Wand Documentation

Release 0.3.9

Hong Minhee

1	Why just another binding?	3			
2	Requirements	5			
3	User's guide				
	3.1 What's new in Wand 0.3?	7			
	3.2 Installation	11			
	3.3 Reading images	14			
	3.4 Writing images	17			
	3.5 Resizing and cropping	18			
	3.6 Transformation	22			
	3.7 Drawing	26			
	3.8 Colorspace	28			
	3.9 Reading EXIF	28			
	3.10 Sequence	29			
	3.11 Resource management	30			
	3.12 Running tests	30			
	3.13 Roadmap	31			
	3.14 Wand Changelog	32			
	3.15 Talks and Presentations	39			
4	References	41			
	4.1 wand — Simple MagickWand API binding for Python	41			
5	Troubleshooting				
	5.1 Mailing list	79			
	5.2 Stack Overflow	79			
	5.3 Quora	79			
6	Open source	81			
7	Indices and tables				
Py	ython Module Index				

Wand is a ctypes-based simple ImageMagick binding for Python.

```
from wand.image import Image
from wand.display import display

with Image(filename='mona-lisa.png') as img:
    print(img.size)
    for r in 1, 2, 3:
        with img.clone() as i:
            i.resize(int(i.width * r * 0.25), int(i.height * r * 0.25))
            i.rotate(90 * r)
            i.save(filename='mona-lisa-{0}.png'.format(r))
            display(i)
```

You can install it from PyPI (and it requires MagickWand library):

```
$ apt-get install libmagickwand-dev
$ pip install Wand
```

Contents 1

2 Contents

Why just another binding?

There are already many MagickWand API bindings for Python, however they are lacking something we need:

- Pythonic and modern interfaces
- Good documentation
- Binding through ctypes (not C API) we are ready to go PyPy!
- Installation using pip or easy_install

Requirements

- Python 2.6 or higher
 - CPython 2.6 or higher
 - CPython 3.2 or higher
 - PyPy 1.5 or higher
- MagickWand library
 - libmagickwand-dev for APT on Debian/Ubuntu
 - imagemagick for MacPorts/Homebrew on Mac
 - ${\tt ImageMagick-devel}$ for Yum on ${\tt CentOS}$

User's guide

3.1 What's new in Wand 0.3?

This guide doesn't cover all changes in 0.3. See also the full list of changes in Version 0.3.0.

3.1.1 Python 3 support

Wand finally becomes to support Python 3, the future of Python. It actually doesn't cover all Python 3 versions, but the most two recent versions, 3.2 and 3.3, are supported. We still support Python 2.6, 2.7, and PyPy as well, so there's no dropped compatibility.

See also:

Wand now works on Python 3.2 and 3.3 The announcement about this on the mailing list.

3.1.2 Sequence

Wand now adds supports to sequential images like animated <code>image/gif</code> images and <code>image/ico</code> images that contains multiple icons. To distinguish between each single image and the container image, newly introduced class <code>SingleImage</code> has been added. The most of operations and properties are commonly available for both types, <code>Image</code> and <code>SingleImage</code>, and these are defined by their common superclass, <code>BaseImage</code>.

So every Image object now has sequence attribute which is list-like. It implements collections. Mutable Sequence protocol. That means you can pass it into for statement, get an item by index from it, slice it, call len() for it, or del an item of it by index. Every item inside it is a Single Image instance.

The following example shows you how to determine the largest icon in a image/ico file:

```
>>> from wand.image import Image
>>> import urllib2
>>> with Image(file=urllib2.urlopen('https://github.com/favicon.ico')) as ico:
... max(ico.sequence, key=lambda i: i.width * i.height)
...
<wand.sequence.SingleImage: 80d158d (32x32)>
```

This feature was initially proposed by Michael Elovskikh (#34), and then he also did initial work on this (#39). Andrey Antukh then improved its API (#66). Bear Dong and Taeho Kim did additional efforts for issues related to animated <code>image/gif</code> images (#88, #103, #112).

See also the guide for sequence as well: Sequence.

3.1.3 Drawing

Wand 0.3 provides basic facilities to draw *Lines* or *Texts*.

The following example code writes "Wand" to the transparent background using caption () method:

```
>>> from wand.font import Font
>>> font = Font(path='tests/assets/League_Gothic.otf', size=64)
>>> with Image(width=300, height=150) as image:
... image.caption('Wand', left=5, top=5, width=490, height=140, font=font)
... image.save(filename='caption-result.png')
```

Wand

Adrian Jung and did the most of work for this (#64). Cha, Hojeong added higher-level APIs on this and more text drawing APIs (#69, #71, #74).

3.1.4 **EXIF**

Wand now can read EXIF metadata from images through metadata property which is a mapping:

```
>>> from __future__ import print_function
>>> url = 'http://farm9.staticflickr.com/8282/7874109806_3fe0080ae4_o_d.jpg'
>>> with Image(file=urllib2.urlopen(url)) as i:
       for key, value in i.metadata.items():
           if key.startswith('exif:'):
                print(key, value)
exif:ApertureValue 8/1
exif:CustomRendered 0
exif:DateTime 2012:08:27 18:42:15
exif:DateTimeDigitized 2012:08:17 02:33:36
exif:DateTimeOriginal 2012:08:17 02:33:36
exif:ExifOffset 204
exif:ExifVersion 48, 50, 50, 49
exif:ExposureBiasValue 0/1
exif:ExposureMode 1
exif:ExposureProgram 1
exif:ExposureTime 1/50
```

Thanks for Michael Elovskikh who worked on this (#25, #56).

See also the guide for this as well: Reading EXIF.

3.1.5 Seam carving

ImageMagick optionally provides seam carving (also known as liquid rescaling or content-aware resizing) through MagickLiquidRescaleImage() function if it's properly configured --with-lqr. It makes you able to magically resize images without distortion.

Wand 0.3 becomes to provide a simple method Image.liquid_rescale() which binds this API.

You can find more detail examples in its guide: Seam carving (also known as content-aware resizing).

3.1.6 Channels

Some channel-related APIs like wand.image.Image.channel_images, channel_depths, and composite_channel() are added in Wand 0.3.

The following example makes the overlayed image (second, composite-channel-result.jpg) from the original image (first, composite-channel.jpg):





```
import shutil
import urllib2
from wand.image import Image
from wand.color import Color
url = 'http://farm6.staticflickr.com/5271/5836279075_c3f8226bc1_z.jpg'
with open('composite-channel.jpg', 'wb') as f:
   u = urllib2.urlopen(url)
    shutil.copyfileobj(u, f)
    u.close()
with Image(filename='composite-channel.jpg') as image:
    with Image(background=Color('black'),
               width=image.width,
               height=image.height / 3) as bar:
        image.composite_channel(
            channel='all_channels',
            image=bar,
            operator='overlay',
            left=0,
            top=(image.height- bar.height) / 2
    image.save(filename='composite-channel-result.jpg')
```

Note: The image composite-channel.jpg used in the above example is taken by Ejja Pahlevi and licensed under CC-BY-2.0. It can be found the original photography from Flickr.

3.1.7 Histogram

Every image now has histogram attribute, which is dictionary-like. Its keys are colors that used once or more in the image, and values are are the numbers of the pixels.

For example, simply get keys () of histogram if you need its palette.

```
>>> url = 'http://farm7.staticflickr.com/6145/5982384872_cb1e01004e_n.jpg'
>>> with Image(file=urllib2.urlopen(url)) as image:
... palette = image.histogram.keys()
```

3.2 Installation

Wand itself can be installed from PyPI using **easy_install** or **pip**:

```
$ easy_install Wand # or
$ pip install Wand
```

Wand is a Python binding of ImageMagick, so you have to install it as well:

- Debian/Ubuntu
- Fedora/CentOS
- Mac
- Windows

Or you can simply install Wand and its entire dependencies using the package manager of your system (it's way convenient but the version might be outdated):

- Debian/Ubuntu
- FreeBSD

3.2.1 Install ImageMagick on Debian/Ubuntu

If you're using Linux distributions based on Debian like Ubuntu, it can be easily installed using APT:

```
$ sudo apt-get install libmagickwand-dev
```

If you need SVG, WMF, OpenEXR, DjVu, and Graphviz support you have to install libmagickcore5-extra as well:

```
$ sudo apt-get install libmagickcore5-extra
```

3.2.2 Install ImageMagick on Fedora/CentOS

If you're using Linux distributions based on Redhat like Fedora or CentOS, it can be installed using Yum:

```
$ yum update
$ yum install ImageMagick-devel
```

3.2. Installation 11

3.2.3 Install ImageMagick on Mac

You need one of Homebrew or MacPorts to install ImageMagick.

Homebrew

```
$ brew install imagemagick
```

MacPorts

```
$ sudo port install imagemagick
```

If your Python in not installed using MacPorts, you have to export MAGICK_HOME path as well. Because Python that is not installed using MacPorts doesn't look up /opt/local, the default path prefix of MacPorts packages.

```
$ export MAGICK_HOME=/opt/local
```

3.2.4 Install ImageMagick on Windows

You could build ImageMagick by yourself, but it requires a build tool chain like Visual Studio to compile it. The easiest way is simply downloading a prebuilt binary of ImageMagick for your architecture (win32 or win64).

You can download it from the following link:

http://www.imagemagick.org/download/binaries/

Choose a binary for your architecture:

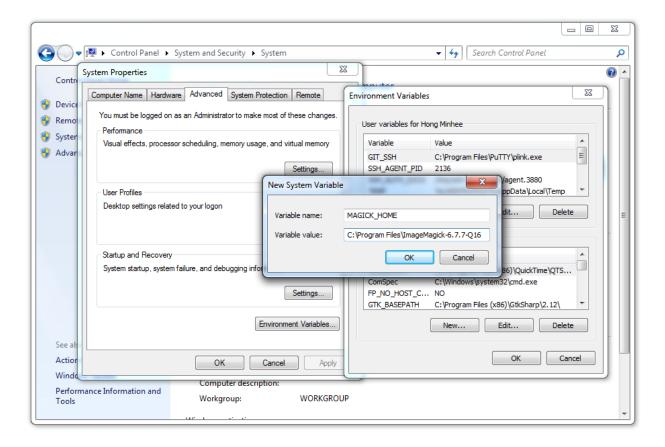
Windows 32-bit ImageMagick-6.7.7-6-Q16-windows-dll.exe

Windows 64-bit ImageMagick-6.7.7-6-Q16-windows-x64-dll.exe



Note that you have to check *Install development headers and libraries for C and C++* to make Wand able to link to it.

3.2. Installation



Lastly you have to set MAGICK_HOME environment variable to the path of ImageMagick (e.g. C:\Program Files\ImageMagick-6.7.7-Q16). You can set it in $Computer \rightarrow Properties \rightarrow Advanced$ system settings $\rightarrow Advanced \rightarrow Environment \ Variables...$

3.2.5 Install Wand on Debian/Ubuntu

Wand itself is already packaged in Debian/Ubuntu APT repository: python-wand. You can install it using **apt-get** command:

\$ sudo apt-get install python-wand

3.2.6 Install Wand on FreeBSD

Wand itself is already packaged in FreeBSD ports collection: py-wand. You can install it using pkg_add command:

\$ pkg_add -r py-wand

3.3 Reading images

There are several ways to open images:

- To open an image file
- To read a input stream (file-like object) that provides an image binary

- To read a binary string that contains image
- To copy an existing image object
- To open an empty image

All of these operations are provided by the constructor of Image class.

3.3.1 Open an image file

The most frequently used way is just to open an image by its filename. Image's constructor can take the parameter named filename:

```
from __future__ import print_function
from wand.image import Image

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

Note: It must be passed by keyword argument exactly. Because the constructor has many parameters that are exclusive to each other.

There is a keyword argument named file as well, but don't confuse it with filename. While filename takes a string of a filename, file takes a input stream (file-like object).

3.3.2 Read a input stream

If an image to open cannot be located by a filename but can be read through input stream interface (e.g. opened by os.popen(), contained in StringIO, read by urllib2.urlopen()), it can be read by Image constructor's file parameter. It takes all file-like objects which implements read() method:

```
from __future__ import print_function
from urllib2 import urlopen
from wand.image import Image

response = urlopen('https://stylesha.re/minhee/29998/images/100x100')
try:
    with Image(file=response) as img:
        print('format =', img.format)
        print('size =', img.size)

finally:
    response.close()
```

In the above example code, response object returned by urlopen() function has read() method, so it also can be used as an input stream for a downloaded image.

3.3.3 Read a blob

If you have just a binary string (str) of the image, you can pass it into Image constructor's blob parameter to read:

```
from __future__ import print_function
from wand.image import Image
with open('pikachu.png') as f:
```

```
image_binary = f.read()
with Image(blob=image_binary) as img:
    print('width =', img.width)
    print('height =', img.height)
```

It is a way of the lowest level to read an image. There will probably not be many cases to use it.

3.3.4 Clone an image

If you have an image already and have to copy it for safe manipulation, use clone () method:

```
from wand.image import Image
with Image(filename='pikachu.png') as original:
    with original.clone() as converted:
        converted.format = 'png'
        # operations on a converted image...
```

For some operations like format converting or cropping, there are safe methods that return a new image of manipulated result like convert () or slicing operator. So the above example code can be replaced by:

```
from wand.image import Image
with Image(filename='pikachu.png') as original:
    with original.convert('png') as converted:
        # operations on a converted image...
```

3.3.5 Hint file format

When it's read from a binary string or a file object, you can explicitly give the hint which indicates file format of an image to read — optional format keyword is for that:

```
from wand.image import Image
with Image(blob=image_binary, format='ico') as image:
    print(image.format)
```

New in version 0.2.1: The format parameter to Image constructor.

3.3.6 Open an empty image

To open an empty image, you have to set its width and height:

```
from wand.image import Image
with Image(width=200, height=100) as img:
    img.save(filename='200x100-transparent.png')
```

Its background color will be transparent by default. You can set background argument as well:

```
from wand.color import Color
from wand.image import Image
with Color('red') as bg:
```

```
with Image(width=200, height=100, background=bg) as img:
    img.save(filename='200x100-red.png')
```

New in version 0.2.2: The width, height, and background parameters to Image constructor.

3.4 Writing images

You can write an Image object into a file or a byte string buffer (blob) as format what you want.

3.4.1 Convert images to JPEG

If you wonder what is image's format, use format property.

```
>>> image.format
'JPEG'
```

The format property is writable, so you can convert images by setting this property.

```
from wand.image import Image
with Image(filename='pikachu.png') as img:
    img.format = 'jpeg'
    # operations to a jpeg image...
```

If you want to convert an image without any changes of the original, use convert () method instead:

```
from wand.image import Image
with Image(filename='pikachu.png') as original:
    with original.convert('jpeg') as converted:
        # operations to a jpeg image...
    pass
```

Note: Support for some of the formats are delegated to libraries or external programs. To get a complete listing of which image formats are supported on your system, use **identify** command provided by ImageMagick:

```
$ identify -list format
```

3.4.2 Save to file

In order to save an image to a file, use save () method with the keyword argument filename:

```
from wand.image import Image
with Image(filename='pikachu.png') as img:
    img.format = 'jpeg'
    img.save(filename='pikachu.jpg')
```

3.4.3 Save to stream

You can write an image into a output stream (file-like object which implements write() method) as well. The parameter file takes a such object (it also is the first positional parameter of save() method).

For example, the following code converts pikachu.png image into JPEG, gzips it, and then saves it to pikachu.jpg.gz:

```
import gzip
from wand.image import Image

gz = gzip.open('pikachu.jpg.gz')
with Image(filename='pikachu.png') as img:
    img.format = 'jpeg'
    img.save(file=gz)
gz.close()
```

3.4.4 Get binary string

Want just a binary string of the image? Use make_blob() method so:

```
from wand.image import Image
with image(filename='pikachu.png') as img:
    img.format = 'jpeg'
    jpeg_bin = img.make_blob()
```

There's the optional format parameter as well. So the above example code can be simpler:

```
from wand.image import Image
with Image(filename='pikachu.png') as img:
    jpeg_bin = img.make_blob('jpeg')
```

3.5 Resizing and cropping

Creating thumbnails (by resizing images) and cropping are most frequent works about images. This guide explains ways to deal with sizes of images.

Above all, to get the current size of the image check width and height properties:

If you want the pair of (width, height), check size property also.

Note: These three properties are all readonly.

3.5.1 Resize images

It scales an image into a desired size even if the desired size is larger than the original size. ImageMagick provides so many algorithms for resizing. The constant FILTER_TYPES contains names of filtering algorithms.

See also:

ImageMagick Resize Filters Demonstrates the results of resampling three images using the various resize filters and blur settings available in ImageMagick, and the file size of the resulting thumbnail images.

Image.resize() method takes width and height of a desired size, optional filter ('undefined' by default which means IM will try to guess best one to use) and optional blur (default is 1). It returns nothing but resizes itself in-place.

```
>>> img.size
(500, 600)
>>> img.resize(50, 60)
>>> img.size
(50, 60)
```

3.5.2 Sample images

Although Image.resize() provides many filter options, it's relatively slow. If speed is important for the job, you'd better use Image.sample() instead. It works in similar way to Image.resize() except it doesn't provide filter and blur options:

```
>>> img.size
(500, 600)
>>> img.sample(50, 60)
>>> img.size
(50, 60)
```

3.5.3 Crop images

To extract a sub-rectangle from an image, use the <code>crop()</code> method. It crops the image in-place. Its parameters are left, top, right, bottom in order.

```
>>> img.size
(200, 300)
>>> img.crop(10, 20, 50, 100)
>>> img.size
(40, 80)
```

It can also take keyword arguments width and height. These parameters replace right and bottom.

```
>>> img.size
(200, 300)
>>> img.crop(10, 20, width=40, height=80)
>>> img.size
(40, 80)
```

There is an another way to crop images: slicing operator. You can crop an image by [left:right, top:bottom] with maintaining the original:

```
>>> img.size
(300, 300)
>>> with img[10:50, 20:100] as cropped:
```

```
... print(cropped.size)
...
(40, 80)
>>> img.size
(300, 300)
```

3.5.4 Transform images

Use this function to crop and resize and image at the same time, using ImageMagick geometry strings. Cropping is performed first, followed by resizing.

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:

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

Other example calls:

```
# crop top left corner
img.transform('50%')

# scale height to 100px and preserve aspect ratio
img.transform(resize='x100')

# if larger than 640x480, fit within box, preserving aspect ratio
img.transform(resize='640x480>')

# crop a 320x320 square starting at 160x160 from the top left
img.transform(crop='320+160+160')
```

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.

3.5.5 Seam carving (also known as content-aware resizing)

New in version 0.3.0.

Seam carving is an algorithm for image resizing that functions by establishing a number of *seams* (paths of least importance) in an image and automatically removes seams to reduce image size or inserts seams to extend it.

In short: you can magickally resize images without distortion! See the following examples:





Cropped



Seam carving



You can easily rescale images with seam carving using Wand: use ${\tt Image.liquid_rescale}$ () method:

```
>>> image = Image(filename='seam.jpg')
>>> image.size
(320, 234)
>>> with image.clone() as resize:
...     resize.resize(234, 234)
...     resize.save(filename='seam-resize.jpg')
...     resize.size
...
(234, 234)
>>> with image[:234, :] as crop:
...     crop.save(filename='seam-crop.jpg')
...     crop.size
...
```

```
(234, 234)
>>> with image.clone() as liquid:
... liquid.liquid_rescale(234, 234)
... liquid.save(filename='seam-liquid.jpg')
... liquid.size
...
(234, 234)
```

Note: It may raise MissingDelegateError if your ImageMagick is configured --without-lqr option. In this case you should recompile ImageMagick.

See also:

Seam carving — Wikipedia The article which explains what seam carving is on Wikipedia.

Note: The image seam. jpg used in the above example is taken by D. Sharon Pruitt and licensed under CC-BY-2.0. It can be found the original photography from Flickr.

3.6 Transformation

Note: The image transform. jpg used in this docs is taken by Megan Trace, and licensed under CC BY-NC 2.0. It can be found the original photography from Flickr.

3.6.1 Rotation

New in version 0.1.8.

Image object provides a simple method to rotate images: rotate(). It takes a degree which can be 0 to 359. (Actually you can pass 360, 361, or more but it will be the same to 0, 1, or more respectively.)

For example, where the given image transform.jpg:



The below code makes the image rotated 90° to right:

```
from wand.image import Image
with Image(filename='transform.jpg') as image:
    with image.clone() as rotated:
        rotated.rotate(90)
        rotated.save(filename='transform-rotated-90.jpg')
```

The generated image transform-rotated-90.jpg looks like:



If degree is not multiples of 90, the optional parameter background will help (its default is transparent):

```
from wand.color import Color
from wand.image import Image

with Image(filename='transform.jpg') as image:
    with image.clone() as rotated:
        rotated.rotate(135, background=Color('rgb(229,221,112)'))
        rotated.save(filename='transform-rotated-135.jpg')
```

The generated image transform-rotated-135.jpg looks like:

3.6. Transformation 23



3.6.2 Flip and flop

New in version 0.3.0.

You can make a mirror image by reflecting the pixels around the central x- or y-axis. For example, where the given image transform.jpg:



The following code flips the image using Image.flip() method:

```
from wand.image import Image
with Image(filename='transform.jpg') as image:
    with image.clone() as flipped:
        flipped.flip()
        flipped.save(filename='transform-flipped.jpg')
```

The image transform-flipped.jpg generated by the above code looks like:



As like flip(), flop() does the same thing except it doesn't make a vertical mirror image but horizontal:

```
from wand.image import Image
with Image(filename='transform.jpg') as image:
    with image.clone() as flopped:
        flopped.flop()
        flopped.save(filename='transform-flopped.jpg')
```

3.6. Transformation 25

The image transform-flopped.jpg generated by the above code looks like:



3.7 Drawing

New in version 0.3.0.

The wand.drawing module provides some basic drawing functions. wand.drawing.Drawing object buffers instructions for drawing shapes into images, and then it can draw these shapes into zero or more images.

It's also callable and takes an Image object:

3.7.1 Lines

You can draw lines using line() method. It simply takes two (x, y) coordinates for start and end of a line. For example, the following code draws a diagonal line into the image:

```
draw.line((0, 0), image.size)
draw(image)
```

Or you can turn this diagonal line upside down:

```
draw.line((0, image.height), (image.width, 0))
draw(image)
```

The line color is determined by fill_color property, and you can change this of course. The following code draws a red diagonal line into the image:

```
from wand.color import Color
with Color('red') as color:
    draw.fill_color = color
    draw.line((0, 0), image.size)
    draw(image)
```

3.7.2 Rectangles

New in version 0.3.6.

If you want to draw rectangles use rectangle() method. It takes left/top coordinate, and right/bottom coordinate, or width and height. For example, the following code draws a square on the image:

```
draw.rectangle(left=10, top=10, right=40, bottom=40)
draw(image)
```

Or using width and height instead of right and bottom:

```
draw.rectangle(left=10, top=10, width=30, height=30)
draw(image)
```

Note that the stoke and the fill are determined by the following properties:

- stroke_color
- stroke_width
- fill_color

3.7.3 Texts

Drawing object can write texts as well using its text() method. It takes x and y coordinates to be drawn and a string to write:

```
draw.font = 'wandtests/assets/League_Gothic.otf'
draw.font_size = 40
draw.text(image.width / 2, image.height / 2, 'Hello, world!')
draw(image)
```

As the above code shows you can adjust several settings before writing texts:

- font
- font_size
- gravity
- text_alignment
- text_antialias
- text_decoration
- text_interline_spacing
- text_interword_spacing
- text_kerning
- text_under_color

3.7. Drawing 27

3.8 Colorspace

3.8.1 Image types

Every Image object has type property which identifies its colorspace. The value can be one of IMAGE_TYPES enumeration, and set of its available values depends on its format as well. For example, 'grayscale' isn't available on JPEG.

```
>>> from wand.image import Image
>>> with Image(filename='wandtests/assets/bilevel.gif') as img:
... img.type
...
'bilevel'
>>> with Image(filename='wandtests/assets/sasha.jpg') as img2:
... img2.type
...
'truecolor'

You can change this value:
with Image(filename='wandtests/assets/bilevel.gif') as img:
    img.type = 'truecolor'
    img.save(filename='truecolor.gif')
```

See also:

-type — ImageMagick: command-line-Options Corresponding command-line option of convert program.

3.8.2 Enable alpha channel

You can find whether an image has alpha channel and change it to have or not to have the alpha channel using alpha_channel property, which is preserving a bool value.

```
>>> with Image(filename='wandtests/assets/sasha.jpg') as img:
... img.alpha_channel
...
False
>>> with Image(filename='wandtests/assets/croptest.png') as img:
... img.alpha_channel
...
True

It's a writable property:
with Image(filename='wandtests/assets/sasha.jpg') as img:
    img.alpha_channel = True
```

3.9 Reading EXIF

New in version 0.3.0.

Image.metadata contains metadata of the image including EXIF. These are prefixed by 'exif:' e.g. 'exif:ExifVersion','exif:Flash'.

Here's a straightforward example to access EXIF of an image:

Note: You can't write into Image.metadata.

3.10 Sequence

Note: The image sequence-animation.gif used in this docs has been released into the public domain by its author, C6541 at Wikipedia project. This applies worldwide. (Source)

New in version 0.3.0.

Some images may actually consist of two or more images. For example, animated <code>image/gif</code> images consist of multiple frames. Some <code>image/ico</code> images have different sizes of icons.

For example, the above image sequence-animation.gif consists of the following frames (actually it has 60 frames, but we sample only few frames to show here):

3.10.1 sequence is a Sequence

If we open this image, Image object has sequence. It's a list-like object that maintain its all frames.

For example, len() for this returns the number of frames:

```
>>> from wand.image import Image
>>> with Image(filename='sequence-animation.gif') as image:
... len(image.sequence)
...
60
```

You can get an item by index from sequence:

```
>>> with Image(filename='sequence-animation.gif') as image:
... image.sequence[0]
...
<wand.sequence.SingleImage: ed84c1b (256x256)>
```

Or slice it:

3.10.2 Image versus SingleImage

Note that each item of sequence is a SingleImage instance, not Image.

3.10. Sequence 29

Image is a container that directly represents *image files* like sequence—animation.gif, and SingleImage is a single image that represents *frames* in animations or *sizes* in *image/ico* files.

They both inherit BaseImage, the common abstract class. They share the most of available operations and properties like resize() and size, but some are not. For example, save() and mimetype are only provided by Image. delay and index are only available for SingleImage.

In most cases, images don't have multiple images, so it's okay if you think that Image and SingleImage are the same, but be careful when you deal with animated image/gif files or image/ico files that contain multiple icons.

3.11 Resource management

See also:

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.

Objects Wand provides are resources to be managed. It has to be closed (destroyed) after using like file or database connection. You can deal with it using with very easily and explicitly:

```
with Image(filename='') as img:
    # deal with img...

Or you can call its destroy() (or close() if it is an Image instance) method manually:

try:
    img = Image(filename='')
    # deal with img...

finally:
    img.destroy()
```

Note: It also implements the destructor that invokes destroy (), and if your program runs on CPython (which does reference counting instead of ordinary garbage collection) most of resources are automatically deallocated.

However it's just depending on CPython's implementation detail of memory management, so it's not a good idea. If your program runs on PyPy (which implements garbage collector) for example, invocation time of destructors is not determined, so the program would be broken.

3.12 Running tests

Wand has unit tests and regression tests. It can be run using setup.py script:

```
$ python setup.py test
```

It uses pytest as its testing library. The above command will automatically install pytest as well if it's not installed yet.

Or you can manually install pytest and then use **py.test** command. It provides more options:

```
$ pip install pytest
$ py.test
```

3.12.1 Skipping tests

There are some time-consuming tests. You can skip these tests using --skip-slow option:

```
$ py.test --skip-slow
```

You can run only tests you want using -k option.

```
$ py.test -k image
```

3.12.2 Using tox

Wand should be compatible with various Python implementations including CPython 2.6, 2.7, PyPy. tox is a testing software that helps Python packages to test on various Python implementations at a time.

It can be installed using **easy_install** or **pip**:

```
$ easy_install tox
```

If you type just tox at Wand directory it will be tested on multiple Python interpreters:

```
$ tox
GLOB sdist-make: /Users/dahlia/Desktop/wand/setup.py
py26 create: /Users/dahlia/Desktop/wand/.tox/py26
py26 installdeps: pytest
py26 sdist-inst: /Users/dahlia/Desktop/wand/.tox/dist/Wand-0.2.2.zip
py26 runtests: commands[0]
...
```

You can use a double -- to pass options to pytest:

```
$ tox -- -k sequence
```

3.12.3 Continuous Integration

Travis CI automatically builds and tests every commit and pull request. The above banner image shows the current status of Wand build. You can see the detail of the current status from the following URL:

https://travis-ci.org/dahlia/wand

3.12.4 Code Coverage

Coveralls support tracking Wand's test coverage. The above banner image shows the current status of Wand coverage. You can see the details of the current status from the following URL:

https://coveralls.io/r/dahlia/wand

3.13 Roadmap

3.13.1 Version 0.4

CFFI Wand 0.4 will move to CFFI from ctypes.

Image layers (#22) Wand 0.4 will be able to deal with layers of an image.

Its branch name will be layer.

3.13. Roadmap 31

3.13.2 Very future versions

- **PIL compatibility layer** PIL has very long history and the most of Python projects still depend on it. We will work on PIL compatibility layer using Wand. It will provide two ways to emulate PIL:
 - Module-level compatibility which can be used by changing import:

```
try:
    from wand.pilcompat import Image
except ImportError:
    from PIL import Image
```

• Global monkeypatcher which changes sys.modules:

```
from wand.pilcompat.monkey import patch; patch()
import PIL.Image # it imports wand.pilcompat.Image module
```

CLI (**covert command**) **to Wand compiler** (#100) Primary interface of ImageMagick is **convert** command. It provides a small *parameter language*, and many answers on the Web contain code using this. The problem is that you can't simply copy-and-paste these code to utilize Wand.

This feature is to make these CLI codes possible to be used with Wand.

Supporting __array_interface__() for NumPy (#65) It makes numpy.asarray() able to take Image object to deal with its pixels as matrix.

Its branch name will be numpy.

3.14 Wand Changelog

3.14.1 0.3 series

Version 0.3.9

To be released.

- 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]

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 -00 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:

What's new in Wand 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 Elovskikh]
- 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.
- Ranamed 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]

3.14.2 0.2 series

Version 0.2.4

Released on May 28, 2013.

• Fix NameError in Resource.resource setter. [#89 forwareded 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 occured 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.

3.14.3 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 omittable.

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.

3.15 Talks and Presentations

3.15.1 Talks in 2012

• Lightning talk at Python Korea November 2012

References

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', 'extract', 'opaque', 'res (tuple) The list of alpha channel types

- •'undefined'
- •'activate'
- •'background'
- •'copy'
- •'deactivate'
- •'extract'
- •'opaque'
- •'reset'
- •'set'
- •'shape'
- •'transparent'
- •'flatten'
- •'remove'

See also:

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

wand.image.CHANNELS = {'opacity': 8, 'true_alpha': 64, 'gray': 1, 'rgb_channels': 128, 'yellow': 4, 'sync_channels': 256, 'd (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', 'ohta', 'lab', 'xyz', 'ycbcr', 'ycc', 'yiq', 'ypbp (tuple) The list of colorspaces.

- •'undefined'
- •'rgb'
- •'gray'
- •'transparent'
- •'ohta'
- •'lab'
- •'xyz'
- •'ycbcr'
- •'ycc'
- •'yiq'
- •'ypbpr'

- •' yuv'
- •'cmyk'
- •'srgb'
- •'hsb'
- •'hsl'
- •'hwb'
- •'rec601luma'
- •'rec601ycbcr'
- •'rec709luma'
- •'rec709ycbcr'
- •'log'
- •' cmy'
- •'luv'
- •'hcl'
- •'lch'
- •'lms'
- •'lchab'
- •'lchuv'
- •'scrgb'
- •'hsi'
- •'hsv'
- •'hclp'
- •'ydbdr'

See also:

ImageMagick Color Management Describes the ImageMagick color management operations

New in version 0.3.4.

wand.image.COMPOSITE_OPERATORS = ('undefined', 'no', 'add', 'atop', 'blend', 'bumpmap', 'change_mask', 'clear', 'colo (tuple) The list of composition operators

- •'undefined'
- •'no'
- •' add'
- •'atop'
- •'blend'
- •'bumpmap'
- •'change_mask'
- •'clear'

- •'color_burn'
- •'color_dodge'
- •'colorize'
- •'copy_black'
- •'copy_blue'
- •' copy'
- •'copy_cyan'
- •'copy_green'
- •'copy_magenta'
- •'copy_opacity'
- •'copy_red'
- •'copy_yellow'
- •'darken'
- •'dst_atop'
- •'dst'
- •'dst_in'
- •'dst_out'
- •'dst_over'
- •'difference'
- •'displace'
- •'dissolve'
- •'exclusion'
- •'hard_light'
- •'hue'
- •'in'
- •'lighten'
- •'linear_light'
- •'luminize'
- •'minus'
- $^{\bullet\prime}\, \texttt{modulate'}$
- $\bullet ' \verb| multiply'|$
- •'out'
- •'over'
- •'overlay'
- •'plus'
- •'replace'

- •'saturate'
- •'screen'
- •'soft_light'
- •'src_atop'
- •'src'
- •'src in'
- •'src_out'
- •'src_over'
- •'subtract'
- •'threshold'
- •'xor'
- •'divide'

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.EVALUATE_OPS = ('undefined', 'add', 'and', 'divide', 'leftshift', 'max', 'min', 'multiply', 'or', 'rightshift', 'set' (tuple) The list of evaluation operators

- •'undefined'
- •' add'
- •' and'
- •'divide'
- •'leftshift'
- •'max'
- •'min'
- •'multiply'
- •'or'
- •'rightshift'
- •'set'
- •'subtract'
- •'xor'
- •'pow'
- •'log'

- •'threshold'
- •'thresholdblack'
- •'thresholdwhite'
- 'gaussiannoise'
- •'impulsenoise'
- •'laplaciannoise'
- •'multiplicativenoise'
- •'poissonnoise'
- •'uniformnoise'
- •'cosine'
- •'sine'
- •'addmodulus'
- •'mean'
- •'abs'
- •'exponential'
- •'median'
- •'sum'

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', 'gaussia (tuple) 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.GRAVITY_TYPES = ('forget', 'north_west', 'north', 'north_east', 'west', 'center', 'east', 'south_west', 'south', ' (tuple) The list of gravity types.

New in version 0.3.0.

wand.image.IMAGE_TYPES = ('undefined', 'bilevel', 'grayscale', 'grayscalematte', 'palette', 'palettematte', 'truecolor', '

- •'undefined'
- •'bilevel'
- •'grayscale'
- •'grayscalematte'
- •'palette'
- •'palettematte'
- •'truecolor'
- •'truecolormatte'
- •'colorseparation'
- •'colorseparationmatte'
- •'optimize'

• 'palettebilevelmatte'

See also:

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

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

New in version 0.3.0.

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 classs, 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.

New in version 0.2.1.

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 consits 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.

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.

caption (*args, **kwargs)

Writes a caption text into the position.

- 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.

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.

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.

composite(*args, **kwargs)

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.

- 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 exceptions. Value Error when the given channel or operator is invalid

New in version 0.3.0.

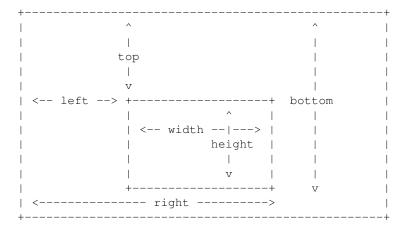
compression_quality

(numbers.Integral) Compression quality of this image.

New in version 0.2.0.

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*.

Raises exceptions. 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 0.1.8: Made to raise ValueError instead of IndexError for invalid width/height arguments.

New in version 0.1.7.

depth

(numbers.Integral) The depth of this image.

New in version 0.2.1.

dirty = None

(bool) Whether the image is changed or not.

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_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.

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.

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.

New in version 0.3.0.

liquid_rescale (*args, **kwargs)

Rescales the image with seam carving, also known as image retargeting, content-aware resizing, or liquid rescaling.

- 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.

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 exceptions. Value Error when one or more arguments are invalid

New in version 0.3.4.

negate (grayscale=False, channel=None)

Negate the colors in the reference image.

Parameters

- **grayscale** (bool) if set, only negate grayscale pixels within 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.

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.

quantum_range

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

New in version 0.2.0.

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. Sequence) The list of SingleImages that the image contains.

New in version 0.3.0.

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).

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

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

xheight Height given, width automagically 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 $\{+-\}\mathbf{x}\{+-\}\mathbf{y}$, that is, one plus or minus sign followed by an \mathbf{x} offset, followed by another plus or minus sign, followed by a \mathbf{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 thing 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.

transparent_color(*args, **kwargs)

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

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.

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 diffence amount

New in version 0.3.4.

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.

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.

 $\begin{array}{lll} \textbf{class} \ \texttt{wand.image.Image} \ (\textit{image=None}, & \textit{blob=None}, & \textit{file=None}, & \textit{filename=None}, & \textit{format=None}, \\ & \textit{width=None}, \textit{height=None}, \textit{background=None}, \textit{resolution=None}) \end{array}$

An image object.

- image (Image) makes an exact copy of the image
- blob (str) 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.
- height (numbers.Integral) the height of new blank imgage.
- background (wand.color.Color) an optional background color. default is transparent
- resolution (collections. Sequence, numbers. Integral) set a resolution value (dpi), usefull 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.

[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.

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 imgage.
- background (wand.color.Color) an optional background color. default is transparent

Returns blank image

Return type Image

New in version 0.3.0.

border (color, width, height)

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

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.

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.

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.

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.

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 omittable

Returns a blob (bytes) string

Return type str

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.

normalize(channel=None)

Normalize color channels.

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

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

Read new image into Image() object.

Parameters

- blob (str) 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.Sequence, numbers.Integral) set a resolution value (DPI), usefull 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.

- **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.

strip()

Strips an image of all profiles and comments.

New in version 0.2.0.

```
trim(color=None, fuzz=0)
```

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.

New in version 0.3.0: Optional color and fuzz parameters.

New in version 0.2.1.

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. 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.

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)
```

Mutable mapping of the image internal options. See available options in OPTIONS constant.

New in version 0.3.0.

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

Mark the operation manipulating itself instead of returning new one.

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.:

Parameters string (basestring) - a color namel 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.

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.

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.

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.

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

62

(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.

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.

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

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

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.
```

4.1.4 wand.drawing — Drawings

The module provides some vector drawing functions.

```
New in version 0.3.0.
wand.drawing.FONT_METRICS_ATTRIBUTES = ('character_width', 'character_height', 'ascender', 'descender', 'text_width', 'character_height', 'ascender', 'text_width', 'character_height', 'ascender', 'text_width', 'character_height', 'ascender', 'text_width', 'character_height', 'ascender', 'text_width', 'text_wi
                                (collections.Sequence) The attribute names of font metrics.
wand.drawing.TEXT_ALIGN_TYPES = ('undefined', 'left', 'center', 'right')
                                (collections. Sequence) The list of text align types.
                                                       •'undefined'
                                                       •'left'
                                                       •'center'
                                                       •'right'
wand.drawing.TEXT_DECORATION_TYPES = ('undefined', 'no', 'underline', 'overline', 'line_through')
                                (collections. Sequence) The list of text decoration types.
                                                       •'undefined'
                                                       •'no'
                                                       •'underline'
                                                       •'overline'
                                                       •'line_through'
wand.drawing.GRAVITY_TYPES = ('forget', 'north_west', 'north', 'north_east', 'west', 'center', 'east', 'south_west', 'south_west
                                (collections. Sequence) The list of text gravity types.
                                                       •'forget'
```

•'north west'

•'north'

•'north_east'

•'west'

•'center'

•'east'

•'south_west'

•'south'

•'south_east'

•'static'

```
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 rathan than this parameter

New in version 0.3.0.

clone()

Copies a drawing object.

Returns a duplication

Return type Drawing

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 (Image) – the image to be drawn

fill_color

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

font

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

font_size

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

get_font_metrics (image, text, multiline=False)

Queries font metrics from the given text.

Parameters

- image (Image) 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.

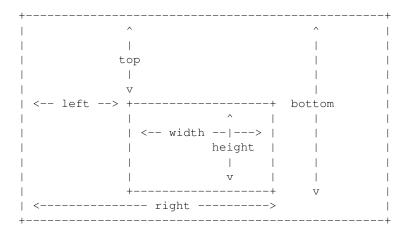
line (start, end)

Draws a line start to end.

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

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

Draws a rectangle using the current stoke color, stroke width, and fill color.



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

New in version 0.3.6.

stroke_color

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

New in version 0.3.3.

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).

- x (numbers.Integral) the left offset where to start writing a text
- ullet y (numbers.Integral) the top offset where to start writing a 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 ${\tt TEXT_DECORATION_TYPES}$ 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 encorouged.

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.

class wand.drawing.FontMetrics

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. Sequence prototocol.

New in version 0.3.0.

current index

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

Note: It's only for internal use.

```
index_context(*args, **kwds)
```

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.

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 invinsible to their containers until they are close()d(destroy()ed).

New in version 0.3.0.

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.

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 (*args, **kwds)

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 <code>DestroyedResourceError</code> when the resource has destroyed already.

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.BlobError

Bases: wand.exceptions.WandError, exceptions.IOError

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

exception wand.exceptions.BlobFatalError

Bases: wand.exceptions.WandFatalError, exceptions.IOError

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

exception wand.exceptions.BlobWarning

Bases: wand.exceptions.WandWarning, exceptions.IOError

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

wand.exceptions.CODE_MAP = [(<class 'wand.exceptions.WandWarning'>, 'Warning'), (<class 'wand.exceptions.WandErractions), (<class 'wand.exceptions), (<cl

exception wand.exceptions.CacheError

Bases: wand.exceptions.WandError

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

exception wand.exceptions.CacheFatalError

Bases: wand.exceptions.WandFatalError

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

exception wand.exceptions.CacheWarning

Bases: wand.exceptions.WandWarning

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

exception wand.exceptions.CoderError

Bases: wand.exceptions.WandError

There was a problem with an image coder.

exception wand.exceptions.CoderFatalError

Bases: wand.exceptions.WandFatalError

There was a problem with an image coder.

exception wand.exceptions.CoderWarning

Bases: wand.exceptions.WandWarning

There was a problem with an image coder.

$exception \verb| wand.exceptions.ConfigureError|\\$

Bases: wand.exceptions.WandError

There was a problem getting a configuration file.

exception wand.exceptions.ConfigureFatalError

Bases: wand.exceptions.WandFatalError

There was a problem getting a configuration file.

exception wand.exceptions.ConfigureWarning

 $Bases: {\tt wand.exceptions.WandWarning}$

There was a problem getting a configuration file.

exception wand.exceptions.CorruptImageError

Bases: wand.exceptions.WandError, exceptions.ValueError

The image file may be corrupt.

$exception \verb| wand.exceptions.CorruptImageFatalError| \\$

Bases: wand.exceptions.WandFatalError, exceptions.ValueError

The image file may be corrupt.

exception wand.exceptions.CorruptImageWarning

Bases: wand.exceptions.WandWarning, exceptions.ValueError

The image file may be corrupt.

wand.exceptions.DOMAIN_MAP = [('ResourceLimit', 'A program resource is exhausted e.g. not enough memory.', (<type '(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.WandError

An ImageMagick delegate failed to complete.

exception wand.exceptions.DelegateFatalError

Bases: wand.exceptions.WandFatalError

An ImageMagick delegate failed to complete.

exception wand.exceptions.DelegateWarning

Bases: wand.exceptions.WandWarning

An ImageMagick delegate failed to complete.

exception wand.exceptions.DrawError

Bases: wand.exceptions.WandError

A drawing operation failed.

exception wand.exceptions.DrawFatalError

Bases: wand.exceptions.WandFatalError

A drawing operation failed.

exception wand.exceptions.DrawWarning

Bases: wand.exceptions.WandWarning

A drawing operation failed.

exception wand.exceptions.FileOpenError

Bases: wand.exceptions.WandError, exceptions.IOError

The image file could not be opened for reading or writing.

exception wand.exceptions.FileOpenFatalError

Bases: wand.exceptions.WandFatalError, exceptions.IOError

The image file could not be opened for reading or writing.

exception wand.exceptions.FileOpenWarning

Bases: wand.exceptions.WandWarning, exceptions.IOError

The image file could not be opened for reading or writing.

exception wand.exceptions.ImageError

Bases: wand.exceptions.WandError

The operation could not complete due to an incompatible image.

exception wand.exceptions.ImageFatalError

 $Bases: \verb|wand.exceptions.WandFatalError| \\$

The operation could not complete due to an incompatible image.

exception wand.exceptions.ImageWarning

Bases: wand.exceptions.WandWarning

The operation could not complete due to an incompatible image.

exception wand.exceptions.MissingDelegateError

Bases: wand.exceptions.WandError, exceptions.ImportError

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

exception wand.exceptions.MissingDelegateFatalError

Bases: wand.exceptions.WandFatalError, exceptions.ImportError

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

exception wand.exceptions.MissingDelegateWarning

Bases: wand.exceptions.WandWarning,exceptions.ImportError

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

exception wand.exceptions.ModuleError

Bases: wand.exceptions.WandError

There was a problem with an image module.

exception wand.exceptions.ModuleFatalError

Bases: wand.exceptions.WandFatalError

There was a problem with an image module.

exception wand.exceptions.ModuleWarning

Bases: wand.exceptions.WandWarning

There was a problem with an image module.

exception wand.exceptions.MonitorError

Bases: wand.exceptions.WandError

There was a problem activating the progress monitor.

exception wand.exceptions.MonitorFatalError

Bases: wand.exceptions.WandFatalError

There was a problem activating the progress monitor.

exception wand.exceptions.MonitorWarning

Bases: wand.exceptions.WandWarning

There was a problem activating the progress monitor.

exception wand.exceptions.OptionError

 $Bases: \verb|wand.exceptions.WandError| \\$

A command-line option was malformed.

exception wand.exceptions.OptionFatalError

Bases: wand.exceptions.WandFatalError

A command-line option was malformed.

exception wand.exceptions.OptionWarning

 $Bases: {\tt wand.exceptions.WandWarning}$

A command-line option was malformed.

exception wand.exceptions.PolicyError

Bases: wand.exceptions.WandError

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

$exception \verb| wand.exceptions.PolicyFatalError|\\$

Bases: wand.exceptions.WandFatalError

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

exception wand.exceptions.PolicyWarning

Bases: wand.exceptions.WandWarning

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

exception wand.exceptions.RandomError

Bases: wand.exceptions.WandError

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

exception wand.exceptions.RandomFatalError

Bases: wand.exceptions.WandFatalError

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

exception wand.exceptions.RandomWarning

Bases: wand.exceptions.WandWarning

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

exception wand.exceptions.RegistryError

Bases: wand.exceptions.WandError

There was a problem getting or setting the registry.

exception wand.exceptions.RegistryFatalError

Bases: wand.exceptions.WandFatalError

There was a problem getting or setting the registry.

exception wand.exceptions.RegistryWarning

Bases: wand.exceptions.WandWarning

There was a problem getting or setting the registry.

exception wand.exceptions.ResourceLimitError

Bases: wand.exceptions.WandError, exceptions.MemoryError

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

exception wand.exceptions.ResourceLimitFatalError

Bases: wand.exceptions.WandFatalError, exceptions.MemoryError

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

exception wand.exceptions.ResourceLimitWarning

Bases: wand.exceptions.WandWarning, exceptions.MemoryError

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

exception wand.exceptions.StreamError

 $Bases: \verb|wand.exceptions.WandError|, exceptions.IOError|$

There was a problem reading or writing from a stream.

exception wand.exceptions.StreamFatalError

Bases: wand.exceptions.WandFatalError, exceptions.IOError

There was a problem reading or writing from a stream.

exception wand.exceptions.StreamWarning

 $Bases: \verb|wand.exceptions.WandWarning|, exceptions.IOError|\\$

There was a problem reading or writing from a stream.

wand.exceptions.TYPE_MAP = {385: <class 'wand.exceptions.MonitorWarning'>, 770: <class 'wand.exceptions.WandFata (dict) The dictionary of (code, exc_type).

exception wand.exceptions.TypeError

Bases: wand.exceptions.WandError

A font is unavailable; a substitution may have occurred.

exception wand.exceptions.TypeFatalError

Bases: wand.exceptions.WandFatalError

A font is unavailable; a substitution may have occurred.

exception wand.exceptions.TypeWarning

Bases: wand.exceptions.WandWarning

A font is unavailable; a substitution may have occurred.

exception wand.exceptions.WandError

Bases: wand.exceptions.WandError

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.WandFatalError

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.WandWarning

Bases: wand.exceptions.WandWarning

There was a problem specific to the MagickWand API.

$exception \verb| wand.exceptions.XServerError|\\$

 $Bases: \verb|wand.exceptions.WandError| \\$

An X resource is unavailable.

exception wand.exceptions.XServerFatalError

Bases: wand.exceptions.WandFatalError

An X resource is unavailable.

exception wand.exceptions.XServerWarning

Bases: wand.exceptions.WandWarning

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.c_magick_char_p

This subclass prevents the automatic conversion behavior of <code>ctypes.c_char_p</code>, allowing memory to be properly freed in the destructor. It must only be used for non-const character pointers returned by ImageMagick functions.

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
4.1.9 wand.compat — Compatibility layer
This module provides several subtle things to support multiple Python versions (2.6, 2.7, 3.2, 3.3) and VM implemen-
tations (CPython, PyPy).
wand.compat.PY3 = False
     (bool) Whether it is Python 3.x or not.
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
     (type) Type for representing binary data. str in Python 2 and bytes in Python 3.
     alias of 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, **kwds)
     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
     (type) Type for text data. basestring in Python 2 and str in Python 3.
     alias of 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
     (type) Type for representing Unicode textual data. unicode in Python 2 and str in Python 3.
     alias of unicode

class wand.compat.xrange
     The xrange() function. Alias for range() in Python 3.
```

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
0.3.9
$ python -m wand.version --verbose
Wand 0.3.9
ImageMagick 6.7.7-6 2012-06-03 Q16 http://www.imagemagick.org
```

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.

```
wand.version.VERSION = '0.3.9'
    (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, 3, 9)
    (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_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

(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_RELEASE_DATE_STRING = None

(datetime.date) The release date of the linked ImageMagick library. The same to the result of GetMagickReleaseDate() function.

New in version 0.2.1.

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.

Troubleshooting

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 sindikat's contribution.

5.3 Quora

There's a Quora topic for Wand: Wand (ImageMagick binding). Be free to add questions to the topic, though it's suitable for higher-level questions rather than troubleshooting.

Open source

Wand is an open source software written by Hong Minhee (initially written for StyleShare). 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/dahlia/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, 41
wand.api, 75
wand.color, 61
wand.compat, 76
wand.display, 77
wand.drawing, 64
wand.exceptions, 70
wand.font, 63
wand.image, 41
wand.resource, 68
wand.sequence, 68
wand.version, 77
```