y=None, paint_method='undefined'*)

Paints on the image's opacity channel in order to set effected pixels to transparent.

To influence the opacity of pixels. The available methods are:

- 'undefined'
- 'point'
- 'replace'
- 'floodfill'
- 'filltoborder'
- 'reset'

Note: This method has been replace by *alpha()* in ImageMagick version 7. An *AttributeError* will be raised if attempting to call on a library without *DrawMatte* support.

New in version 0.4.0.

opacity

(*Real*) returns the opacity used when drawing with the fill or stroke color or texture. Fully opaque is 1.0. This method only affects vector graphics, and is experimental. To set the opacity of a drawing, use *Drawing.fill_opacity* & *Drawing.stroke_opacity*

New in version 0.4.0.

path_close()

Adds a path element to the current path which closes the current subpath by drawing a straight line from the current point to the current subpath's most recent starting point.

New in version 0.4.0.

path_curve (*to=None, controls=None, smooth=False, relative=False*)

Draws a cubic Bezier curve from the current point to given *to* (x,y) coordinate using *controls* points at the beginning and the end of the curve. If *smooth* is set to True, only one *controls* is expected and the previous control is used, else two pair of coordinates are expected to define the control points. The *to* coordinate then becomes the new current point.

Parameters

- **to** (*collections.abc.Sequence*) – (*Real, numbers.Real*) pair which represents coordinates to draw to
- **controls** (*collections.abc.Sequence*) – (*Real, numbers.Real*) coordinate to used to influence curve
- **smooth** (*bool*) – *bool* assume last defined control coordinate
- **relative** (*bool*) – treat given coordinates as relative to current point

New in version 0.4.0.

path_curve_to_quadratic_bezier (*to=None, control=None, smooth=False, relative=False*)

Draws a quadratic Bezier curve from the current point to given *to* coordinate. The control point is assumed to be the reflection of the control point on the previous command if *smooth* is True, else a pair of control coordinates must be given. Each coordinates can be relative, or absolute, to the current point by setting the *relative* flag. The *to* coordinate then becomes the new current point, and the *control* coordinate will be assumed when called again when *smooth* is set to true.

Parameters

- **to** (*collections.abc.Sequence*) – (*Real, numbers.Real*) pair which represents coordinates to draw to
- **control** (*collections.abc.Sequence*) – (*Real, numbers.Real*) coordinate to used to influence curve
- **smooth** (*bool*) – assume last defined control coordinate
- **relative** (*bool*) – treat given coordinates as relative to current point

New in version 0.4.0.

path_elliptic_arc (*to=None, radius=None, rotation=0.0, large_arc=False, clockwise=False, relative=False*)

Draws an elliptical arc from the current point to given *to* coordinates. The *to* coordinates can be relative, or absolute, to the current point by setting the *relative* flag. The size and orientation of the ellipse are defined by two radii (rx, ry) in *radius* and an *rotation* parameters, which indicates how the ellipse as a whole is rotated relative to the current coordinate system. The center of the ellipse is calculated automatically to satisfy the constraints imposed by the other parameters. *large_arc* and *clockwise* contribute to the automatic calculations and help determine how the arc is drawn. If *large_arc* is True

then draw the larger of the available arcs. If `clockwise` is `true`, then draw the arc matching a clock-wise rotation.

Parameters

- **to** (`collections.abc.Sequence`) – (`Real`, `numbers.Real`) pair which represents coordinates to draw to
- **radius** (`collections.abc.Sequence`) – (`Real`, `numbers.Real`) pair which represents the radii of the ellipse to draw
- **rotate** (`Real`) – degree to rotate ellipse on x-axis
- **large_arc** (`bool`) – draw largest available arc
- **clockwise** (`bool`) – draw arc path clockwise from start to target
- **relative** (`bool`) – treat given coordinates as relative to current point

New in version 0.4.0.

path_finish()

Terminates the current path.

New in version 0.4.0.

path_horizontal_line (*x=None, relative=False*)

Draws a horizontal line path from the current point to the target point. Given `x` parameter can be relative, or absolute, to the current point by setting the `relative` flag. The target point then becomes the new current point.

Parameters

- **x** (`Real`) – `Real` x-axis point to draw to.
- **relative** (`bool`) – `bool` treat given point as relative to current point

New in version 0.4.0.

path_line (*to=None, relative=False*)

Draws a line path from the current point to the given `to` coordinate. The `to` coordinates can be relative, or absolute, to the current point by setting the `relative` flag. The coordinate then becomes the new current point.

Parameters

- **to** (`collections.abc.Sequence`) – (`Real`, `numbers.Real`) pair which represents coordinates to draw to.
- **relative** (`bool`) – `bool` treat given coordinates as relative to current point

New in version 0.4.0.

path_move (*to=None, relative=False*)

Starts a new sub-path at the given coordinates. Given `to` parameter can be relative, or absolute, by setting the `relative` flag.

Parameters

- **to** (`collections.abc.Sequence`) – (`Real`, `numbers.Real`) pair which represents coordinates to draw to.
- **relative** (`bool`) – `bool` treat given coordinates as relative to current point

New in version 0.4.0.

path_start()

Declares the start of a path drawing list which is terminated by a matching `path_finish()` command. All other `path_*` commands must be enclosed between a `path_start()` and a `path_finish()` command. This is because path drawing commands are subordinate commands and they do not function by themselves.

New in version 0.4.0.

path_vertical_line (*y=None, relative=False*)

Draws a vertical line path from the current point to the target point. Given `y` parameter can be relative, or absolute, to the current point by setting the `relative` flag. The target point then becomes the new current point.

Parameters

- **y** (*Real*) – Real y-axis point to draw to.
- **relative** (*bool*) – *bool* treat given point as relative to current point

New in version 0.4.0.

point (*x, y*)

Draws a point at given `x` and `y`

Parameters

- **x** (*Real*) – Real x of point
- **y** (*Real*) – Real y of point

New in version 0.4.0.

polygon (*points=None*)

Draws a polygon using the current `stoke_color`, `stroke_width`, and `fill_color`, using the specified array of coordinates.

Example polygon on image

```
with Drawing() as draw:
    points = [(40,10), (20,50), (90,10), (70,40)]
    draw.polygon(points)
    draw.draw(image)
```

Parameters **points** (*list*) – list of x,y tuples

New in version 0.4.0.

polyline (*points=None*)

Draws a polyline using the current `stoke_color`, `stroke_width`, and `fill_color`, using the specified array of coordinates.

Identical to `polygon`, but without closed stroke line.

Parameters **points** (*list*) – list of x,y tuples

New in version 0.4.0.

pop()

Pop destroys the current tip of the drawing context stack, and restores the parent style context. See `push()` method for an example.

Note: Popping the graphical context stack will not erase, or alter, any previously executed drawing commands.

Returns success of pop operation.

Return type *bool*

New in version 0.4.0.

pop_clip_path()

Terminates a clip path definition.

New in version 0.4.0.

pop_defs()

Terminates a definition list.

New in version 0.4.0.

pop_pattern()

Terminates a pattern definition.

New in version 0.4.0.

push()

Grows the current drawing context stack by one, and inherits the previous style attributes. Use *Drawing.pop* to return to restore previous style attributes.

This is useful for drawing shapes with different styles without repeatedly setting the similar *fill_color* & *stroke_color* properties.

For example:

```
with Drawing() as ctx:
    ctx.fill_color = Color('GREEN')
    ctx.stroke_color = Color('ORANGE')
    ctx.push()
    ctx.fill_color = Color('RED')
    ctx.text(x1, y1, 'this is RED with ORANGE outline')
    ctx.push()
    ctx.stroke_color = Color('BLACK')
    ctx.text(x2, y2, 'this is RED with BLACK outline')
    ctx.pop()
    ctx.pop()
    ctx.text(x3, y3, 'this is GREEN with ORANGE outline')
```

Which translate to the following MVG:

```
push graphic-context
  fill "GREEN"
  stroke "ORANGE"
  push graphic-context
    fill "RED"
    text x1,y1 "this is RED with ORANGE outline"
    push graphic-context
      stroke "BLACK"
      text x2,y2 "this is RED with BLACK outline"
    pop graphic-context
```

(continues on next page)

(continued from previous page)

```
pop graphic-context
text x3,y3 "this is GREEN with ORANGE outline"
pop graphic-context
```

Note: Pushing graphical context does not reset any previously drawn artifacts.

Returns success of push operation.

Return type *bool*

New in version 0.4.0.

push_clip_path (*clip_mask_id*)

Starts a clip path definition which is comprised of any number of drawing commands and terminated by a *Drawing.pop_clip_path* command.

Parameters *clip_mask_id* (basestring) – string identifier to associate with the clip path.

New in version 0.4.0.

push_defs ()

Indicates that commands up to a terminating *Drawing.pop_defs* command create named elements (e.g. clip-paths, textures, etc.) which may safely be processed earlier for the sake of efficiency.

New in version 0.4.0.

push_pattern (*pattern_id*, *left*, *top*, *width*, *height*)

Indicates that subsequent commands up to a *Drawing.pop_pattern* command comprise the definition of a named pattern. The pattern space is assigned top left corner coordinates, a width and height, and becomes its own drawing space. Anything which can be drawn may be used in a pattern definition. Named patterns may be used as stroke or brush definitions.

Parameters

- **pattern_id** (basestring) – a unique identifier for the pattern.
- **left** (*numbers.Real*) – x ordinate of top left corner.
- **top** (*numbers.Real*) – y ordinate of top left corner.
- **width** (*numbers.Real*) – width of pattern space.
- **height** (*numbers.Real*) – height of pattern space.

Returns success of push operation

Return type *bool*

New in version 0.4.0.

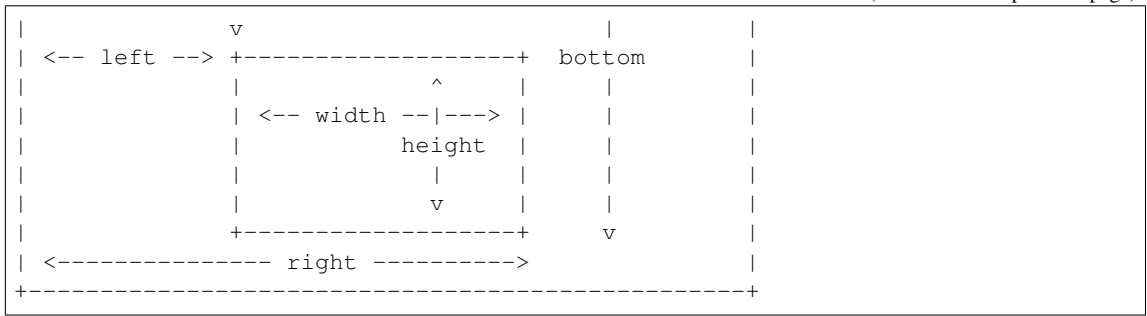
rectangle (*left=None*, *top=None*, *right=None*, *bottom=None*, *width=None*, *height=None*, *radius=None*, *xradius=None*, *yradius=None*)

Draws a rectangle using the current *stroke_color*, *stroke_width*, and *fill_color*.

```
+-----+
|               ^               ^               |
|               |               |               |
|               top            |               |
|               |               |               |
+-----+
```

(continues on next page)

(continued from previous page)

**Parameters**

- **left** (`numbers.Real`) – x-offset of the rectangle to draw
- **top** (`numbers.Real`) – y-offset of the rectangle to draw
- **right** (`numbers.Real`) – second x-offset of the rectangle to draw. this parameter and `width` parameter are exclusive each other
- **bottom** (`numbers.Real`) – second y-offset of the rectangle to draw. this parameter and `height` parameter are exclusive each other
- **width** (`numbers.Real`) – the width of the rectangle to draw. this parameter and `right` parameter are exclusive each other
- **height** (`numbers.Real`) – the height of the rectangle to draw. this parameter and `bottom` parameter are exclusive each other
- **radius** (`numbers.Real`) – the corner rounding. this is a short-cut for setting both `xradius`, and `yradius`
- **xradius** (`numbers.Real`) – the `xradius` corner in horizontal direction.
- **yradius** (`numbers.Real`) – the `yradius` corner in vertical direction.

New in version 0.3.6.

Changed in version 0.4.0: Radius keywords added to create rounded rectangle.

rotate (*degree*)

Applies the specified rotation to the current coordinate space.

Parameters **degree** (`Real`) – degree to rotate

New in version 0.4.0.

scale (*x=None, y=None*)

Adjusts the scaling factor to apply in the horizontal and vertical directions to the current coordinate space.

Parameters

- **x** (`Real`) – Horizontal scale factor
- **y** (`Real`) – Vertical scale factor

New in version 0.4.0.

set_fill_pattern_url (*url*)

Sets the URL to use as a fill pattern for filling objects. Only local URLs (“#identifier”) are supported at this time. These local URLs are normally created by defining a named fill pattern with `Drawing.push_pattern` & `Drawing.pop_pattern`.

Parameters `url` (basestring) – URL to use to obtain fill pattern.

New in version 0.4.0.

set_stroke_pattern_url (`url`)

Sets the pattern used for stroking object outlines. Only local URLs (“#identifier”) are supported at this time. These local URLs are normally created by defining a named stroke pattern with `Drawing.push_pattern` & `Drawing.pop_pattern`.

Parameters `url` (basestring) – URL to use to obtain stroke pattern.

New in version 0.4.0.

skew (`x=None`, `y=None`)

Skews the current coordinate system in the horizontal direction if `x` is given, and vertical direction if `y` is given.

Parameters

- `x` (Real) – Skew horizontal direction
- `y` (Real) – Skew vertical direction

New in version 0.4.0.

stroke_antialias

(`bool`) Controls whether stroked outlines are antialiased. Stroked outlines are antialiased by default. When antialiasing is disabled stroked pixels are thresholded to determine if the stroke color or underlying canvas color should be used.

It also can be set.

New in version 0.4.0.

stroke_color

(`Color`) The current color of stroke. It also can be set.

New in version 0.3.3.

stroke_dash_array

(`Sequence`) - (`numbers.Real`) An array representing the pattern of dashes & gaps used to stroke paths. It also can be set.

New in version 0.4.0.

stroke_dash_offset

(`numbers.Real`) The stroke dash offset. It also can be set.

New in version 0.4.0.

stroke_line_cap

(basestring) The stroke line cap. It also can be set.

New in version 0.4.0.

stroke_line_join

(basestring) The stroke line join. It also can be set.

New in version 0.4.0.

stroke_miter_limit

(`Integral`) The current miter limit. It also can be set.

New in version 0.4.0.

stroke_opacity

(*Real*) The current stroke opacity. It also can be set.

New in version 0.4.0.

stroke_width

(*numbers.Real*) The stroke width. It also can be set.

New in version 0.3.3.

text (*x, y, body*)

Writes a text *body* into (*x, y*).

Parameters

- **x** (*numbers.Integral*) – the left offset where to start writing a text
- **y** (*numbers.Integral*) – the baseline where to start writing text
- **body** (*basestring*) – the body string to write

text_alignment

(*basestring*) The current text alignment setting. It's a string value from *TEXT_ALIGN_TYPES* list. It also can be set.

text_antialias

(*bool*) The boolean value which represents whether antialiasing is used for text rendering. It also can be set to *True* or *False* to switch the setting.

text_decoration

(*basestring*) The text decoration setting, a string from *TEXT_DECORATION_TYPES* list. It also can be set.

text_direction

(*basestring*) The text direction setting. a string from *TEXT_DIRECTION_TYPES* list. It also can be set.

text_encoding

(*basestring*) The internally used text encoding setting. Although it also can be set, but it's not encouraged.

text_interline_spacing

(*numbers.Real*) The setting of the text line spacing. It also can be set.

text_interword_spacing

(*numbers.Real*) The setting of the word spacing. It also can be set.

text_kerning

(*numbers.Real*) The setting of the text kerning. It also can be set.

text_under_color

(*Color*) The color of a background rectangle to place under text annotations. It also can be set.

translate (*x=None, y=None*)

Applies a translation to the current coordinate system which moves the coordinate system origin to the specified coordinate.

Parameters

- **x** (*Real*) – Skew horizontal direction
- **y** (*Real*) – Skew vertical direction

New in version 0.4.0.

vector_graphics

(*basestring*) The XML text of the Vector Graphics. It also can be set. The drawing-wand XML is experimental, and subject to change.

Setting this property to None will reset all vector graphic properties to the default state.

New in version 0.4.0.

viewbox (*left, top, right, bottom*)

Viewbox sets the overall canvas size to be recorded with the drawing vector data. Usually this will be specified using the same size as the canvas image. When the vector data is saved to SVG or MVG formats, the viewbox is used to specify the size of the canvas image that a viewer will render the vector data on.

Parameters

- **left** (*Integral*) – the left most point of the viewbox.
- **top** (*Integral*) – the top most point of the viewbox.
- **right** (*Integral*) – the right most point of the viewbox.
- **bottom** (*Integral*) – the bottom most point of the viewbox.

New in version 0.4.0.

class wand.drawing.**FontMetrics** (*character_width, character_height, ascender, descender, text_width, text_height, maximum_horizontal_advance, x1, y1, x2, y2, x, y*)

The tuple subtype which consists of font metrics data.

ascender

Alias for field number 2

character_height

Alias for field number 1

character_width

Alias for field number 0

descender

Alias for field number 3

maximum_horizontal_advance

Alias for field number 6

text_height

Alias for field number 5

text_width

Alias for field number 4

x

Alias for field number 11

x1

Alias for field number 7

x2

Alias for field number 9

y

Alias for field number 12

y1

Alias for field number 8

y2

Alias for field number 10

4.1.5 wand.sequence — Sequences

New in version 0.3.0.

class wand.sequence.Sequence(*image*)

The list-like object that contains every *SingleImage* in the *Image* container. It implements `collections.abc.Sequence` protocol.

New in version 0.3.0.

append(*image*)

S.append(object) – append object to the end of the sequence

current_index

(`numbers.Integral`) The current index of its internal iterator.

Note: It's only for internal use.

extend(*images*, *offset=None*)

S.extend(iterable) – extend sequence by appending elements from the iterable

index_context(***kws*)

Scoped setter of `current_index`. Should be used for `with` statement e.g.:

```
with image.sequence.index_context(3):
    print(image.size)
```

Note: It's only for internal use.

insert(*index*, *image*)

S.insert(index, object) – insert object before index

class wand.sequence.SingleImage(*wand*, *container*, *c_original_resource*)

Each single image in *Image* container. For example, it can be a frame of GIF animation.

Note that all changes on single images are invisible to their containers unless they are altered a `with ...` context manager.

with Image(filename='animation.gif') as container:

with container.sequence[0] as frame: frame.negate()

New in version 0.3.0.

Changed in version 0.5.1: Only sync changes of a *SingleImage* when exiting a `with ...` context. Not when parent *Image* closes.

container = None

(`wand.image.Image`) The container image.

delay

(`numbers.Integral`) The delay to pause before display the next image (in the *sequence* of its *container*). It's hundredths of a second.

index

(`numbers.Integral`) The index of the single image in the *container* image.

4.1.6 wand.resource — Global resource management

There is the global resource to manage in MagickWand API. This module implements automatic global resource management through reference counting.

`wand.resource.genesis()`
Instantiates the MagickWand API.

Warning: Don't call this function directly. Use `increment_refcount()` and `decrement_refcount()` functions instead.

`wand.resource.terminus()`
Cleans up the MagickWand API.

Warning: Don't call this function directly. Use `increment_refcount()` and `decrement_refcount()` functions instead.

`wand.resource.increment_refcount()`
Increments the `reference_count` and instantiates the MagickWand API if it is the first use.

`wand.resource.decrement_refcount()`
Decrements the `reference_count` and cleans up the MagickWand API if it will be no more used.

`wand.resource.limits = <wand.resource.ResourceLimits object>`
(*ResourceLimits*) Helper to get & set Magick Resource Limits.

New in version 0.5.1.

class `wand.resource.Resource`

Abstract base class for MagickWand object that requires resource management. Its all subclasses manage the resource semiautomatically and support `with` statement as well:

```
with Resource() as resource:
    # use the resource...
    pass
```

It doesn't implement constructor by itself, so subclasses should implement it. Every constructor should assign the pointer of its resource data into `resource` attribute inside of `with allocate()` context. For example:

```
class Pizza(Resource):
    '''My pizza yummy.'''

    def __init__(self):
        with self.allocate():
            self.resource = library.NewPizza()
```

New in version 0.1.2.

allocate (***kws*)

Allocates the memory for the resource explicitly. Its subclasses should assign the created resource into `resource` attribute inside of this context. For example:

```
with resource.allocate():
    resource.resource = library.NewResource()
```

c_clear_exception = NotImplemented

(ctypes.CFUNCTYPE) The `ctypes` function that clears an exception of the `resource`.

Note: It is an abstract attribute that has to be implemented in the subclass.

c_destroy_resource = NotImplemented

(ctypes.CFUNCTYPE) The `ctypes` function that destroys the `resource`.

Note: It is an abstract attribute that has to be implemented in the subclass.

c_get_exception = NotImplemented

(ctypes.CFUNCTYPE) The `ctypes` function that gets an exception from the `resource`.

Note: It is an abstract attribute that has to be implemented in the subclass.

c_is_resource = NotImplemented

(ctypes.CFUNCTYPE) The `ctypes` predicate function that returns whether the given pointer (that contains a resource data usually) is a valid resource.

Note: It is an abstract attribute that has to be implemented in the subclass.

destroy()

Cleans up the resource explicitly. If you use the resource in `with` statement, it was called implicitly so have not to call it.

get_exception()

Gets a current exception instance.

Returns a current exception. it can be `None` as well if any errors aren't occurred

Return type `wand.exceptions.WandException`

raise_exception(stacklevel=1)

Raises an exception or warning if it has occurred.

resource

Internal pointer to the resource instance. It may raise `DestroyedResourceError` when the resource has destroyed already.

class wand.resource.ResourceLimits

Wrapper for MagickCore resource limits. Useful for dynamically reducing system resources before attempting risky, or slow running, `Image` operations.

For example:

```
from wand.image import Image
from wand.resource import limits

# Use 100MB of ram before writing temp data to disk.
limits['memory'] = 1024 * 1024 * 100
```

(continues on next page)

(continued from previous page)

```
# Reject images larger than 1000x1000.
limits['width'] = 1000
limits['height'] = 1000

# Debug resources used.
with Image(filename='user.jpg') as img:
    print('Using {0} of {1} memory'.format(limits.resource('memory'),
                                          limits['memory']))

# Dump list of all limits.
for label in limits:
    print('{0} => {1}'.format(label, limits[label]))
```

Available resource keys:

- 'area' - Maximum *width* * *height* of a pixel cache before writing to disk.
- 'disk' - Maximum bytes used by pixel cache on disk before exception is thrown.
- 'file' - Maximum cache files opened at any given time.
- 'height' - Maximum height of image before exception is thrown.
- 'list_length' - Maximum images in sequence. Only available with recent version of ImageMagick.
- 'map' - Maximum memory map in bytes to allocated for pixel cache before using disk.
- 'memory' - Maximum bytes to allocated for pixel cache before using disk.
- 'thread' - Maximum parallel task sub-routines can spawn - if using OpenMP.
- 'throttle' - Total milliseconds to yield to CPU - if possible.
- 'time' - Maximum seconds before exception is thrown.
- 'width' - Maximum width of image before exception is thrown.

New in version 0.5.1.

get_resource_limit (*resource*)

Get the current limit for the resource type.

Parameters **resource** (basestring) – Resource type.

Return type numeric.Integral

New in version 0.5.1.

resource (*resource*)

Get the current value for the resource type.

Parameters **resource** (basestring) – Resource type.

Return type numeric.Integral

New in version 0.5.1.

set_resource_limit (*resource*, *limit*)

Sets a new limit for resource type.

Note: The new limit value must be equal to, or less then, the maximum limit defined by the `policy.xml`. Any values set outside normal bounds will be ignored silently.

Parameters

- **resource** (basestring) – Resource type.
- **limit** (numeric.Integral) – New limit value.

New in version 0.5.1.

exception `wand.resource.DestroyedResourceError`

An error that rises when some code tries access to an already destroyed resource.

Changed in version 0.3.0: It becomes a subtype of `wand.exceptions.WandException`.

4.1.7 `wand.exceptions` — Errors and warnings

This module maps MagickWand API's errors and warnings to Python's native exceptions and warnings. You can catch all MagickWand errors using Python's natural way to catch errors.

See also:

[ImageMagick Exceptions](#)

New in version 0.1.1.

exception `wand.exceptions.BaseError`

Bases: `wand.exceptions.WandException`

Base class for Wand-related errors.

New in version 0.4.4.

exception `wand.exceptions.BaseFatalError`

Bases: `wand.exceptions.WandException`

Base class for Wand-related fatal errors.

New in version 0.4.4.

exception `wand.exceptions.BaseWarning`

Bases: `wand.exceptions.WandException`, `exceptions.Warning`

Base class for Wand-related warnings.

New in version 0.4.4.

exception `wand.exceptions.BlobError`

Bases: `wand.exceptions.BaseError`, `exceptions.IOError`

A binary large object could not be allocated, read, or written.

exception `wand.exceptions.BlobFatalError`

Bases: `wand.exceptions.BaseFatalError`, `exceptions.IOError`

A binary large object could not be allocated, read, or written.

exception `wand.exceptions.BlobWarning`

Bases: `wand.exceptions.BaseWarning`, `exceptions.IOError`

A binary large object could not be allocated, read, or written.

`wand.exceptions.CODE_MAP = [(<class 'wand.exceptions.BaseWarning'>, 'Warning'), (<class 'wand.exceptions.BlobError'>, 'BlobError'), (<class 'wand.exceptions.BlobFatalError'>, 'BlobFatalError'), (<class 'wand.exceptions.BlobWarning'>, 'BlobWarning')]`
(list) The list of (base_class, suffix) pairs (for each code). It would be zipped with `DOMAIN_MAP` pairs' last element.

exception `wand.exceptions.CacheError`

Bases: `wand.exceptions.BaseError`

Pixels could not be read or written to the pixel cache.

exception `wand.exceptions.CacheFatalError`

Bases: `wand.exceptions.BaseFatalError`

Pixels could not be read or written to the pixel cache.

exception `wand.exceptions.CacheWarning`

Bases: `wand.exceptions.BaseWarning`

Pixels could not be read or written to the pixel cache.

exception `wand.exceptions.CoderError`

Bases: `wand.exceptions.BaseError`

There was a problem with an image coder.

exception `wand.exceptions.CoderFatalError`

Bases: `wand.exceptions.BaseFatalError`

There was a problem with an image coder.

exception `wand.exceptions.CoderWarning`

Bases: `wand.exceptions.BaseWarning`

There was a problem with an image coder.

exception `wand.exceptions.ConfigureError`

Bases: `wand.exceptions.BaseError`

There was a problem getting a configuration file.

exception `wand.exceptions.ConfigureFatalError`

Bases: `wand.exceptions.BaseFatalError`

There was a problem getting a configuration file.

exception `wand.exceptions.ConfigureWarning`

Bases: `wand.exceptions.BaseWarning`

There was a problem getting a configuration file.

exception `wand.exceptions.CorruptImageError`

Bases: `wand.exceptions.BaseError`, `exceptions.ValueError`

The image file may be corrupt.

exception `wand.exceptions.CorruptImageFatalError`

Bases: `wand.exceptions.BaseFatalError`, `exceptions.ValueError`

The image file may be corrupt.

exception `wand.exceptions.CorruptImageWarning`

Bases: `wand.exceptions.BaseWarning`, `exceptions.ValueError`

The image file may be corrupt.

`wand.exceptions.DOMAIN_MAP = [('ResourceLimit', 'A program resource is exhausted e.g. not enough memory')]`
 (list) A list of error/warning domains, these descriptions and codes. The form of elements is like: (domain name, description, codes).

exception `wand.exceptions.DelegateError`
Bases: `wand.exceptions.BaseError`
An ImageMagick delegate failed to complete.

exception `wand.exceptions.DelegateFatalError`
Bases: `wand.exceptions.BaseFatalError`
An ImageMagick delegate failed to complete.

exception `wand.exceptions.DelegateWarning`
Bases: `wand.exceptions.BaseWarning`
An ImageMagick delegate failed to complete.

exception `wand.exceptions.DrawError`
Bases: `wand.exceptions.BaseError`
A drawing operation failed.

exception `wand.exceptions.DrawFatalError`
Bases: `wand.exceptions.BaseFatalError`
A drawing operation failed.

exception `wand.exceptions.DrawWarning`
Bases: `wand.exceptions.BaseWarning`
A drawing operation failed.

exception `wand.exceptions.FileOpenError`
Bases: `wand.exceptions.BaseError`, `exceptions.IOError`
The image file could not be opened for reading or writing.

exception `wand.exceptions.FileOpenFatalError`
Bases: `wand.exceptions.BaseFatalError`, `exceptions.IOError`
The image file could not be opened for reading or writing.

exception `wand.exceptions.FileOpenWarning`
Bases: `wand.exceptions.BaseWarning`, `exceptions.IOError`
The image file could not be opened for reading or writing.

exception `wand.exceptions.ImageError`
Bases: `wand.exceptions.BaseError`
The operation could not complete due to an incompatible image.

exception `wand.exceptions.ImageFatalError`
Bases: `wand.exceptions.BaseFatalError`
The operation could not complete due to an incompatible image.

exception `wand.exceptions.ImageWarning`
Bases: `wand.exceptions.BaseWarning`
The operation could not complete due to an incompatible image.

exception `wand.exceptions.MissingDelegateError`
Bases: `wand.exceptions.BaseError`, `exceptions.ImportError`
The image type can not be read or written because the appropriate; delegate is missing.

exception `wand.exceptions.MissingDelegateFatalError`

Bases: `wand.exceptions.BaseFatalError`, `exceptions.ImportError`

The image type can not be read or written because the appropriate; delegate is missing.

exception `wand.exceptions.MissingDelegateWarning`

Bases: `wand.exceptions.BaseWarning`, `exceptions.ImportError`

The image type can not be read or written because the appropriate; delegate is missing.

exception `wand.exceptions.ModuleError`

Bases: `wand.exceptions.BaseError`

There was a problem with an image module.

exception `wand.exceptions.ModuleFatalError`

Bases: `wand.exceptions.BaseFatalError`

There was a problem with an image module.

exception `wand.exceptions.ModuleWarning`

Bases: `wand.exceptions.BaseWarning`

There was a problem with an image module.

exception `wand.exceptions.MonitorError`

Bases: `wand.exceptions.BaseError`

There was a problem activating the progress monitor.

exception `wand.exceptions.MonitorFatalError`

Bases: `wand.exceptions.BaseFatalError`

There was a problem activating the progress monitor.

exception `wand.exceptions.MonitorWarning`

Bases: `wand.exceptions.BaseWarning`

There was a problem activating the progress monitor.

exception `wand.exceptions.OptionError`

Bases: `wand.exceptions.BaseError`

A command-line option was malformed.

exception `wand.exceptions.OptionFatalError`

Bases: `wand.exceptions.BaseFatalError`

A command-line option was malformed.

exception `wand.exceptions.OptionWarning`

Bases: `wand.exceptions.BaseWarning`

A command-line option was malformed.

exception `wand.exceptions.PolicyError`

Bases: `wand.exceptions.BaseError`

A policy denies access to a delegate, coder, filter, path, or resource.

exception `wand.exceptions.PolicyFatalError`

Bases: `wand.exceptions.BaseFatalError`

A policy denies access to a delegate, coder, filter, path, or resource.

exception `wand.exceptions.PolicyWarning`

Bases: `wand.exceptions.BaseWarning`

A policy denies access to a delegate, coder, filter, path, or resource.

exception `wand.exceptions.RandomError`

Bases: `wand.exceptions.BaseError`

There is a problem generating a true or pseudo-random number.

exception `wand.exceptions.RandomFatalError`

Bases: `wand.exceptions.BaseFatalError`

There is a problem generating a true or pseudo-random number.

exception `wand.exceptions.RandomWarning`

Bases: `wand.exceptions.BaseWarning`

There is a problem generating a true or pseudo-random number.

exception `wand.exceptions.RegistryError`

Bases: `wand.exceptions.BaseError`

There was a problem getting or setting the registry.

exception `wand.exceptions.RegistryFatalError`

Bases: `wand.exceptions.BaseFatalError`

There was a problem getting or setting the registry.

exception `wand.exceptions.RegistryWarning`

Bases: `wand.exceptions.BaseWarning`

There was a problem getting or setting the registry.

exception `wand.exceptions.ResourceLimitError`

Bases: `wand.exceptions.BaseError`, `exceptions.MemoryError`

A program resource is exhausted e.g. not enough memory.

exception `wand.exceptions.ResourceLimitFatalError`

Bases: `wand.exceptions.BaseFatalError`, `exceptions.MemoryError`

A program resource is exhausted e.g. not enough memory.

exception `wand.exceptions.ResourceLimitWarning`

Bases: `wand.exceptions.BaseWarning`, `exceptions.MemoryError`

A program resource is exhausted e.g. not enough memory.

exception `wand.exceptions.StreamError`

Bases: `wand.exceptions.BaseError`, `exceptions.IOError`

There was a problem reading or writing from a stream.

exception `wand.exceptions.StreamFatalError`

Bases: `wand.exceptions.BaseFatalError`, `exceptions.IOError`

There was a problem reading or writing from a stream.

exception `wand.exceptions.StreamWarning`

Bases: `wand.exceptions.BaseWarning`, `exceptions.IOError`

There was a problem reading or writing from a stream.

`wand.exceptions.TYPE_MAP = {300: <class 'wand.exceptions.ResourceLimitWarning'>, 305: <c
(dict) The dictionary of (code, exc_type).`

exception `wand.exceptions.TypeError`

Bases: `wand.exceptions.BaseError`

A font is unavailable; a substitution may have occurred.

exception `wand.exceptions.TypeFatalError`

Bases: `wand.exceptions.BaseFatalError`

A font is unavailable; a substitution may have occurred.

exception `wand.exceptions.TypeWarning`

Bases: `wand.exceptions.BaseWarning`

A font is unavailable; a substitution may have occurred.

exception `wand.exceptions.WandError`

Bases: `wand.exceptions.BaseError`

There was a problem specific to the MagickWand API.

exception `wand.exceptions.WandException`

Bases: `exceptions.Exception`

All Wand-related exceptions are derived from this class.

exception `wand.exceptions.WandFatalError`

Bases: `wand.exceptions.BaseFatalError`

There was a problem specific to the MagickWand API.

exception `wand.exceptions.WandLibraryVersionError`

Bases: `wand.exceptions.WandException`

Base class for Wand-related ImageMagick version errors.

New in version 0.3.2.

exception `wand.exceptions.WandRuntimeError`

Bases: `wand.exceptions.WandException`, `exceptions.RuntimeError`

Generic class for Wand-related runtime errors.

New in version 0.5.2.

exception `wand.exceptions.WandWarning`

Bases: `wand.exceptions.BaseWarning`

There was a problem specific to the MagickWand API.

exception `wand.exceptions.XServerError`

Bases: `wand.exceptions.BaseError`

An X resource is unavailable.

exception `wand.exceptions.XServerFatalError`

Bases: `wand.exceptions.BaseFatalError`

An X resource is unavailable.

exception `wand.exceptions.XServerWarning`

Bases: `wand.exceptions.BaseWarning`

An X resource is unavailable.

4.1.8 wand.api — Low-level interfaces

Changed in version 0.1.10: Changed to throw `ImportError` instead of `AttributeError` when the shared library fails to load.

class `wand.api.AffineMatrix`

class `wand.api MagickPixelPacket`

`wand.api.library`
(`ctypes.CDLL`) The MagickWand library.

`wand.api.libc`
(`ctypes.CDLL`) The C standard library.

`wand.api.libmagick`
(`ctypes.CDLL`) The ImageMagick library. It is the same with `library` on platforms other than Windows.
New in version 0.1.10.

`wand.api.load_library()`
Loads the MagickWand library.

Returns the MagickWand library and the ImageMagick library

Return type `ctypes.CDLL`

class `wand.api.PixelInfo`

class `wand.api.PointInfo`

4.1.9 wand.compat — Compatibility layer

This module provides several subtle things to support multiple Python versions (2.6, 2.7, 3.3+) and VM implementations (CPython, PyPy).

`wand.compat.PY3 = False`
(`bool`) Whether it is Python 3.x or not.

`wand.compat.abc = <module 'collections' from '/home/docs/.pyenv/versions/2.7.14/lib/python2.7/collections.py'>`
(`module`) Module containing abstract base classes. `collections` in Python 2 and `collections.abc` in Python 3.

`wand.compat.binary(string, var=None)`
Makes string to `str` in Python 2. Makes string to `bytes` in Python 3.

Parameters

- **string** (`bytes`, `str`, `unicode`) – a string to cast it to `binary_type`
- **var** (`str`) – an optional variable name to be used for error message

`wand.compat.binary_type`
alias of `__builtin__.str`

`wand.compat.encode_filename(filename)`
If `filename` is a `text_type`, encode it to `binary_type` according to filesystem's default encoding.

`wand.compat.file_types = (<class 'io.RawIOBase'>, <type 'file'>)`
(`type`, `tuple`) Types for file objects that have `fileno()`.

`wand.compat.nested(*args, **kws)`

Combine multiple context managers into a single nested context manager.

This function has been deprecated in favour of the multiple manager form of the with statement.

The one advantage of this function over the multiple manager form of the with statement is that argument unpacking allows it to be used with a variable number of context managers as follows:

```
with nested(*managers): do_something()
```

`wand.compat.string_type`

alias of `__builtin__.basestring`

`wand.compat.text(string)`

Makes `string` to `str` in Python 3. Does nothing in Python 2.

Parameters `string` (`bytes`, `str`, `unicode`) – a string to cast it to `text_type`

`wand.compat.text_type`

alias of `__builtin__.unicode`

class `wand.compat.xrangle(stop)` → xrange object

`xrange(start, stop[, step])` → xrange object

Like `range()`, but instead of returning a list, returns an object that generates the numbers in the range on demand. For looping, this is slightly faster than `range()` and more memory efficient.

4.1.10 wand.display — Displaying images

The `display()` functions shows you the image. It is useful for debugging.

If you are in Mac, the image will be opened by your default image application (**Preview.app** usually).

If you are in Windows, the image will be opened by **imdisplay.exe**, or your default image application (**Windows Photo Viewer** usually) if **imdisplay.exe** is unavailable.

You can use it from CLI also. Execute `wand.display` module through `python -m` option:

```
$ python -m wand.display wandtests/assets/mona-lisa.jpg
```

New in version 0.1.9.

`wand.display.display(image, server_name=':0')`

Displays the passed image.

Parameters

- **image** (`Image`) – an image to display
- **server_name** (`str`) – X11 server name to use. it is ignored and not used for Mac. default is `':0'`

4.1.11 wand.version — Version data

You can find the current version in the command line interface:

```
$ python -m wand.version
0.5.2
$ python -m wand.version --verbose
Wand 0.5.2
```

(continues on next page)

(continued from previous page)

```
ImageMagick 6.7.7-6 2012-06-03 Q16 http://www.imagemagick.org
$ python -m wand.version --config | grep CC | cut -d : -f 2
gcc -std=gnu99 -std=gnu99
$ python -m wand.version --fonts | grep Helvetica
Helvetica
Helvetica-Bold
Helvetica-Light
Helvetica-Narrow
Helvetica-Oblique
$ python -m wand.version --formats | grep CMYK
CMYK
CMYKA
```

New in version 0.2.0: The command line interface.

New in version 0.2.2: The `--verbose/-v` option which also prints ImageMagick library version for CLI.

New in version 0.4.1: The `--fonts`, `--formats`, & `--config` option allows printing additional information about ImageMagick library.

`wand.version.VERSION = '0.5.2'`
(basestring) The version string e.g. '0.1.2'.

Changed in version 0.1.9: Becomes string. (It was `tuple` before.)

`wand.version.VERSION_INFO = (0, 5, 2)`
(`tuple`) The version tuple e.g. (0, 1, 2).

Changed in version 0.1.9: Becomes `tuple`. (It was string before.)

`wand.version.MAGICK_VERSION = None`
(basestring) The version string of the linked ImageMagick library. The exactly same string to the result of `GetMagickVersion()` function.

Example:

```
'ImageMagick 6.7.7-6 2012-06-03 Q16 http://www.imagemagick.org'
```

New in version 0.2.1.

`wand.version.MAGICK_VERSION_FEATURES = 'Cipher DPC Modules OpenMP '`
(basestring) A string of all features enabled. This value is identical to what is returned by `GetMagickFeatures()`

New in version 0.5.0.

`wand.version.MAGICK_VERSION_INFO = None`
(`tuple`) The version tuple e.g. (6, 7, 7, 6) of `MAGICK_VERSION`.

New in version 0.2.1.

`wand.version.MAGICK_VERSION_NUMBER = None`
(`numbers.Integral`) The version number of the linked ImageMagick library.

New in version 0.2.1.

`wand.version.MAGICK_RELEASE_DATE = None`
(`datetime.date`) The release date of the linked ImageMagick library. Equivalent to the result of `GetMagickReleaseDate()` function.

New in version 0.2.1.

`wand.version.MAGICK_RELEASE_DATE_STRING = None`

(*basestring*) The date string e.g. '2012-06-03' of `MAGICK_RELEASE_DATE_STRING`. This value is the exactly same string to the result of `GetMagickReleaseDate()` function.

New in version 0.2.1.

`wand.version.MAGICK_HDRI = None`

(*bool*) True if ImageMagick is compiled for High Dynamic Range Image.

`wand.version.QUANTUM_DEPTH = None`

(*numbers.Integral*) The quantum depth configuration of the linked ImageMagick library. One of 8, 16, 32, or 64.

New in version 0.3.0.

`wand.version.QUANTUM_RANGE = None`

(*numbers.Integral*) The quantum range configuration of the linked ImageMagick library.

New in version 0.5.0.

`wand.version.configure_options(pattern='*')`

Queries ImageMagick library for configurations options given at compile-time.

Example: Find where the ImageMagick documents are installed:

```
>>> from wand.version import configure_options
>>> configure_options('DOC*')
{'DOCUMENTATION_PATH': '/usr/local/share/doc/ImageMagick-6'}
```

Parameters `pattern` (*basestring*) – A term to filter queries against. Supports wildcard '*' characters. Default patterns '*' for all options.

Returns Directory of configuration options matching given pattern

Return type `collections.defaultdict`

`wand.version.fonts(pattern='*')`

Queries ImageMagick library for available fonts.

Available fonts can be configured by defining `types.xml`, `type-ghostscript.xml`, or `type-windows.xml`. Use `wand.version.configure_options()` to locate system search path, and [resources](#) article for defining xml file.

Example: List all bold Helvetica fonts:

```
>>> from wand.version import fonts
>>> fonts('*Helvetica*Bold*')
['Helvetica-Bold', 'Helvetica-Bold-Oblique', 'Helvetica-BoldOblique',
 'Helvetica-Narrow-Bold', 'Helvetica-Narrow-BoldOblique']
```

Parameters `pattern` (*basestring*) – A term to filter queries against. Supports wildcard '*' characters. Default patterns '*' for all options.

Returns Sequence of matching fonts

Return type `collections.Sequence`

`wand.version.formats(pattern='*')`

Queries ImageMagick library for supported formats.

Example: List supported PNG formats:

```
>>> from wand.version import formats
>>> formats('PNG*')
['PNG', 'PNG00', 'PNG8', 'PNG24', 'PNG32', 'PNG48', 'PNG64']
```

Parameters **pattern** (basestring) – A term to filter formats against. Supports wildcards ‘*’ characters. Default pattern ‘*’ for all formats.

Returns Sequence of matching formats

Return type `collections.Sequence`

5.1 Mailing list

Wand has the list for users. If you want to subscribe the list, just send a mail to:

wand@librelist.com

The [list archive](#) provided by [Librelist](#) is synchronized every hour.

5.2 Stack Overflow

There's a Stack Overflow tag for Wand:

<http://stackoverflow.com/questions/tagged/wand>

Freely ask questions about Wand including troubleshooting. Thanks for [sindikar](#)'s contribution.

5.3 Documentation

The [documentation](#) for Wand is hosted by [ReadTheDocs.org](#). The nightly development docs can be found under the [latest](#) version, and the most recent release under [stable](#). Previous & maintenance releases are also available.

CHAPTER 6

Open source

Wand is an open source software initially written by [Hong Minhee](#) (for [StyleShare](#)), and is currently maintained by E. McConville. See also the complete list of [contributors](#) as well. The source code is distributed under [MIT license](#) and you can find it at [GitHub repository](#). Check out now:

```
$ git clone git://github.com/emcconville/wand.git
```

If you find a bug, please notify to [our issue tracker](#). Pull requests are always welcome!

We discuss about Wand's development on IRC. Come [#wand](#) channel on freenode network.

Check out [Wand Changelog](#) also.

CHAPTER 7

Indices and tables

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

W

- wand, [75](#)
- wand.api, [147](#)
- wand.color, [117](#)
- wand.compat, [148](#)
- wand.display, [149](#)
- wand.drawing, [121](#)
- wand.exceptions, [142](#)
- wand.font, [120](#)
- wand.image, [75](#)
- wand.resource, [139](#)
- wand.sequence, [138](#)
- wand.version, [149](#)

A

abc (*in module wand.compat*), 148
 affine() (*wand.drawing.Drawing method*), 124
 AffineMatrix (*class in wand.api*), 148
 allocate() (*wand.resource.Resource method*), 139
 alpha (*wand.color.Color attribute*), 118
 alpha() (*wand.drawing.Drawing method*), 124
 alpha_channel (*wand.image.BaseImage attribute*), 88
 ALPHA_CHANNEL_TYPES (*in module wand.image*), 75
 alpha_int8 (*wand.color.Color attribute*), 118
 alpha_quantum (*wand.color.Color attribute*), 118
 animation (*wand.image.BaseImage attribute*), 89
 antialias (*wand.font.Font attribute*), 121
 antialias (*wand.image.BaseImage attribute*), 89
 append() (*wand.sequence.Sequence method*), 138
 arc() (*wand.drawing.Drawing method*), 125
 artifacts (*wand.image.Image attribute*), 113
 ArtifactTree (*class in wand.image*), 116
 ascender (*wand.drawing.FontMetrics attribute*), 137
 auto_orient() (*wand.image.BaseImage method*), 89

B

background_color (*wand.image.BaseImage attribute*), 89
 BaseError, 142
 BaseFatalError, 142
 BaseImage (*class in wand.image*), 88
 BaseWarning, 142
 bezier() (*wand.drawing.Drawing method*), 125
 binary() (*in module wand.compat*), 148
 binary_type (*in module wand.compat*), 148
 black (*wand.color.Color attribute*), 118
 black_int8 (*wand.color.Color attribute*), 118
 black_quantum (*wand.color.Color attribute*), 118
 blank() (*wand.image.Image method*), 113
 BlobError, 142
 BlobFatalError, 142
 BlobWarning, 142

blue (*wand.color.Color attribute*), 118
 blue_int8 (*wand.color.Color attribute*), 118
 blue_primary (*wand.image.BaseImage attribute*), 89
 blue_quantum (*wand.color.Color attribute*), 118
 blur() (*wand.image.BaseImage method*), 89
 border() (*wand.image.BaseImage method*), 89
 border_color (*wand.drawing.Drawing attribute*), 125

C

c_clear_exception (*wand.resource.Resource attribute*), 140
 c_destroy_resource (*wand.resource.Resource attribute*), 140
 c_equals() (*wand.color.Color static method*), 118
 c_get_exception (*wand.resource.Resource attribute*), 140
 c_is_resource (*wand.resource.Resource attribute*), 140
 CacheError, 142
 CacheFatalError, 143
 CacheWarning, 143
 caption() (*wand.image.BaseImage method*), 90
 channel_depths (*wand.image.Image attribute*), 113
 channel_images (*wand.image.Image attribute*), 113
 ChannelDepthDict (*class in wand.image*), 111
 ChannelImageDict (*class in wand.image*), 111
 CHANNELS (*in module wand.image*), 76
 character_height (*wand.drawing.FontMetrics attribute*), 137
 character_width (*wand.drawing.FontMetrics attribute*), 137
 circle() (*wand.drawing.Drawing method*), 125
 clamp() (*wand.image.BaseImage method*), 90
 clear() (*wand.image.Image method*), 113
 clip_path (*wand.drawing.Drawing attribute*), 125
 CLIP_PATH_UNITS (*in module wand.drawing*), 121
 clip_rule (*wand.drawing.Drawing attribute*), 126
 clip_units (*wand.drawing.Drawing attribute*), 126
 clone() (*wand.drawing.Drawing method*), 126

`clone()` (*wand.image.BaseImage* method), 90
`clone()` (*wand.image.Iterator* method), 116
`close()` (*wand.image.Image* method), 113
`ClosedImageError`, 111
`clut()` (*wand.image.BaseImage* method), 90
`coalesce()` (*wand.image.BaseImage* method), 91
`CODE_MAP` (in module *wand.exceptions*), 142
`CoderError`, 143
`CoderFatalError`, 143
`CoderWarning`, 143
`Color` (class in *wand.color*), 117
`color` (*wand.font.Font* attribute), 121
`color()` (*wand.drawing.Drawing* method), 126
`colorspace` (*wand.image.BaseImage* attribute), 91
`COLORSPACE_TYPES` (in module *wand.image*), 76
`comment()` (*wand.drawing.Drawing* method), 126
`compare()` (*wand.image.BaseImage* method), 91
`compare_layers()` (*wand.image.Image* method), 113
`COMPARE_METRICS` (in module *wand.image*), 77
`compose` (*wand.image.BaseImage* attribute), 91
`composite()` (*wand.drawing.Drawing* method), 126
`composite()` (*wand.image.BaseImage* method), 91
`composite_channel()` (*wand.image.BaseImage* method), 91
`COMPOSITE_OPERATORS` (in module *wand.image*), 78
`compression` (*wand.image.BaseImage* attribute), 92
`compression_quality` (*wand.image.BaseImage* attribute), 92
`COMPRESSION_TYPES` (in module *wand.image*), 80
`concat()` (*wand.image.BaseImage* method), 92
`configure_options()` (in module *wand.version*), 151
`ConfigureError`, 143
`ConfigureFatalError`, 143
`ConfigureWarning`, 143
`container` (*wand.sequence.SingleImage* attribute), 138
`contrast_stretch()` (*wand.image.BaseImage* method), 92
`convert()` (*wand.image.Image* method), 114
`CorruptImageError`, 143
`CorruptImageFatalError`, 143
`CorruptImageWarning`, 143
`crop()` (*wand.image.BaseImage* method), 92
`current_index` (*wand.sequence.Sequence* attribute), 138
`cyan` (*wand.color.Color* attribute), 118
`cyan_int8` (*wand.color.Color* attribute), 119
`cyan_quantum` (*wand.color.Color* attribute), 119

D

`deconstruct()` (*wand.image.BaseImage* method), 93

`decrement_refcount()` (in module *wand.resource*), 139
`delay` (*wand.sequence.SingleImage* attribute), 138
`DelegateError`, 143
`DelegateFatalError`, 144
`DelegateWarning`, 144
`depth` (*wand.image.BaseImage* attribute), 93
`descender` (*wand.drawing.FontMetrics* attribute), 137
`deskew()` (*wand.image.BaseImage* method), 94
`despeckle()` (*wand.image.BaseImage* method), 94
`destroy()` (*wand.image.Image* method), 114
`destroy()` (*wand.resource.Resource* method), 140
`DestroyedResourceError`, 142
`dirty` (*wand.color.Color* attribute), 119
`dirty` (*wand.image.BaseImage* attribute), 94
`display()` (in module *wand.display*), 149
`dispose` (*wand.image.BaseImage* attribute), 94
`DISPOSE_TYPES` (in module *wand.image*), 80
`distort()` (*wand.image.BaseImage* method), 94
`DISTORTION_METHODS` (in module *wand.image*), 80
`DITHER_METHODS` (in module *wand.image*), 81
`DOMAIN_MAP` (in module *wand.exceptions*), 143
`draw()` (*wand.drawing.Drawing* method), 127
`DrawError`, 144
`DrawFatalError`, 144
`Drawing` (class in *wand.drawing*), 124
`DrawWarning`, 144

E

`edge()` (*wand.image.BaseImage* method), 94
`ellipse()` (*wand.drawing.Drawing* method), 127
`emboss()` (*wand.image.BaseImage* method), 94
`encode_filename()` (in module *wand.compat*), 148
`enhance()` (*wand.image.BaseImage* method), 94
environment variable
 MAGICK_HOME, 11–13
`equalize()` (*wand.image.BaseImage* method), 95
`evaluate()` (*wand.image.BaseImage* method), 95
`EVALUATE_OPS` (in module *wand.image*), 81
`export_pixels()` (*wand.image.BaseImage* method), 95
`extend()` (*wand.sequence.Sequence* method), 138
`extent()` (*wand.image.BaseImage* method), 96

F

`file_types` (in module *wand.compat*), 148
`FileOpenError`, 144
`FileOpenFatalError`, 144
`FileOpenWarning`, 144
`fill_color` (*wand.drawing.Drawing* attribute), 127
`fill_opacity` (*wand.drawing.Drawing* attribute), 127
`fill_rule` (*wand.drawing.Drawing* attribute), 127
`FILL_RULE_TYPES` (in module *wand.drawing*), 122

`FILTER_TYPES` (in module `wand.image`), 82
`flip()` (`wand.image.BaseImage` method), 96
`flop()` (`wand.image.BaseImage` method), 96
`Font` (class in `wand.font`), 121
`font` (`wand.drawing.Drawing` attribute), 127
`font` (`wand.image.BaseImage` attribute), 96
`font_antialias` (`wand.image.BaseImage` attribute), 96
`font_family` (`wand.drawing.Drawing` attribute), 127
`FONT_METRICS_ATTRIBUTES` (in module `wand.drawing`), 122
`font_path` (`wand.image.BaseImage` attribute), 96
`font_resolution` (`wand.drawing.Drawing` attribute), 127
`font_size` (`wand.drawing.Drawing` attribute), 127
`font_size` (`wand.image.BaseImage` attribute), 96
`font_stretch` (`wand.drawing.Drawing` attribute), 127
`font_style` (`wand.drawing.Drawing` attribute), 128
`font_weight` (`wand.drawing.Drawing` attribute), 128
`FontMetrics` (class in `wand.drawing`), 137
`fonts()` (in module `wand.version`), 151
`format` (`wand.image.BaseImage` attribute), 97
`formats()` (in module `wand.version`), 151
`frame()` (`wand.image.BaseImage` method), 97
`from_hsl()` (`wand.color.Color` class method), 119
`function()` (`wand.image.BaseImage` method), 97
`FUNCTION_TYPES` (in module `wand.image`), 83
`fx()` (`wand.image.BaseImage` method), 98

G

`gamma()` (`wand.image.BaseImage` method), 98
`gaussian_blur()` (`wand.image.BaseImage` method), 98
`genesis()` (in module `wand.resource`), 139
`get_exception()` (`wand.resource.Resource` method), 140
`get_font_metrics()` (`wand.drawing.Drawing` method), 128
`get_resource_limit()` (`wand.resource.ResourceLimits` method), 141
`gravity` (`wand.drawing.Drawing` attribute), 128
`gravity` (`wand.image.BaseImage` attribute), 98
`GRAVITY_TYPES` (in module `wand.drawing`), 122
`GRAVITY_TYPES` (in module `wand.image`), 83
`green` (`wand.color.Color` attribute), 119
`green_int8` (`wand.color.Color` attribute), 119
`green_primary` (`wand.image.BaseImage` attribute), 98
`green_quantum` (`wand.color.Color` attribute), 119

H

`hald_clut()` (`wand.image.BaseImage` method), 99

`height` (`wand.image.BaseImage` attribute), 99
`histogram` (`wand.image.BaseImage` attribute), 99
`HistogramDict` (class in `wand.image`), 111
`hsl()` (`wand.color.Color` method), 119

I

`Image` (class in `wand.image`), 111
`image` (`wand.image.ImageProperty` attribute), 115
`IMAGE_LAYER_METHOD` (in module `wand.image`), 84
`IMAGE_TYPES` (in module `wand.image`), 84
`ImageError`, 144
`ImageFatalError`, 144
`ImageProperty` (class in `wand.image`), 115
`ImageWarning`, 144
`implode()` (`wand.image.BaseImage` method), 99
`import_pixels()` (`wand.image.BaseImage` method), 99
`increment_refcount()` (in module `wand.resource`), 139
`index` (`wand.sequence.SingleImage` attribute), 138
`index_context()` (`wand.sequence.Sequence` method), 138
`insert()` (`wand.sequence.Sequence` method), 138
`interlace_scheme` (`wand.image.BaseImage` attribute), 100
`INTERLACE_TYPES` (in module `wand.image`), 85
`interpolate_method` (`wand.image.BaseImage` attribute), 100
`Iterator` (class in `wand.image`), 115

K

`KERNEL_INFO_TYPES` (in module `wand.image`), 85

L

`level()` (`wand.image.BaseImage` method), 100
`libc` (in module `wand.api`), 148
`libmagick` (in module `wand.api`), 148
`library` (in module `wand.api`), 148
`limits` (in module `wand.resource`), 139
`line()` (`wand.drawing.Drawing` method), 128
`LINE_CAP_TYPES` (in module `wand.drawing`), 122
`LINE_JOIN_TYPES` (in module `wand.drawing`), 122
`linear_stretch()` (`wand.image.BaseImage` method), 101
`liquid_rescale()` (`wand.image.BaseImage` method), 101
`load_library()` (in module `wand.api`), 148
`loop` (`wand.image.BaseImage` attribute), 101

M

`magenta` (`wand.color.Color` attribute), 119
`magenta_int8` (`wand.color.Color` attribute), 119
`MAGICK_HDRI` (in module `wand.version`), 151

MAGICK_HOME, 11–13
MAGICK_RELEASE_DATE (in module *wand.version*), 150
MAGICK_RELEASE_DATE_STRING (in module *wand.version*), 150
MAGICK_VERSION (in module *wand.version*), 150
MAGICK_VERSION_FEATURES (in module *wand.version*), 150
MAGICK_VERSION_INFO (in module *wand.version*), 150
MAGICK_VERSION_NUMBER (in module *wand.version*), 150
MagickPixelPacket (class in *wand.api*), 148
make_blob() (*wand.image.Image* method), 114
manipulative() (in module *wand.image*), 116
matte() (*wand.drawing.Drawing* method), 128
matte_color (*wand.image.BaseImage* attribute), 101
maximum_horizontal_advance (*wand.drawing.FontMetrics* attribute), 137
merge_layers() (*wand.image.BaseImage* method), 101
Metadata (class in *wand.image*), 116
metadata (*wand.image.Image* attribute), 114
mimetype (*wand.image.Image* attribute), 114
MissingDelegateError, 144
MissingDelegateFatalError, 144
MissingDelegateWarning, 145
modulate() (*wand.image.BaseImage* method), 102
ModuleError, 145
ModuleFatalError, 145
ModuleWarning, 145
MonitorError, 145
MonitorFatalError, 145
MonitorWarning, 145
morphology() (*wand.image.BaseImage* method), 102
MORPHOLOGY_METHODS (in module *wand.image*), 86

N

negate() (*wand.image.BaseImage* method), 103
nested() (in module *wand.compat*), 148
normalize() (*wand.image.BaseImage* method), 103
normalized_string (*wand.color.Color* attribute), 120

O

opacity (*wand.drawing.Drawing* attribute), 128
optimize_layers() (*wand.image.BaseImage* method), 103
optimize_transparency() (*wand.image.BaseImage* method), 103
OptionDict (class in *wand.image*), 116
OptionError, 145
OptionFatalError, 145
options (*wand.image.BaseImage* attribute), 103

OptionWarning, 145
orientation (*wand.image.BaseImage* attribute), 103
ORIENTATION_TYPES (in module *wand.image*), 87

P

page (*wand.image.BaseImage* attribute), 103
page_height (*wand.image.BaseImage* attribute), 103
page_width (*wand.image.BaseImage* attribute), 104
page_x (*wand.image.BaseImage* attribute), 104
page_y (*wand.image.BaseImage* attribute), 104
PAINT_METHOD_TYPES (in module *wand.drawing*), 122
path (*wand.font.Font* attribute), 121
path_close() (*wand.drawing.Drawing* method), 129
path_curve() (*wand.drawing.Drawing* method), 129
path_curve_to_quadratic_bezier() (*wand.drawing.Drawing* method), 129
path_elliptic_arc() (*wand.drawing.Drawing* method), 129
path_finish() (*wand.drawing.Drawing* method), 130
path_horizontal_line() (*wand.drawing.Drawing* method), 130
path_line() (*wand.drawing.Drawing* method), 130
path_move() (*wand.drawing.Drawing* method), 130
path_start() (*wand.drawing.Drawing* method), 130
path_vertical_line() (*wand.drawing.Drawing* method), 131
PIXEL_INTERPOLATE_METHODS (in module *wand.image*), 87
PixelInfo (class in *wand.api*), 148
point() (*wand.drawing.Drawing* method), 131
PointInfo (class in *wand.api*), 148
PolicyError, 145
PolicyFatalError, 145
PolicyWarning, 145
polygon() (*wand.drawing.Drawing* method), 131
polyline() (*wand.drawing.Drawing* method), 131
pop() (*wand.drawing.Drawing* method), 131
pop_clip_path() (*wand.drawing.Drawing* method), 132
pop_defs() (*wand.drawing.Drawing* method), 132
pop_pattern() (*wand.drawing.Drawing* method), 132
posterize() (*wand.image.BaseImage* method), 104
ProfileDict (class in *wand.image*), 116
profiles (*wand.image.Image* attribute), 114
pseudo() (*wand.image.Image* method), 114
push() (*wand.drawing.Drawing* method), 132
push_clip_path() (*wand.drawing.Drawing* method), 133
push_defs() (*wand.drawing.Drawing* method), 133
push_pattern() (*wand.drawing.Drawing* method), 133

PY3 (in module *wand.compat*), 148

Q

quantize() (*wand.image.BaseImage* method), 104
QUANTUM_DEPTH (in module *wand.version*), 151
QUANTUM_RANGE (in module *wand.version*), 151
quantum_range (*wand.image.BaseImage* attribute), 104

R

raise_exception() (*wand.resource.Resource* method), 140
RandomError, 146
RandomFatalError, 146
RandomWarning, 146
read() (*wand.image.Image* method), 115
rectangle() (*wand.drawing.Drawing* method), 133
red (*wand.color.Color* attribute), 120
red_int8 (*wand.color.Color* attribute), 120
red_primary (*wand.image.BaseImage* attribute), 105
red_quantum (*wand.color.Color* attribute), 120
RegistryError, 146
RegistryFatalError, 146
RegistryWarning, 146
resample() (*wand.image.BaseImage* method), 105
reset_coords() (*wand.image.BaseImage* method), 105
resize() (*wand.image.BaseImage* method), 105
resolution (*wand.image.BaseImage* attribute), 105
Resource (class in *wand.resource*), 139
resource (*wand.resource.Resource* attribute), 140
resource() (*wand.resource.ResourceLimits* method), 141
ResourceLimitError, 146
ResourceLimitFatalError, 146
ResourceLimits (class in *wand.resource*), 140
ResourceLimitWarning, 146
rotate() (*wand.drawing.Drawing* method), 134
rotate() (*wand.image.BaseImage* method), 105

S

sample() (*wand.image.BaseImage* method), 106
save() (*wand.image.Image* method), 115
scale() (*wand.drawing.Drawing* method), 134
scale_quantum_to_int8() (in module *wand.color*), 120
Sequence (class in *wand.sequence*), 138
sequence (*wand.image.BaseImage* attribute), 106
set_fill_pattern_url() (*wand.drawing.Drawing* method), 134
set_resource_limit() (*wand.resource.ResourceLimits* method), 141

set_stroke_pattern_url() (*wand.drawing.Drawing* method), 135
shade() (*wand.image.BaseImage* method), 106
shadow() (*wand.image.BaseImage* method), 106
sharpen() (*wand.image.BaseImage* method), 106
shave() (*wand.image.BaseImage* method), 107
signature (*wand.image.BaseImage* attribute), 107
SingleImage (class in *wand.sequence*), 138
size (*wand.font.Font* attribute), 121
size (*wand.image.BaseImage* attribute), 107
skew() (*wand.drawing.Drawing* method), 135
STORAGE_TYPES (in module *wand.image*), 87
StreamError, 146
StreamFatalError, 146
StreamWarning, 146
STRETCH_TYPES (in module *wand.drawing*), 123
string (*wand.color.Color* attribute), 120
string_type (in module *wand.compat*), 149
strip() (*wand.image.BaseImage* method), 107
stroke_antialias (*wand.drawing.Drawing* attribute), 135
stroke_color (*wand.drawing.Drawing* attribute), 135
stroke_color (*wand.font.Font* attribute), 121
stroke_dash_array (*wand.drawing.Drawing* attribute), 135
stroke_dash_offset (*wand.drawing.Drawing* attribute), 135
stroke_line_cap (*wand.drawing.Drawing* attribute), 135
stroke_line_join (*wand.drawing.Drawing* attribute), 135
stroke_miter_limit (*wand.drawing.Drawing* attribute), 135
stroke_opacity (*wand.drawing.Drawing* attribute), 135
stroke_width (*wand.drawing.Drawing* attribute), 136
stroke_width (*wand.font.Font* attribute), 121
STYLE_TYPES (in module *wand.drawing*), 123

T

terminus() (in module *wand.resource*), 139
text() (in module *wand.compat*), 149
text() (*wand.drawing.Drawing* method), 136
TEXT_ALIGN_TYPES (in module *wand.drawing*), 123
text_alignment (*wand.drawing.Drawing* attribute), 136
text_antialias (*wand.drawing.Drawing* attribute), 136
text_decoration (*wand.drawing.Drawing* attribute), 136
TEXT_DECORATION_TYPES (in module *wand.drawing*), 123

`text_direction` (*wand.drawing.Drawing* attribute), 136
`TEXT_DIRECTION_TYPES` (*in module wand.drawing*), 124
`text_encoding` (*wand.drawing.Drawing* attribute), 136
`text_height` (*wand.drawing.FontMetrics* attribute), 137
`text_interline_spacing`
 (*wand.drawing.Drawing* attribute), 136
`text_interword_spacing`
 (*wand.drawing.Drawing* attribute), 136
`text_kerning` (*wand.drawing.Drawing* attribute), 136
`text_type` (*in module wand.compat*), 149
`text_under_color` (*wand.drawing.Drawing* attribute), 136
`text_width` (*wand.drawing.FontMetrics* attribute), 137
`threshold()` (*wand.image.BaseImage* method), 107
`transform()` (*wand.image.BaseImage* method), 107
`transform_colorspace()`
 (*wand.image.BaseImage* method), 108
`translate()` (*wand.drawing.Drawing* method), 136
`transparent_color()` (*wand.image.BaseImage* method), 108
`transparentize()` (*wand.image.BaseImage* method), 109
`transpose()` (*wand.image.BaseImage* method), 109
`transverse()` (*wand.image.BaseImage* method), 109
`trim()` (*wand.image.BaseImage* method), 109
`type` (*wand.image.BaseImage* attribute), 109
`TYPE_MAP` (*in module wand.exceptions*), 146
`TypeError`, 146
`TypeFatalError`, 147
`TypeWarning`, 147

U

`unique_colors()` (*wand.image.BaseImage* method), 109
`UNIT_TYPES` (*in module wand.image*), 88
`units` (*wand.image.BaseImage* attribute), 110
`unsharp_mask()` (*wand.image.BaseImage* method), 110

V

`vector_graphics` (*wand.drawing.Drawing* attribute), 136
`VERSION` (*in module wand.version*), 150
`VERSION_INFO` (*in module wand.version*), 150
`viewbox()` (*wand.drawing.Drawing* method), 137
`vignette()` (*wand.image.BaseImage* method), 110
`virtual_pixel` (*wand.image.BaseImage* attribute), 110

`VIRTUAL_PIXEL_METHOD` (*in module wand.image*), 88

W

`wand` (*module*), 75
`wand` (*wand.image.BaseImage* attribute), 110
`wand.api` (*module*), 147
`wand.color` (*module*), 117
`wand.compat` (*module*), 148
`wand.display` (*module*), 149
`wand.drawing` (*module*), 121
`wand.exceptions` (*module*), 142
`wand.font` (*module*), 120
`wand.image` (*module*), 75
`wand.resource` (*module*), 139
`wand.sequence` (*module*), 138
`wand.version` (*module*), 149
`WandError`, 147
`WandException`, 147
`WandFatalError`, 147
`WandLibraryVersionError`, 147
`WandRuntimeError`, 147
`WandWarning`, 147
`watermark()` (*wand.image.BaseImage* method), 110
`wave()` (*wand.image.BaseImage* method), 111
`white_point` (*wand.image.BaseImage* attribute), 111
`white_threshold()` (*wand.image.BaseImage* method), 111
`width` (*wand.image.BaseImage* attribute), 111

X

`x` (*wand.drawing.FontMetrics* attribute), 137
`x1` (*wand.drawing.FontMetrics* attribute), 137
`x2` (*wand.drawing.FontMetrics* attribute), 137
`xrange` (*class in wand.compat*), 149
`XServerError`, 147
`XServerFatalError`, 147
`XServerWarning`, 147

Y

`y` (*wand.drawing.FontMetrics* attribute), 137
`y1` (*wand.drawing.FontMetrics* attribute), 137
`y2` (*wand.drawing.FontMetrics* attribute), 137
`yellow` (*wand.color.Color* attribute), 120
`yellow_int8` (*wand.color.Color* attribute), 120
`yellow_quantum` (*wand.color.Color* attribute), 120