The ImageDraw Module

The **ImageDraw** module provide 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 Grey Cross Over an Image

```
import Image, ImageDraw

im = Image.open("lena.pgm")

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

# 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.

Colours

To specify colours, you can use numbers or tuples just as you would use with **Image.new** or **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 colour indexes. In 1.1.4 and later, you can also use RGB 3-tuples or colour names (see below). The drawing layer will automatically assign colour indexes, as long as you don’t draw with more than 256 colours.

Colour Names

In PIL 1.1.4 and later, you can also use string constants when drawing in “RGB” images. PIL supports the following string formats:

- Hexadecimal color specifiers, given as “#rgb” or “#rrggbb”. For example, **“#ff0000”** specifies pure red.
- RGB functions, given as “rgb(red, green, blue)” where the colour values are integers in the range 0 to 255. Alternatively, the color values can be given as three percentages (0% to 100%). For example, **“rgb(255,0,0)”** and **“rgb(100%,0%,0%)”** both specify pure red.
- Hue-Saturation-Lightness (HSL) functions, given as **“hsl(hue, saturation%, lightness%)”** where hue is the colour given as an angle between 0 and 360 (red=0, green=120, blue=240), saturation is a value between 0% and 100% (gray=0%, full color=100%), and lightness is a value between 0% and 100% (black=0%, normal=50%,

white=100%). For example, **“hsl(0,100%,50%)”** is pure red.

- Common HTML colour names. The **ImageDraw** provides some 140 standard colour names, based on the colors supported by the X Window system and most web browsers. Colour names are case insensitive, and may contain whitespace. For example, **“red”** and **“Red”** both specify pure red.

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 a 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.

(IronPIL) To load a built-in font, use the **Font** constructor in the **ImageFont** module.

Functions

Draw

Draw(image) ⇒ Draw instance

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

(IronPIL) Instead of an image, you can use HWND or HDC objects from the **ImageWin** module. This allows you to draw directly to the screen.

Note that the image will be modified in place.

Methods

arc

draw.arc(xy, start, end, options)

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

The **outline** option gives the colour to use for the arc.

bitmap

draw.bitmap(xy, bitmap, options)

Draws a bitmap (mask) at the given position, using the current fill colour 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.

chord

draw.chord(xy, start, end, options)

Same as **arc**, but connects the end points with a straight line.

The **outline** option gives the colour to use for the chord outline. The **fill** option gives the colour to use for the chord interior.

ellipse

draw.ellipse(xy, options)

Draws an ellipse inside the given bounding box.

The **outline** option gives the colour to use for the ellipse outline. The **fill** option gives the colour to use for the ellipse interior.

line

draw.line(xy, options)

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

The coordinate list can be any sequence object containing either 2-tuples [(**x**, **y**), ...] or numeric values [**x**, **y**, ...]. It should contain at least two coordinates.

The **fill** option gives the colour to use for the line.

(New in 1.1.5) The **width** option gives the line width, in pixels. Note that line joins are not handled well, so wide polylines will not look good.

> **Note:** The **width** option is broken in 1.1.5. The line is drawn with twice the width you specify. This will be fixed in 1.1.6.

pieslice

draw.pieslice(xy, start, end, options)

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

The **outline** option gives the colour to use for the pieslice outline. The **fill** option gives the colour to use for the pieslice interior.

point

draw.point(xy, options)

Draws points (individual pixels) at the given coordinates.

The coordinate list can be any sequence object containing either 2-tuples [(**x**, **y**), ...] or numeric values [**x**, **y**, ...].

The **fill** option gives the colour to use for the points.

polygon

draw.polygon(xy, options)

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.

The coordinate list can be any sequence object containing either 2-tuples [(**x**, **y**), ...] or numeric values [**x**, **y**, ...]. It should contain at least three coordinates.

The **outline** option gives the colour to use for the polygon outline. The **fill** option gives the colour to use for the polygon interior.

rectangle

draw.rectangle(box, options)

Draws a rectangle.

The box can be any sequence object containing either 2-tuples [(**x**, **y**), (**x**, **y**)] or numeric values [**x**, **y**, **x**, **y**]. It should contain two coordinates.

Note that the second coordinate pair defines a point just outside the rectangle, also when the rectangle is not filled.

The **outline** option gives the colour to use for the rectangle outline. The **fill** option gives the colour to use for the rectangle interior.

text

draw.text(position, string, options)

Draws the string at the given position. The position gives the upper left corner of the text.

The **font** option is used to specify which font to use. It should be an instance of the **ImageFont** class, typically loaded from file using the **load** method in the **ImageFont** module.

The **fill** option gives the colour to use for the text.

textsize

draw.textsize(string, options) ⇒ (width, height)

Return the size of the given string, in pixels.

The **font** option is used to specify which font to use. It should be an instance of the **ImageFont** class, typically loaded from file using the **load** method in the **ImageFont** module.

Options

outline

outline integer or tuple

fill

fill integer or tuple

font

font ImageFont instance

Compatibility

The **Draw** class contains a constructor and a number of methods which are provided for backwards compatibility only. For this to work properly, you should *either* use options on the drawing primitives, or these methods. Do not mix the old and new calling conventions.

(IronPIL) The compatibility methods are not supported by IronPIL.

ImageDraw

ImageDraw(image) ⇒ Draw instance

(Deprecated). Same as **Draw**. Don't use this name in new code.

setink

draw.setink(ink)

(Deprecated). Sets the colour to use for subsequent draw and fill operations.

setfill

draw.setfill(mode)

(Deprecated). Sets the fill mode.

If the mode is 0, subsequently drawn shapes (like polygons and rectangles) are outlined. If the mode is 1, they are filled.

setfont

draw.setfont(font)

(Deprecated). Sets the default font to use for the **text** method.

The **font** argument should be an instance of the **ImageFont** class, typically loaded from file using the **load** method in the **ImageFont** module.

back next