ImageDraw Module

The ImageDraw module provides simple 2D graphics for Image objects. You can use this module to create new images, annotate or retouch existing images, and to generate graphics on the fly for web use.

For a more advanced drawing library for PIL, see the aggdraw module.

Example: Draw a gray cross over an image

```
from PIL import Image, ImageDraw

with Image.open("hopper.jpg") as im:

    draw = ImageDraw.Draw(im)
    draw.line((0, 0) + im.size, fill=128)
    draw.line((0, im.size[1], im.size[0], 0), fill=128)

    # write to stdout
    im.save(sys.stdout, "PNG")
```

Concepts

Coordinates

The graphics interface uses the same coordinate system as PIL itself, with (0, 0) in the upper left corner. Any pixels drawn outside of the image bounds will be discarded.

Colors

To specify colors, you can use numbers or tuples just as you would use with PIL.Image.new() or PIL.Image.Image.putpixel(). For “1”, “L”, and “I” images, use integers. For “RGB” images, use a 3-tuple containing integer values. For “F” images, use integer or floating point values.

For palette images (mode “P”), use integers as color indexes. In 1.1.4 and later, you can also use RGB 3-tuples or color names (see below). The drawing layer will automatically assign color indexes, as long as you don’t draw with more than 256 colors.

Color Names

See Color Names for the color names supported by Pillow.

Fonts

PIL can use bitmap fonts or OpenType/TrueType fonts.

Bitmap fonts are stored in PIL's own format, where each font typically consists of two files, one named .pil and the other usually named .pbm. The former contains font metrics, the latter raster data.

To load a bitmap font, use the load functions in the `ImageFont` module.

To load a OpenType/TrueType font, use the truetype function in the `ImageFont` module. Note that this function depends on third-party libraries, and may not available in all PIL builds.

Example: Draw Partial Opacity Text

```
from PIL import Image, ImageDraw, ImageFont
# get an image
base = Image.open("Pillow/Tests/images/hopper.png").convert("RGBA")

# make a blank image for the text, initialized to transparent text color
txt = Image.new("RGBA", base.size, (255,255,255,0))

# get a font
fnt = ImageFont.truetype("Pillow/Tests/fonts/FreeMono.ttf", 40)
# get a drawing context
d = ImageDraw.Draw(txt)

# draw text, half opacity
d.text((10,10), "Hello", font=fnt, fill=(255,255,255,128))
# draw text, full opacity
d.text((10,60), "World", font=fnt, fill=(255,255,255,255))

out = Image.alpha_composite(base, txt)

out.show()
```

Example: Draw Multiline Text

```
from PIL import Image, ImageDraw, ImageFont

# create an image
out = Image.new("RGB", (150, 100), (255, 255, 255))

# get a font
fnt = ImageFont.truetype("Pillow/Tests/fonts/FreeMono.ttf", 40)
# get a drawing context
d = ImageDraw.Draw(out)

# draw multiline text
d.multiline_text((10,10), "Hello\nWorld", font=fnt, fill=(0, 0, 0))

out.show()
```

Functions

PIL.ImageDraw.Draw(*im, mode=None*) [source]

Creates an object that can be used to draw in the given image.

Note that the image will be modified in place.

> **Parameters**
>
> - **im** – The image to draw in.
> - **mode** – Optional mode to use for color values. For RGB images, this argument can be RGB or RGBA (to blend the drawing into the image). For all other modes, this argument must be the same as the image mode. If omitted, the mode defaults to the mode of the image.

Methods

ImageDraw.getfont() [source]

Get the current default font.

> **Returns**
>
> An image font.

ImageDraw.arc(*xy, start, end, fill=None, width=0*) [source]

Draws an arc (a portion of a circle outline) between the start and end angles, inside the given bounding box.

> **Parameters**
>
> - **xy** – Two points to define the bounding box. Sequence of `[(x0, y0), (x1, y1)]` or `[x0, y0, x1, y1]`, where `x1 >= x0` and `y1 >= y0`.

- **start** – Starting angle, in degrees. Angles are measured from 3 o'clock, increasing clockwise.
- **end** – Ending angle, in degrees.
- **fill** – Color to use for the arc.
- **width** –

  The line width, in pixels.

  New in version 5.3.0.

`ImageDraw.bitmap`*(xy, bitmap, fill=None)* [source]

Draws a bitmap (mask) at the given position, using the current fill color for the non-zero portions. The bitmap should be a valid transparency mask (mode "1") or matte (mode "L" or "RGBA").

This is equivalent to doing `image.paste(xy, color, bitmap)`.

To paste pixel data into an image, use the `paste()` method on the image itself.

`ImageDraw.chord`*(xy, start, end, fill=None, outline=None, width=1)* [source]

Same as `arc()`, but connects the end points with a straight line.

> **Parameters**

- **xy** – Two points to define the bounding box. Sequence of `[(x0, y0), (x1, y1)]` or `[x0, y0, x1, y1]`, where `x1 >= x0` and `y1 >= y0`.
- **outline** – Color to use for the outline.
- **fill** – Color to use for the fill.
- **width** –

  The line width, in pixels.

  New in version 5.3.0.

`ImageDraw.ellipse`*(xy, fill=None, outline=None, width=1)* [source]

Draws an ellipse inside the given bounding box.

> **Parameters**

- **xy** – Two points to define the bounding box. Sequence of either `[(x0, y0), (x1, y1)]` or `[x0, y0, x1, y1]`, where `x1 >= x0` and `y1 >= y0`.
- **outline** – Color to use for the outline.

- **fill** – Color to use for the fill.
- **width** –

  The line width, in pixels.

  New in version 5.3.0.

`ImageDraw.line`(*xy, fill=None, width=0, joint=None*) [source]

Draws a line between the coordinates in the `xy` list.

> Parameters

- **xy** – Sequence of either 2-tuples like `[(x, y), (x, y), ...]` or numeric values like `[x, y, x, y, ...]`.
- **fill** – Color to use for the line.
- **width** –

  The line width, in pixels.

  New in version 1.1.5.

  > Note
  >
  > This option was broken until version 1.1.6.

- **joint** –

  Joint type between a sequence of lines. It can be `"curve"`, for rounded edges, or `None`.

  New in version 5.3.0.

`ImageDraw.pieslice`(*xy, start, end, fill=None, outline=None, width=1*) [source]

Same as arc, but also draws straight lines between the end points and the center of the bounding box.

> Parameters

- **xy** – Two points to define the bounding box. Sequence of `[(x0, y0), (x1, y1)]` or `[x0, y0, x1, y1]`, where `x1 >= x0` and `y1 >= y0`.
- **start** – Starting angle, in degrees. Angles are measured from 3 o'clock, increasing clockwise.
- **end** – Ending angle, in degrees.
- **fill** – Color to use for the fill.

- **outline** – Color to use for the outline.
- **width** –

  The line width, in pixels.

  New in version 5.3.0.

`ImageDraw.point`(*xy, fill=None*) [source]

Draws points (individual pixels) at the given coordinates.

> Parameters

- **xy** – Sequence of either 2-tuples like `[(x, y), (x, y), ...]` or numeric values like `[x, y, x, y, ...]`.
- **fill** – Color to use for the point.

`ImageDraw.polygon`(*xy, fill=None, outline=None*) [source]

Draws a polygon.

The polygon outline consists of straight lines between the given coordinates, plus a straight line between the last and the first coordinate.

> Parameters

- **xy** – Sequence of either 2-tuples like `[(x, y), (x, y), ...]` or numeric values like `[x, y, x, y, ...]`.
- **outline** – Color to use for the outline.
- **fill** – Color to use for the fill.

`ImageDraw.regular_polygon`(*bounding_circle, n_sides, rotation=0, fill=None, outline=None*) [source]

Draws a regular polygon inscribed in `bounding_circle`, with `n_sides`, and rotation of `rotation` degrees.

> Parameters

- **bounding_circle** – The bounding circle is a tuple defined by a point and radius. (e.g. `bounding_circle=(x, y, r)` or `((x, y), r)`). The polygon is inscribed in this circle.
- **n_sides** – Number of sides (e.g. `n_sides=3` for a triangle, `6` for a hexagon).
- **rotation** – Apply an arbitrary rotation to the polygon (e.g. `rotation=90`, applies a 90 degree rotation).
- **fill** – Color to use for the fill.
- **outline** – Color to use for the outline.

ImageDraw.rectangle(*xy, fill=None, outline=None, width=1*) [source]

Draws a rectangle.

Parameters

- **xy** – Two points to define the bounding box. Sequence of either `[(x0, y0), (x1, y1)]` or `[x0, y0, x1, y1]`. The second point is just outside the drawn rectangle.
- **outline** – Color to use for the outline.
- **fill** – Color to use for the fill.
- **width** –

  The line width, in pixels.

  New in version 5.3.0.

ImageDraw.shape(*shape, fill=None, outline=None*) [source]

Warning

This method is experimental.

Draw a shape.

ImageDraw.text(*xy, text, fill=None, font=None, anchor=None, spacing=4, align='left', direction=None, features=None, language=None, stroke_width=0, stroke_fill=None, embedded_color=False*) [source]

Draws the string at the given position.

Parameters

- **xy** – The anchor coordinates of the text.
- **text** – Text to be drawn. If it contains any newline characters, the text is passed on to `multiline_text()`.
- **fill** – Color to use for the text.
- **font** – An `ImageFont` instance.
- **anchor** –

  The text anchor alignment. Determines the relative location of the anchor to the text. The default alignment is top left. See Text anchors for valid values. This parameter is ignored for non-TrueType fonts.

Note

This parameter was present in earlier versions of Pillow, but implemented only in version 8.0.0.

- **spacing** – If the text is passed on to `multiline_text()`, the number of pixels between lines.
- **align** – If the text is passed on to `multiline_text()`, `"left"`, `"center"` or `"right"`. Determines the relative alignment of lines. Use the `anchor` parameter to specify the alignment to `xy`.
- **direction** –

  Direction of the text. It can be `"rtl"` (right to left), `"ltr"` (left to right) or `"ttb"` (top to bottom). Requires libraqm.

  New in version 4.2.0.

- **features** –

  A list of OpenType font features to be used during text layout. This is usually used to turn on optional font features that are not enabled by default, for example `"dlig"` or `"ss01"`, but can be also used to turn off default font features, for example `"-liga"` to disable ligatures or `"-kern"` to disable kerning. To get all supported features, see OpenType docs. Requires libraqm.

  New in version 4.2.0.

- **language** –

  Language of the text. Different languages may use different glyph shapes or ligatures. This parameter tells the font which language the text is in, and to apply the correct substitutions as appropriate, if available. It should be a BCP 47 language code. Requires libraqm.

  New in version 6.0.0.

- **stroke_width** –

  The width of the text stroke.

  New in version 6.2.0.

- **stroke_fill** –

  Color to use for the text stroke. If not given, will default to the `fill` parameter.

  New in version 6.2.0.

- **embedded_color** –

Whether to use font embedded color glyphs (COLR or CBDT).

New in version 8.0.0.

ImageDraw.multiline_text(*xy, text, fill=None, font=None, anchor=None, spacing=4, align='left', direction=None, features=None, language=None, stroke_width=0, stroke_fill=None, embedded_color=False*)
[source]

Draws the string at the given position.

> **Parameters**

- **xy** – The anchor coordinates of the text.
- **text** – Text to be drawn.
- **fill** – Color to use for the text.
- **font** – An `ImageFont` instance.
- **anchor** –

  The text anchor alignment. Determines the relative location of the anchor to the text. The default alignment is top left. See Text anchors for valid values. This parameter is ignored for non-TrueType fonts.

  > **Note**
  >
  > This parameter was present in earlier versions of Pillow, but implemented only in version 8.0.0.

- **spacing** – The number of pixels between lines.
- **align** – `"left"`, `"center"` or `"right"`. Determines the relative alignment of lines. Use the `anchor` parameter to specify the alignment to `xy`.
- **direction** –

  Direction of the text. It can be `"rtl"` (right to left), `"ltr"` (left to right) or `"ttb"` (top to bottom). Requires libraqm.

  New in version 4.2.0.

- **features** –

  A list of OpenType font features to be used during text layout. This is usually used to turn on optional font features that are not enabled by default, for example `"dlig"` or `"ss01"`, but can be also used to turn off default font features, for example `"-liga"` to disable ligatures or `"-kern"` to disable kerning. To get all supported features, see OpenType docs. Requires libraqm.

New in version 4.2.0.

- **language** –

  Language of the text. Different languages may use different glyph shapes or ligatures. This parameter tells the font which language the text is in, and to apply the correct substitutions as appropriate, if available. It should be a BCP 47 language code. Requires libraqm.

  New in version 6.0.0.

- **stroke_width** –

  The width of the text stroke.

  New in version 6.2.0.

- **stroke_fill** –

  > **Color to use for the text stroke. If not given, will default to**
  >
  > the `fill` parameter.

  New in version 6.2.0.

- **embedded_color** –

  Whether to use font embedded color glyphs (COLR or CBDT).

  New in version 8.0.0.

`ImageDraw.textsize`(*text, font=None, spacing=4, direction=None, features=None, language=None, stroke_width=0*) [source]

Return the size of the given string, in pixels.

Use `textlength()` to measure the offset of following text with 1/64 pixel precision. Use `textbbox()` to get the exact bounding box based on an anchor.

> **Note**
>
> For historical reasons this function measures text height from the ascender line instead of the top, see Text anchors. If you wish to measure text height from the top, it is recommended to use `textbbox()` with `anchor='lt'` instead.

> **Parameters**

- **text** – Text to be measured. If it contains any newline characters, the text is passed on to `multiline_textsize()`.
- **font** – An `ImageFont` instance.

- **spacing** – If the text is passed on to `multiline_textsize()`, the number of pixels between lines.
- **direction** –

  Direction of the text. It can be `"rtl"` (right to left), `"ltr"` (left to right) or `"ttb"` (top to bottom). Requires libraqm.

  New in version 4.2.0.
- **features** –

  A list of OpenType font features to be used during text layout. This is usually used to turn on optional font features that are not enabled by default, for example `"dlig"` or `"ss01"`, but can be also used to turn off default font features, for example `"-liga"` to disable ligatures or `"-kern"` to disable kerning. To get all supported features, see OpenType docs. Requires libraqm.

  New in version 4.2.0.
- **language** –

  Language of the text. Different languages may use different glyph shapes or ligatures. This parameter tells the font which language the text is in, and to apply the correct substitutions as appropriate, if available. It should be a BCP 47 language code. Requires libraqm.

  New in version 6.0.0.
- **stroke_width** –

  The width of the text stroke.

  New in version 6.2.0.

ImageDraw.multiline_textsize(*text, font=None, spacing=4, direction=None, features=None, language=None, stroke_width=0*) [source]

Return the size of the given string, in pixels.

Use `textlength()` to measure the offset of following text with 1/64 pixel precision. Use `textbbox()` to get the exact bounding box based on an anchor.

> **Note**
>
> For historical reasons this function measures text height as the distance between the top ascender line and bottom descender line, not the top and bottom of the text, see Text anchors. If you wish to measure text height from the top to the bottom of text, it is recommended to use `multiline_textbbox()` instead.

Parameters

- **text** – Text to be measured.
- **font** – An `ImageFont` instance.
- **spacing** – The number of pixels between lines.
- **direction** –

  Direction of the text. It can be `"rtl"` (right to left), `"ltr"` (left to right) or `"ttb"` (top to bottom). Requires libraqm.

  New in version 4.2.0.

- **features** –

  A list of OpenType font features to be used during text layout. This is usually used to turn on optional font features that are not enabled by default, for example `"dlig"` or `"ss01"`, but can be also used to turn off default font features, for example `"-liga"` to disable ligatures or `"-kern"` to disable kerning. To get all supported features, see OpenType docs. Requires libraqm.

  New in version 4.2.0.

- **language** –

  Language of the text. Different languages may use different glyph shapes or ligatures. This parameter tells the font which language the text is in, and to apply the correct substitutions as appropriate, if available. It should be a BCP 47 language code. Requires libraqm.

  New in version 6.0.0.

- **stroke_width** –

  The width of the text stroke.

  New in version 6.2.0.

`ImageDraw.textlength`(*text, font=None, direction=None, features=None, language=None, embedded_color=False*) [source]

Returns length (in pixels with 1/64 precision) of given text when rendered in font with provided direction, features, and language.

This is the amount by which following text should be offset. Text bounding box may extend past the length in some fonts, e.g. when using italics or accents.

The result is returned as a float; it is a whole number if using basic layout.

Note that the sum of two lengths may not equal the length of a concatenated string due to kerning. If you need to adjust for kerning, include the following character and subtract its length.

For example, instead of

```
hello = draw.textlength("Hello", font)
world = draw.textlength("World", font)
hello_world = hello + world  # not adjusted for kerning
assert hello_world == draw.textlength("HelloWorld", font)  # may fail
```

use

```
hello = draw.textlength("HelloW", font) - draw.textlength("W", font)  # adjusted for kerning
world = draw.textlength("World", font)
hello_world = hello + world  # adjusted for kerning
assert hello_world == draw.textlength("HelloWorld", font)  # True
```

or disable kerning with (requires libraqm)

```
hello = draw.textlength("Hello", font, features=["-kern"])
world = draw.textlength("World", font, features=["-kern"])
hello_world = hello + world  # kerning is disabled, no need to adjust
assert hello_world == draw.textlength("HelloWorld", font, features=["-kern"])  # True
```

New in version 8.0.0.

Parameters

- **text** – Text to be measured. May not contain any newline characters.
- **font** – An `ImageFont` instance.
- **direction** – Direction of the text. It can be `"rtl"` (right to left), `"ltr"` (left to right) or `"ttb"` (top to bottom). Requires libraqm.
- **features** – A list of OpenType font features to be used during text layout. This is usually used to turn on optional font features that are not enabled by default, for example `"dlig"` or `"ss01"`, but can be also used to turn off default font features, for example `"-liga"` to disable ligatures or `"-kern"` to disable kerning. To get all supported features, see OpenType docs. Requires libraqm.
- **language** – Language of the text. Different languages may use different glyph shapes or ligatures. This parameter tells the font which language the text is in, and to apply the correct substitutions as appropriate, if available. It should be a BCP 47 language code. Requires libraqm.
- **embedded_color** – Whether to use font embedded color glyphs (COLR or CBDT).

ImageDraw.textbbox(*xy, text, font=None, anchor=None, spacing=4, align='left', direction=None,*

features=None, language=None, stroke_width=0, embedded_color=False) [source]

Returns bounding box (in pixels) of given text relative to given anchor when rendered in font with provided direction, features, and language. Only supported for TrueType fonts.

Use `textlength()` to get the offset of following text with 1/64 pixel precision. The bounding box includes extra margins for some fonts, e.g. italics or accents.

New in version 8.0.0.

> **Parameters**

- **xy** – The anchor coordinates of the text.
- **text** – Text to be measured. If it contains any newline characters, the text is passed on to `multiline_textbbox()`.
- **font** – A `FreeTypeFont` instance.
- **anchor** – The text anchor alignment. Determines the relative location of the anchor to the text. The default alignment is top left. See Text anchors for valid values. This parameter is ignored for non-TrueType fonts.
- **spacing** – If the text is passed on to `multiline_textbbox()`, the number of pixels between lines.
- **align** – If the text is passed on to `multiline_textbbox()`, `"left"`, `"center"` or `"right"`. Determines the relative alignment of lines. Use the `anchor` parameter to specify the alignment to `xy`.
- **direction** – Direction of the text. It can be `"rtl"` (right to left), `"ltr"` (left to right) or `"ttb"` (top to bottom). Requires libraqm.
- **features** – A list of OpenType font features to be used during text layout. This is usually used to turn on optional font features that are not enabled by default, for example `"dlig"` or `"ss01"`, but can be also used to turn off default font features, for example `"-liga"` to disable ligatures or `"-kern"` to disable kerning. To get all supported features, see OpenType docs. Requires libraqm.
- **language** – Language of the text. Different languages may use different glyph shapes or ligatures. This parameter tells the font which language the text is in, and to apply the correct substitutions as appropriate, if available. It should be a BCP 47 language code. Requires libraqm.
- **stroke_width** – The width of the text stroke.
- **embedded_color** – Whether to use font embedded color glyphs (COLR or CBDT).

`ImageDraw.multiline_textbbox`(*xy, text, font=None, anchor=None, spacing=4, align='left', direction=None, features=None, language=None, stroke_width=0, embedded_color=False*) [source]

Returns bounding box (in pixels) of given text relative to given anchor when rendered in font with provided direction, features, and language. Only supported for TrueType fonts.

Use `textlength()` to get the offset of following text with 1/64 pixel precision. The bounding box includes extra margins for some fonts, e.g. italics or accents.

New in version 8.0.0.

> **Parameters**

- **xy** – The anchor coordinates of the text.
- **text** – Text to be measured.
- **font** – A `FreeTypeFont` instance.
- **anchor** – The text anchor alignment. Determines the relative location of the anchor to the text. The default alignment is top left. See Text anchors for valid values. This parameter is ignored for non-TrueType fonts.
- **spacing** – The number of pixels between lines.
- **align** – `"left"`, `"center"` or `"right"`. Determines the relative alignment of lines. Use the `anchor` parameter to specify the alignment to `xy`.
- **direction** – Direction of the text. It can be `"rtl"` (right to left), `"ltr"` (left to right) or `"ttb"` (top to bottom). Requires libraqm.
- **features** – A list of OpenType font features to be used during text layout. This is usually used to turn on optional font features that are not enabled by default, for example `"dlig"` or `"ss01"`, but can be also used to turn off default font features, for example `"-liga"` to disable ligatures or `"-kern"` to disable kerning. To get all supported features, see OpenType docs. Requires libraqm.
- **language** – Language of the text. Different languages may use different glyph shapes or ligatures. This parameter tells the font which language the text is in, and to apply the correct substitutions as appropriate, if available. It should be a BCP 47 language code. Requires libraqm.
- **stroke_width** – The width of the text stroke.
- **embedded_color** – Whether to use font embedded color glyphs (COLR or CBDT).

`PIL.ImageDraw.getdraw`(*im=None, hints=None*) [source]

> **Warning**
>
> This method is experimental.

A more advanced 2D drawing interface for PIL images, based on the WCK interface.

Parameters

- **im** – The image to draw in.
- **hints** – An optional list of hints.

Returns

A (drawing context, drawing resource factory) tuple.

`PIL.ImageDraw.floodfill`*(image, xy, value, border=None, thresh=0)* [source]

Warning

This method is experimental.

Fills a bounded region with a given color.

Parameters

- **image** – Target image.
- **xy** – Seed position (a 2-item coordinate tuple).
- **value** – Fill color.
- **border** – Optional border value. If given, the region consists of pixels with a color different from the border color. If not given, the region consists of pixels having the same color as the seed pixel.
- **thresh** – Optional threshold value which specifies a maximum tolerable difference of a pixel value from the 'background' in order for it to be replaced. Useful for filling regions of non- homogeneous, but similar, colors.