
Freetype Python Documentation

Release 0.4.1

Nicolas P. Rougier

Sep 18, 2017

Contents

1	Introduction	3
2	Usage example	5
3	Screenshots	7
4	API	11
5	Release notes	51
6	License	53
7	Indices and tables	55

Freetype python provides bindings for the FreeType library. Only the high-level API is bound.

Freetype-py lives at <https://github.com/rougier/freetype-py/>

To be able to use freetype python, you need the freetype library version 2 installed on your system.

Pre-requisites

You need to have the Freetype library installed on your system to be able to use the freetype python bindings.

<p>Warning: If you don't compile the Freetype library yourself, chances are subpixel anti-aliasing will be disabled due to patent problems. Have a look at Freetype FAQ to know how to enable it.</p>
--

Mac users

Freetype should be already installed on your system. If not, either install it using [homebrew](#) or compile it and place the library binary file in `'usr/local/lib'`.

Linux users

Freetype should be already installed on your system. If not, either install relevant package from your package manager or compile from sources and place the library binary file in `'usr/local/lib'`.

Window users

You can try to install a window binaries available from the Freetype site or you can compile it from sources. In such a case, make sure the resulting library binaries is named `'Freetype.dll'` (and not something like `Freetype245.dll`) and make sure to place a copy in `Windows/System32` directory.

Installation

The easiest way to install freetype-py is to use pip:

```
pip install freetype-py
```

Or you can get the latest version from git and install yourself:

```
git clone https://github.com/rougier/freetype-py.git
cd freetype-py
python setup.py install
```


CHAPTER 2

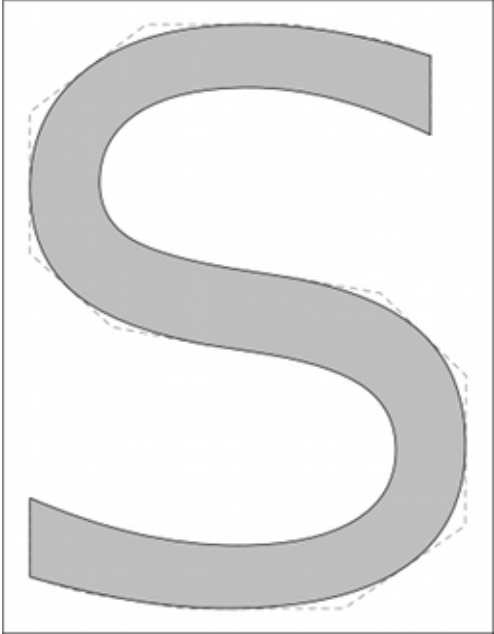
Usage example

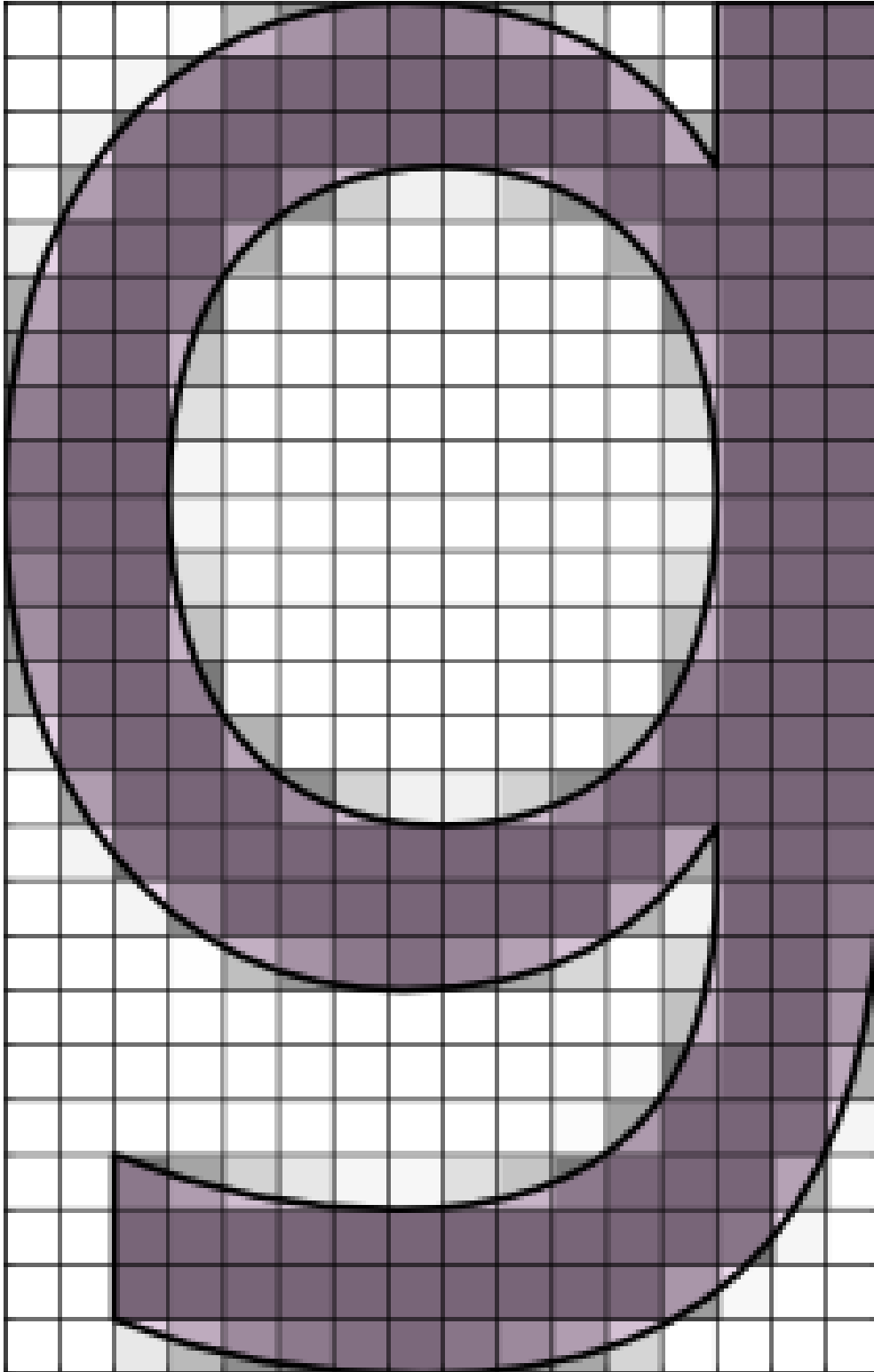
```
import freetype
face = freetype.Face("Vera.ttf")
face.set_char_size( 48*64 )
face.load_char('S')
bitmap = face.glyph.bitmap
print bitmap.buffer
```


Screenshot below comes from the `wordle.py` example. No clever tricks here, just brute force.

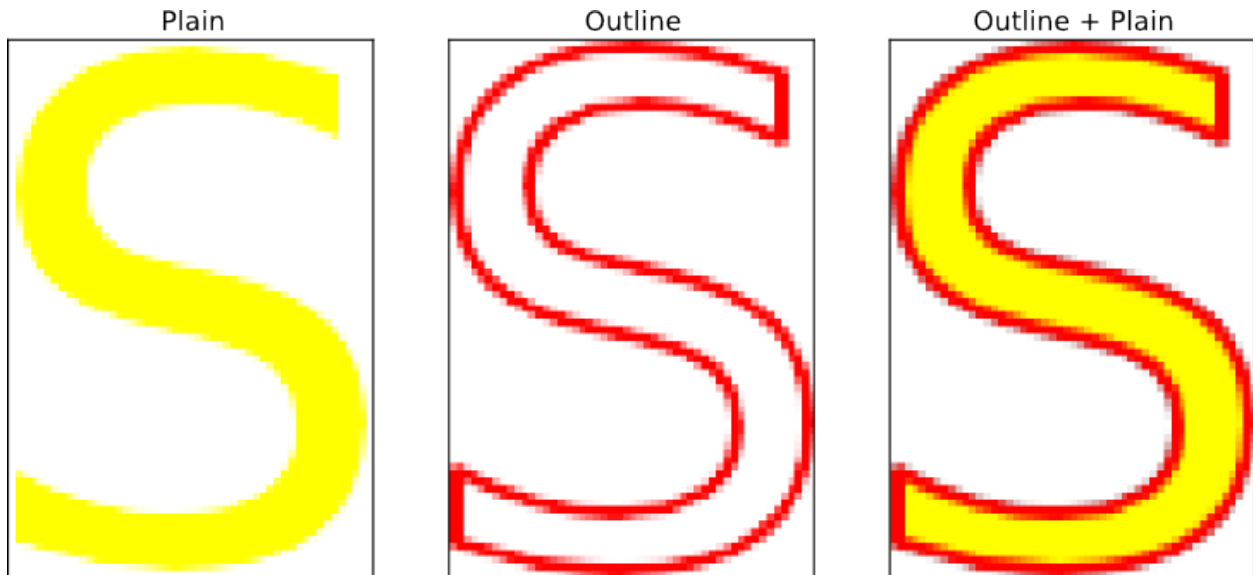


Screenshots below comes from the `glyph-vector.py` and `glyph-vectopr-2.py` examples showing how to access a glyph outline information and use it to draw the glyph. Rendering (with Bézier curves) is done using `matplotlib`.





Screenshot below comes from the `glyph-color.py` showing how to draw and combine a glyph outline with the regular glyph.



The screenshot below comes from the `hello-world.py` example showing how to draw text in a bitmap (that has been zoomed in to show antialiasing).



The screenshot below comes from the `agg-trick.py` example showing an implementation of ideas from the [Texts Rasterization Exposures](#) by Maxim Shemarev.

```
A Quick Brown Fox Jumps Over The Lazy Dog 0123456789  
A Quick Brown Fox Jumps Over The Lazy Dog 0123456789  
A Quick Brown Fox Jumps Over The Lazy Dog 0123456789  
A Quick Brown Fox Jumps Over The Lazy Dog 0123456789  
A Quick Brown Fox Jumps Over The Lazy Dog 0123456789  
A Quick Brown Fox Jumps Over The Lazy Dog 0123456789  
A Quick Brown Fox Jumps Over The Lazy Dog 0123456789  
A Quick Brown Fox Jumps Over The Lazy Dog 0123456789  
A Quick Brown Fox Jumps Over The Lazy Dog 0123456789  
A Quick Brown Fox Jumps Over The Lazy Dog 0123456789  
A Quick Brown Fox Jumps Over The Lazy Dog 0123456789  
A Quick Brown Fox Jumps Over The Lazy Dog 0123456789  
A Quick Brown Fox Jumps Over The Lazy Dog 0123456789
```

Face

class freetype.**Face** (*filename*, *index=0*)
FT_Face wrapper

FreeType root face class structure. A face object models a typeface in a font file.

ascender

The typographic ascender of the face, expressed in font units. For font formats not having this information, it is set to 'bbox.yMax'. Only relevant for scalable formats.

attach_file (*filename*)

Attach data to a face object. Normally, this is used to read additional information for the face object. For example, you can attach an AFM file that comes with a Type 1 font to get the kerning values and other metrics.

Parameters filename – Filename to attach

Note

The meaning of the 'attach' (i.e., what really happens when the new file is read) is not fixed by FreeType itself. It really depends on the font format (and thus the font driver).

Client applications are expected to know what they are doing when invoking this function. Most drivers simply do not implement file attachments.

available_sizes

A list of FT_Bitmap_Size for all bitmap strikes in the face. It is set to NULL if there is no bitmap strike.

bbox

The font bounding box. Coordinates are expressed in font units (see 'units_per_EM'). The box is large enough to contain any glyph from the font. Thus, 'bbox.yMax' can be seen as the 'maximal ascender', and 'bbox.yMin' as the 'minimal descender'. Only relevant for scalable formats.

Note that the bounding box might be off by (at least) one pixel for hinted fonts. See FT_Size_Metrics for further discussion.

charmap

The current active charmap for this face.

charmaps

A list of the charmaps of the face.

descender

The typographic descender of the face, expressed in font units. For font formats not having this information, it is set to 'bbox.yMin'. Note that this field is usually negative. Only relevant for scalable formats.

face_flags

A set of bit flags that give important information about the face; see FT_FACE_FLAG_XXX for the details.

face_index

The index of the face in the font file. It is set to 0 if there is only one face in the font file.

family_name

The face's family name. This is an ASCII string, usually in English, which describes the typeface's family (like 'Times New Roman', 'Bodoni', 'Garamond', etc). This is a least common denominator used to list fonts. Some formats (TrueType & OpenType) provide localized and Unicode versions of this string. Applications should use the format specific interface to access them. Can be NULL (e.g., in fonts embedded in a PDF file).

get_advance (*gindex, flags*)

Retrieve the advance value of a given glyph outline in an FT_Face. By default, the unhinted advance is returned in font units.

Parameters

- **gindex** – The glyph index.
- **flags** – A set of bit flags similar to those used when calling FT_Load_Glyph, used to determine what kind of advances you need.

Returns

The advance value, in either font units or 16.16 format.

If FT_LOAD_VERTICAL_LAYOUT is set, this is the vertical advance corresponding to a vertical layout. Otherwise, it is the horizontal advance in a horizontal layout.

get_char_index (*charcode*)

Return the glyph index of a given character code. This function uses a charmap object to do the mapping.

Parameters **charcode** – The character code.

Note:

If you use FreeType to manipulate the contents of font files directly, be aware that the glyph index returned by this function doesn't always correspond to the internal indices used within the file. This is done to ensure that value 0 always corresponds to the 'missing glyph'.

get_chars ()

This generator function is used to return all unicode character codes in the current charmap of a given face. For each character it also returns the corresponding glyph index.

Returns character code, glyph index

Note: Note that 'agindex' is set to 0 if the charmap is empty. The character code itself can be 0 in two cases: if the charmap is empty or if the value 0 is the first valid character code.

get_first_char ()

This function is used to return the first character code in the current charmap of a given face. It also returns the corresponding glyph index.

Returns Glyph index of first character code. 0 if charmap is empty.

Note:

You should use this function with `get_next_char` to be able to parse all character codes available in a given charmap. The code should look like this:

Note that ‘agindex’ is set to 0 if the charmap is empty. The result itself can be 0 in two cases: if the charmap is empty or if the value 0 is the first valid character code.

get_format ()

Return a string describing the format of a given face, using values which can be used as an X11 FONT_PROPERTY. Possible values are ‘TrueType’, ‘Type 1’, ‘BDF’, ‘PCF’, ‘Type 42’, ‘CID Type 1’, ‘CFF’, ‘PFR’, and ‘Windows FNT’.

get_fstype ()

Return the fsType flags for a font (embedding permissions).

The return value is a tuple containing the freetype enum name as a string and the actual flag as an int

get_glyph_name (*agindex, buffer_max=64*)

This function is used to return the glyph name for the given charcode.

Parameters

- **agindex** – The glyph index.
- **buffer_max** – The maximum number of bytes to use to store the glyph name.
- **glyph_name** – The glyph name, possibly truncated.

get_kerning (*left, right, mode=0*)

Return the kerning vector between two glyphs of a same face.

Parameters

- **left** – The index of the left glyph in the kern pair.
- **right** – The index of the right glyph in the kern pair.
- **mode** – See FT_Kerning_Mode for more information. Determines the scale and dimension of the returned kerning vector.

Note:

Only horizontal layouts (left-to-right & right-to-left) are supported by this method. Other layouts, or more sophisticated kernings, are out of the scope of this API function – they can be implemented through format-specific interfaces.

get_name_index (*name*)

Return the glyph index of a given glyph name. This function uses driver specific objects to do the translation.

Parameters **name** – The glyph name.

get_next_char (*charcode, agindex*)

This function is used to return the next character code in the current charmap of a given face following the value ‘charcode’, as well as the corresponding glyph index.

Parameters

- **charcode** – The starting character code.

- **agindex** – Glyph index of next character code. 0 if charmap is empty.

Note:

You should use this function with `FT_Get_First_Char` to walk over all character codes available in a given charmap. See the note for this function for a simple code example.

Note that ‘agindex’ is set to 0 when there are no more codes in the charmap.

get_sfnt_name (*index*)

Retrieve a string of the SFNT ‘name’ table for a given index

Parameters **index** – The index of the ‘name’ string.

Note:

The ‘string’ array returned in the ‘aname’ structure is not null-terminated. The application should deallocate it if it is no longer in use.

Use `FT_Get_Sfnt_Name_Count` to get the total number of available ‘name’ table entries, then do a loop until you get the right platform, encoding, and name ID.

glyph

The face’s associated glyph slot(s).

has_fixed_sizes

True whenever a face object contains some embedded bitmaps. See the ‘available_sizes’ field of the `FT_FaceRec` structure.

has_glyph_names

True whenever a face object contains some glyph names that can be accessed through `FT_Get_Glyph_Name`.

has_horizontal

True whenever a face object contains horizontal metrics (this is true for all font formats though).

has_kerning

True whenever a face object contains kerning data that can be accessed with `FT_Get_Kerning`.

has_multiple_masters

True whenever a face object contains some multiple masters. The functions provided by `FT_MULTIPLE_MASTERS_H` are then available to choose the exact design you want.

has_vertical

True whenever a face object contains vertical metrics.

height

The height is the vertical distance between two consecutive baselines, expressed in font units. It is always positive. Only relevant for scalable formats.

is_cid_keyed

True whenever a face object contains a CID-keyed font. See the discussion of `FT_FACE_FLAG_CID_KEYED` for more details.

If this macro is true, all functions defined in `FT_CID_H` are available.

is_fixed_width

True whenever a face object contains a font face that contains fixed-width (or ‘monospace’, ‘fixed-pitch’, etc.) glyphs.

is_scalable

true whenever a face object contains a scalable font face (true for TrueType, Type 1, Type 42, CID, OpenType/CFE, and PFR font formats).

is_sfnt

true whenever a face object contains a font whose format is based on the SFNT storage scheme. This usually means: TrueType fonts, OpenType fonts, as well as SFNT-based embedded bitmap fonts.

If this macro is true, all functions defined in FT_SFNT_NAMES_H and FT_TRUETYPE_TABLES_H are available.

is_tricky

True whenever a face represents a ‘tricky’ font. See the discussion of FT_FACE_FLAG_TRICKY for more details.

load_char (*char, flags=4*)

A function used to load a single glyph into the glyph slot of a face object, according to its character code.

Parameters

- **char** – The glyph’s character code, according to the current charmap used in the face.
- **flags** – A flag indicating what to load for this glyph. The FT_LOAD_XXX constants can be used to control the glyph loading process (e.g., whether the outline should be scaled, whether to load bitmaps or not, whether to hint the outline, etc).

Note:

This function simply calls FT_Get_Char_Index and FT_Load_Glyph.

load_glyph (*index, flags=4*)

A function used to load a single glyph into the glyph slot of a face object.

Parameters

- **index** – The index of the glyph in the font file. For CID-keyed fonts (either in PS or in CFF format) this argument specifies the CID value.
- **flags** – A flag indicating what to load for this glyph. The FT_LOAD_XXX constants can be used to control the glyph loading process (e.g., whether the outline should be scaled, whether to load bitmaps or not, whether to hint the outline, etc).

Note:

The loaded glyph may be transformed. See FT_Set_Transform for the details.

For subsetted CID-keyed fonts, ‘FT_Err_Invalid_Argument’ is returned for invalid CID values (this is, for CID values which don’t have a corresponding glyph in the font). See the discussion of the FT_FACE_FLAG_CID_KEYED flag for more details.

max_advance_height

The maximal advance height, in font units, for all glyphs in this face. This is only relevant for vertical layouts, and is set to ‘height’ for fonts that do not provide vertical metrics. Only relevant for scalable formats.

max_advance_width

The maximal advance width, in font units, for all glyphs in this face. This can be used to make word wrapping computations faster. Only relevant for scalable formats.

num_faces

The number of faces in the font file. Some font formats can have multiple faces in a font file.

num_fixed_sizes

The number of bitmap strikes in the face. Even if the face is scalable, there might still be bitmap strikes, which are called ‘sbits’ in that case.

num_glyphs

The number of glyphs in the face. If the face is scalable and has sbits (see ‘num_fixed_sizes’), it is set to the number of outline glyphs.

For CID-keyed fonts, this value gives the highest CID used in the font.

postscript_name

ASCII PostScript name of face, if available. This only works with PostScript and TrueType fonts.

select_charmap (*encoding*)

Select a given charmap by its encoding tag (as listed in ‘freetype.h’).

Note:

This function returns an error if no charmap in the face corresponds to the encoding queried here.

Because many fonts contain more than a single cmap for Unicode encoding, this function has some special code to select the one which covers Unicode best (‘best’ in the sense that a UCS-4 cmap is preferred to a UCS-2 cmap). It is thus preferable to FT_Set_Charmap in this case.

select_size (*strike_index*)

Select a bitmap strike.

Parameters **strike_index** – The index of the bitmap strike in the ‘available_sizes’ field of Face object.

set_char_size (*width=0, height=0, hres=72, vres=72*)

This function calls FT_Request_Size to request the nominal size (in points).

Parameters

- **width** (*float*) – The nominal width, in 26.6 fractional points.
- **height** (*float*) – The nominal height, in 26.6 fractional points.
- **hres** (*float*) – The horizontal resolution in dpi.
- **vres** (*float*) – The vertical resolution in dpi.

Note

If either the character width or height is zero, it is set equal to the other value.

If either the horizontal or vertical resolution is zero, it is set equal to the other value.

A character width or height smaller than 1pt is set to 1pt; if both resolution values are zero, they are set to 72dpi.

Don’t use this function if you are using the FreeType cache API.

set_charmap (*charmap*)

Select a given charmap for character code to glyph index mapping.

Parameters **charmap** – A handle to the selected charmap, or an index to face->charmaps[]

set_pixel_sizes (*width, height*)

This function calls FT_Request_Size to request the nominal size (in pixels).

Parameters

- **width** – The nominal width, in pixels.
- **height** – The nominal height, in pixels.

set_transform (*matrix, delta*)

A function used to set the transformation that is applied to glyph images when they are loaded into a glyph slot through FT_Load_Glyph.

Parameters `matrix` – A pointer to the transformation’s 2x2 matrix. Use 0 for the identity matrix.

Parm `delta` A pointer to the translation vector. Use 0 for the null vector.

Note:

The transformation is only applied to scalable image formats after the glyph has been loaded. It means that hinting is unaltered by the transformation and is performed on the character size given in the last call to `FT_Set_Char_Size` or `FT_Set_Pixel_Sizes`.

Note that this also transforms the `‘face.glyph.advance’` field, but not the values in `‘face.glyph.metrics’`.

`sfnt_name_count`

Number of name strings in the SFNT ‘name’ table.

`size`

The current active size for this face.

`style_flags`

A set of bit flags indicating the style of the face; see `FT_STYLE_FLAG_XXX` for the details.

`style_name`

The face’s style name. This is an ASCII string, usually in English, which describes the typeface’s style (like ‘Italic’, ‘Bold’, ‘Condensed’, etc). Not all font formats provide a style name, so this field is optional, and can be set to NULL. As for `‘family_name’`, some formats provide localized and Unicode versions of this string. Applications should use the format specific interface to access them.

`underline_position`

The position, in font units, of the underline line for this face. It is the center of the underlining stem. Only relevant for scalable formats.

`underline_thickness`

The thickness, in font units, of the underline for this face. Only relevant for scalable formats.

`units_per_EM`

The number of font units per EM square for this face. This is typically 2048 for TrueType fonts, and 1000 for Type 1 fonts. Only relevant for scalable formats.

BBox

class `freetype.BBox` (*bbox*)

FT_BBox wrapper.

A structure used to hold an outline’s bounding box, i.e., the coordinates of its extrema in the horizontal and vertical directions.

Note

The bounding box is specified with the coordinates of the lower left and the upper right corner. In PostScript, those values are often called (llx, lly) and (urx, ury), respectively.

If `‘yMin’` is negative, this value gives the glyph’s descender. Otherwise, the glyph doesn’t descend below the baseline. Similarly, if `‘ymax’` is positive, this value gives the glyph’s ascender.

`‘xMin’` gives the horizontal distance from the glyph’s origin to the left edge of the glyph’s bounding box. If `‘xMin’` is negative, the glyph extends to the left of the origin.

`xMax`

The horizontal maximum (right-most).

xMin
The horizontal minimum (left-most).

yMax
The vertical maximum (top-most).

yMin
The vertical minimum (bottom-most).

Size Metrics

class freetype.**SizeMetrics** (*metrics*)
The size metrics structure gives the metrics of a size object.

Note

The scaling values, if relevant, are determined first during a size changing operation. The remaining fields are then set by the driver. For scalable formats, they are usually set to scaled values of the corresponding fields in Face.

Note that due to glyph hinting, these values might not be exact for certain fonts. Thus they must be treated as unreliable with an error margin of at least one pixel!

Indeed, the only way to get the exact metrics is to render all glyphs. As this would be a definite performance hit, it is up to client applications to perform such computations.

The SizeMetrics structure is valid for bitmap fonts also.

ascender
The ascender in 26.6 fractional pixels. See Face for the details.

descender
The descender in 26.6 fractional pixels. See Face for the details.

height
The height in 26.6 fractional pixels. See Face for the details.

max_advance
The maximal advance width in 26.6 fractional pixels. See Face for the details.

x_ppem
The width of the scaled EM square in pixels, hence the term ‘ppem’ (pixels per EM). It is also referred to as ‘nominal width’.

x_scale
A 16.16 fractional scaling value used to convert horizontal metrics from font units to 26.6 fractional pixels. Only relevant for scalable font formats.

y_ppem
The height of the scaled EM square in pixels, hence the term ‘ppem’ (pixels per EM). It is also referred to as ‘nominal height’.

y_scale
A 16.16 fractional scaling value used to convert vertical metrics from font units to 26.6 fractional pixels. Only relevant for scalable font formats.

Bitmap size

class `freetype.BitmapSize` (*size*)
 FT_Bitmap_Size wrapper

This structure models the metrics of a bitmap strike (i.e., a set of glyphs for a given point size and resolution) in a bitmap font. It is used for the ‘available_sizes’ field of Face.

Note

Windows FNT: The nominal size given in a FNT font is not reliable. Thus when the driver finds it incorrect, it sets ‘size’ to some calculated values and sets ‘x_ppem’ and ‘y_ppem’ to the pixel width and height given in the font, respectively.

TrueType embedded bitmaps: ‘size’, ‘width’, and ‘height’ values are not contained in the bitmap strike itself. They are computed from the global font parameters.

height

The vertical distance, in pixels, between two consecutive baselines. It is always positive.

size

The nominal size of the strike in 26.6 fractional points. This field is not very useful.

width

The average width, in pixels, of all glyphs in the strike.

x_ppem

The horizontal ppem (nominal width) in 26.6 fractional pixels.

y_ppem

The vertical ppem (nominal width) in 26.6 fractional pixels.

Bitmap

class `freetype.Bitmap` (*bitmap*)
 FT_Bitmap wrapper

A structure used to describe a bitmap or pixmap to the raster. Note that we now manage pixmaps of various depths through the ‘pixel_mode’ field.

Note:

For now, the only pixel modes supported by FreeType are mono and grays. However, drivers might be added in the future to support more ‘colorful’ options.

buffer

A typeless pointer to the bitmap buffer. This value should be aligned on 32-bit boundaries in most cases.

num_grays

This field is only used with FT_PIXEL_MODE_GRAY; it gives the number of gray levels used in the bitmap.

palette

A typeless pointer to the bitmap palette; this field is intended for paletted pixel modes. Not used currently.

palette_mode

This field is intended for paletted pixel modes; it indicates how the palette is stored. Not used currently.

pitch

The pitch's absolute value is the number of bytes taken by one bitmap row, including padding. However, the pitch is positive when the bitmap has a 'down' flow, and negative when it has an 'up' flow. In all cases, the pitch is an offset to add to a bitmap pointer in order to go down one row.

Note that 'padding' means the alignment of a bitmap to a byte border, and FreeType functions normally align to the smallest possible integer value.

For the B/W rasterizer, 'pitch' is always an even number.

To change the pitch of a bitmap (say, to make it a multiple of 4), use `FT_Bitmap_Convert`. Alternatively, you might use callback functions to directly render to the application's surface; see the file 'example2.py' in the tutorial for a demonstration.

pixel_mode

The pixel mode, i.e., how pixel bits are stored. See `FT_Pixel_Mode` for possible values.

rows

The number of bitmap rows.

width

The number of pixels in bitmap row.

Charmap

class `freetype.Charmap` (*charmap*)

`FT_Charmap` wrapper.

A handle to a given character map. A charmap is used to translate character codes in a given encoding into glyph indexes for its parent's face. Some font formats may provide several charmaps per font.

Each face object owns zero or more charmaps, but only one of them can be 'active' and used by `FT_Get_Char_Index` or `FT_Load_Char`.

The list of available charmaps in a face is available through the 'face.num_charmaps' and 'face.charmaps' fields of `FT_FaceRec`.

The currently active charmap is available as 'face.charmap'. You should call `FT_Set_Charmap` to change it.

Note:

When a new face is created (either through `FT_New_Face` or `FT_Open_Face`), the library looks for a Unicode charmap within the list and automatically activates it.

See also:

See `FT_CharMapRec` for the publicly accessible fields of a given character map.

cmap_format

The format of 'charmap'. If 'charmap' doesn't belong to a TrueType/sfnt face, return -1.

cmap_language_id

The language ID of 'charmap'. If 'charmap' doesn't belong to a TrueType/sfnt face, just return 0 as the default value.

encoding

An `FT_Encoding` tag identifying the charmap. Use this with `FT_Select_Charmap`.

encoding_id

A platform specific encoding number. This also comes from the TrueType specification and should be emulated similarly.

encoding_name

A platform specific encoding name. This also comes from the TrueType specification and should be emulated similarly.

index

The index into the array of character maps within the face to which ‘charmap’ belongs. If an error occurs, -1 is returned.

platform_id

An ID number describing the platform for the following encoding ID. This comes directly from the TrueType specification and should be emulated for other formats.

Outline

class `freetype.Outline` (*outline*)

FT_Outline wrapper.

This structure is used to describe an outline to the scan-line converter.

contours

The number of contours in the outline.

flags

A set of bit flags used to characterize the outline and give hints to the scan-converter and hinter on how to convert/grid-fit it. See FT_OUTLINE_FLAGS.

get_bbox ()

Compute the exact bounding box of an outline. This is slower than computing the control box. However, it uses an advanced algorithm which returns very quickly when the two boxes coincide. Otherwise, the outline Bezier arcs are traversed to extract their extrema.

get_cbox ()

Return an outline’s ‘control box’. The control box encloses all the outline’s points, including Bezier control points. Though it coincides with the exact bounding box for most glyphs, it can be slightly larger in some situations (like when rotating an outline which contains Bezier outside arcs).

Computing the control box is very fast, while getting the bounding box can take much more time as it needs to walk over all segments and arcs in the outline. To get the latter, you can use the ‘ftbbox’ component which is dedicated to this single task.

get_inside_border ()

Retrieve the FT_StrokerBorder value corresponding to the ‘inside’ borders of a given outline.

Returns The border index. FT_STROKER_BORDER_RIGHT for empty or invalid outlines.

get_outside_border ()

Retrieve the FT_StrokerBorder value corresponding to the ‘outside’ borders of a given outline.

Returns The border index. FT_STROKER_BORDER_RIGHT for empty or invalid outlines.

points

The number of points in the outline.

tags

A list of ‘n_points’ chars, giving each outline point’s type.

If bit 0 is unset, the point is ‘off’ the curve, i.e., a Bezier control point, while it is ‘on’ if set.

Bit 1 is meaningful for ‘off’ points only. If set, it indicates a third-order Bezier arc control point; and a second-order control point if unset.

If bit 2 is set, bits 5-7 contain the drop-out mode (as defined in the OpenType specification; the value is the same as the argument to the SCANMODE instruction).

Bits 3 and 4 are reserved for internal purposes.

Glyph

class `freetype.Glyph` (*glyph*)
 FT_Glyph wrapper.

The root glyph structure contains a given glyph image plus its advance width in 16.16 fixed float format.

format

The format of the glyph's image.

get_cbox (*bbox_mode*)

Return an outline's 'control box'. The control box encloses all the outline's points, including Bezier control points. Though it coincides with the exact bounding box for most glyphs, it can be slightly larger in some situations (like when rotating an outline which contains Bezier outside arcs).

Computing the control box is very fast, while getting the bounding box can take much more time as it needs to walk over all segments and arcs in the outline. To get the latter, you can use the 'ftbbox' component which is dedicated to this single task.

Parameters mode – The mode which indicates how to interpret the returned bounding box values.

Note:

Coordinates are relative to the glyph origin, using the y upwards convention.

If the glyph has been loaded with FT_LOAD_NO_SCALE, 'bbox_mode' must be set to FT_GLYPH_BBOX_UNSCALED to get unscaled font units in 26.6 pixel format. The value FT_GLYPH_BBOX_SUBPIXELS is another name for this constant.

Note that the maximum coordinates are exclusive, which means that one can compute the width and height of the glyph image (be it in integer or 26.6 pixels) as:

`width = bbox.xMax - bbox.xMin; height = bbox.yMax - bbox.yMin;`

Note also that for 26.6 coordinates, if 'bbox_mode' is set to FT_GLYPH_BBOX_GRIDFIT, the coordinates will also be grid-fitted, which corresponds to:

`bbox.xMin = FLOOR(bbox.xMin); bbox.yMin = FLOOR(bbox.yMin); bbox.xMax = CEILING(bbox.xMax); bbox.yMax = CEILING(bbox.yMax);`

To get the bbox in pixel coordinates, set 'bbox_mode' to FT_GLYPH_BBOX_TRUNCATE.

To get the bbox in grid-fitted pixel coordinates, set 'bbox_mode' to FT_GLYPH_BBOX_PIXELS.

stroke (*stroker, destroy=False*)

Stroke a given outline glyph object with a given stroker.

Parameters

- **stroker** – A stroker handle.
- **destroy** – A Boolean. If 1, the source glyph object is destroyed on success.

Note:

The source glyph is untouched in case of error.

to_bitmap (*mode, origin, destroy=False*)
 Convert a given glyph object to a bitmap glyph object.

Parameters

- **mode** – An enumeration that describes how the data is rendered.
- **origin** – A pointer to a vector used to translate the glyph image before rendering. Can be 0 (if no translation). The origin is expressed in 26.6 pixels.
 We also detect a plain vector and make a pointer out of it, if that’s the case.
- **destroy** – A boolean that indicates that the original glyph image should be destroyed by this function. It is never destroyed in case of error.

Note:

This function does nothing if the glyph format isn’t scalable.
 The glyph image is translated with the ‘origin’ vector before rendering.
 The first parameter is a pointer to an FT_Glyph handle, that will be replaced by this function (with newly allocated data). Typically, you would use (omitting error handling):

Bitmap glyph

class freetype.**BitmapGlyph** (*glyph*)
 FT_BitmapGlyph wrapper.

A structure used for bitmap glyph images. This really is a ‘sub-class’ of FT_GlyphRec.

bitmap

A descriptor for the bitmap.

format

The format of the glyph’s image.

left

The left-side bearing, i.e., the horizontal distance from the current pen position to the left border of the glyph bitmap.

top

The top-side bearing, i.e., the vertical distance from the current pen position to the top border of the glyph bitmap. This distance is positive for upwards y!

Glyph slot

class freetype.**GlyphSlot** (*slot*)
 FT_GlyphSlot wrapper.

FreeType root glyph slot class structure. A glyph slot is a container where individual glyphs can be loaded, be they in outline or bitmap format.

advance

This shorthand is, depending on FT_LOAD_IGNORE_TRANSFORM, the transformed advance width for the glyph (in 26.6 fractional pixel format). As specified with FT_LOAD_VERTICAL_LAYOUT, it uses either the ‘horiAdvance’ or the ‘vertAdvance’ value of ‘metrics’ field.

bitmap

This field is used as a bitmap descriptor when the slot format is `FT_GLYPH_FORMAT_BITMAP`. Note that the address and content of the bitmap buffer can change between calls of `FT_Load_Glyph` and a few other functions.

bitmap_left

This is the bitmap's left bearing expressed in integer pixels. Of course, this is only valid if the format is `FT_GLYPH_FORMAT_BITMAP`.

bitmap_top

This is the bitmap's top bearing expressed in integer pixels. Remember that this is the distance from the baseline to the top-most glyph scanline, upwards y coordinates being positive.

format

This field indicates the format of the image contained in the glyph slot. Typically `FT_GLYPH_FORMAT_BITMAP`, `FT_GLYPH_FORMAT_OUTLINE`, or `FT_GLYPH_FORMAT_COMPOSITE`, but others are possible.

get_glyph()

A function used to extract a glyph image from a slot. Note that the created `FT_Glyph` object must be released with `FT_Done_Glyph`.

linearHoriAdvance

The advance width of the unhinted glyph. Its value is expressed in 16.16 fractional pixels, unless `FT_LOAD_LINEAR_DESIGN` is set when loading the glyph. This field can be important to perform correct WYSIWYG layout. Only relevant for outline glyphs.

linearVertAdvance

The advance height of the unhinted glyph. Its value is expressed in 16.16 fractional pixels, unless `FT_LOAD_LINEAR_DESIGN` is set when loading the glyph. This field can be important to perform correct WYSIWYG layout. Only relevant for outline glyphs.

metrics

The metrics of the last loaded glyph in the slot. The returned values depend on the last load flags (see the `FT_Load_Glyph` API function) and can be expressed either in 26.6 fractional pixels or font units. Note that even when the glyph image is transformed, the metrics are not.

next

In some cases (like some font tools), several glyph slots per face object can be a good thing. As this is rare, the glyph slots are listed through a direct, single-linked list using its 'next' field.

outline

The outline descriptor for the current glyph image if its format is `FT_GLYPH_FORMAT_OUTLINE`. Once a glyph is loaded, 'outline' can be transformed, distorted, embolded, etc. However, it must not be freed.

SFNT name

class `freetype.SfntName` (*name*)

SfntName wrapper

A structure used to model an SFNT 'name' table entry.

encoding_id

The encoding ID for 'string'.

language_id

The language ID for 'string'.

name_id

An identifier for ‘string’.

platform_id

The platform ID for ‘string’.

string

The ‘name’ string. Note that its format differs depending on the (platform,encoding) pair. It can be a Pascal String, a UTF-16 one, etc.

Generally speaking, the string is not zero-terminated. Please refer to the TrueType specification for details.

string_len

The length of ‘string’ in bytes.

Stroker

class `freetype.Stroker`

FT_Stroker wrapper

This component generates stroked outlines of a given vectorial glyph. It also allows you to retrieve the ‘outside’ and/or the ‘inside’ borders of the stroke.

This can be useful to generate ‘bordered’ glyph, i.e., glyphs displayed with a coloured (and anti-aliased) border around their shape.

begin_subpath (*to*, *_open*)

Start a new sub-path in the stroker.

:param *to* A pointer to the start vector.

Parameters *_open* – A boolean. If 1, the sub-path is treated as an open one.

Note:

This function is useful when you need to stroke a path that is not stored as an ‘Outline’ object.

conic_to (*control*, *to*)

‘Draw’ a single quadratic Bezier in the stroker’s current sub-path, from the last position.

Parameters

- **control** – A pointer to a Bezier control point.
- **to** – A pointer to the destination point.

Note:

You should call this function between ‘begin_subpath’ and ‘end_subpath’.

cubic_to (*control1*, *control2*, *to*)

‘Draw’ a single quadratic Bezier in the stroker’s current sub-path, from the last position.

Parameters

- **control1** – A pointer to the first Bezier control point.
- **control2** – A pointer to second Bezier control point.
- **to** – A pointer to the destination point.

Note:

You should call this function between ‘begin_subpath’ and ‘end_subpath’.

end_subpath ()

Close the current sub-path in the stroker.

Note:

You should call this function after 'begin_subpath'. If the subpath was not 'opened', this function 'draws' a single line segment to the start position when needed.

export (*outline*)

Call this function after `get_border_counts` to export all borders to your own 'Outline' structure.

Note that this function appends the border points and contours to your outline, but does not try to resize its arrays.

Parameters `outline` – The target outline.

export_border (*border, outline*)

Call this function after 'get_border_counts' to export the corresponding border to your own 'Outline' structure.

Note that this function appends the border points and contours to your outline, but does not try to resize its arrays.

Parameters

- **border** – The border index.
- **outline** – The target outline.

Note:

Always call this function after `get_border_counts` to get sure that there is enough room in your 'Outline' object to receive all new data.

When an outline, or a sub-path, is 'closed', the stroker generates two independent 'border' outlines, named 'left' and 'right'

When the outline, or a sub-path, is 'opened', the stroker merges the 'border' outlines with caps. The 'left' border receives all points, while the 'right' border becomes empty.

Use the function `export` instead if you want to retrieve all borders at once.

get_border_counts (*border*)

Call this function once you have finished parsing your paths with the stroker. It returns the number of points and contours necessary to export one of the 'border' or 'stroke' outlines generated by the stroker.

Parameters `border` – The border index.

Returns number of points, number of contours

get_counts ()

Call this function once you have finished parsing your paths with the stroker. It returns the number of points and contours necessary to export all points/borders from the stroked outline/path.

Returns number of points, number of contours

line_to (*to*)

'Draw' a single line segment in the stroker's current sub-path, from the last position.

Parameters `to` – A pointer to the destination point.

Note:

You should call this function between 'begin_subpath' and 'end_subpath'.

parse_outline (*outline, opened*)

A convenience function used to parse a whole outline with the stroker. The resulting outline(s) can be retrieved later by functions like `FT_Stroker_GetCounts` and `FT_Stroker_Export`.

Parameters **outline** – The source outline.

Param opened A boolean. If 1, the outline is treated as an open path instead of a closed one.

Note:

If ‘opened’ is 0 (the default), the outline is treated as a closed path, and the stroker generates two distinct ‘border’ outlines.

If ‘opened’ is 1, the outline is processed as an open path, and the stroker generates a single ‘stroke’ outline.

This function calls ‘rewind’ automatically.

rewind ()

Reset a stroker object without changing its attributes. You should call this function before beginning a new series of calls to `FT_Stroker_BeginSubPath` or `FT_Stroker_EndSubPath`.

set (*radius, line_cap, line_join, miter_limit*)

Reset a stroker object’s attributes.

Parameters

- **radius** – The border radius.
- **line_cap** – The line cap style.
- **line_join** – The line join style.
- **miter_limit** – The miter limit for the `FT_STROKER_LINEJOIN_MITER` style, expressed as 16.16 fixed point value.

Note:

The radius is expressed in the same units as the outline coordinates.

Constants

FT_ENCODINGS

An enumeration used to specify character sets supported by charmaps. Used in the `FT_Select_Charmap` API function.

FT_ENCODING_NONE

The encoding value 0 is reserved.

FT_ENCODING_UNICODE

Corresponds to the Unicode character set. This value covers all versions of the Unicode repertoire, including ASCII and Latin-1. Most fonts include a Unicode charmap, but not all of them.

For example, if you want to access Unicode value U+1F028 (and the font contains it), use value 0x1F028 as the input value for `FT_Get_Char_Index`.

FT_ENCODING_MS_SYMBOL

Corresponds to the Microsoft Symbol encoding, used to encode mathematical symbols in the 32..255 character code range. For more information, see ‘<http://www.ceviz.net/symbol.htm>’.

FT_ENCODING_SJIS

Corresponds to Japanese SJIS encoding. More info at at '<http://langsupport.japanreference.com/encoding.shtml>'. See note on multi-byte encodings below.

FT_ENCODING_GB2312

Corresponds to an encoding system for Simplified Chinese as used used in mainland China.

FT_ENCODING_BIG5

Corresponds to an encoding system for Traditional Chinese as used in Taiwan and Hong Kong.

FT_ENCODING_WANSUNG

Corresponds to the Korean encoding system known as Wansung. For more information see '<http://www.microsoft.com/typography/unicode/949.txt>'.

FT_ENCODING_JOHAB

The Korean standard character set (KS C 5601-1992), which corresponds to MS Windows code page 1361. This character set includes all possible Hangeul character combinations.

FT_ENCODING_ADOBE_LATIN_1

Corresponds to a Latin-1 encoding as defined in a Type 1 PostScript font. It is limited to 256 character codes.

FT_ENCODING_ADOBE_STANDARD

Corresponds to the Adobe Standard encoding, as found in Type 1, CFF, and OpenType/CFF fonts. It is limited to 256 character codes.

FT_ENCODING_ADOBE_EXPERT

Corresponds to the Adobe Expert encoding, as found in Type 1, CFF, and OpenType/CFF fonts. It is limited to 256 character codes.

FT_ENCODING_ADOBE_CUSTOM

Corresponds to a custom encoding, as found in Type 1, CFF, and OpenType/CFF fonts. It is limited to 256 character codes.

FT_ENCODING_APPLE_ROMAN

Corresponds to the 8-bit Apple roman encoding. Many TrueType and OpenType fonts contain a charmap for this encoding, since older versions of Mac OS are able to use it.

FT_ENCODING_OLD_LATIN_2

This value is deprecated and was never used nor reported by FreeType. Don't use or test for it.

FT_FACE_FLAGS

A list of bit flags used in the 'face_flags' field of the FT_FaceRec structure. They inform client applications of properties of the corresponding face.

FT_FACE_FLAG_SCALABLE

Indicates that the face contains outline glyphs. This doesn't prevent bitmap strikes, i.e., a face can have both this and FT_FACE_FLAG_FIXED_SIZES set.

FT_FACE_FLAG_FIXED_SIZES

Indicates that the face contains bitmap strikes. See also the 'num_fixed_sizes' and 'available_sizes' fields of FT_FaceRec.

FT_FACE_FLAG_FIXED_WIDTH

Indicates that the face contains fixed-width characters (like Courier, Lucido, MonoType, etc.).

FT_FACE_FLAG_SFNT

Indicates that the face uses the 'sfnt' storage scheme. For now, this means TrueType and OpenType.

FT_FACE_FLAG_HORIZONTAL

Indicates that the face contains horizontal glyph metrics. This should be set for all common formats.

FT_FACE_FLAG_VERTICAL

Indicates that the face contains vertical glyph metrics. This is only available in some formats, not all of them.

FT_FACE_FLAG_KERNING

Indicates that the face contains kerning information. If set, the kerning distance can be retrieved through the function `FT_Get_Kerning`. Otherwise the function always return the vector (0,0). Note that FreeType doesn't handle kerning data from the 'GPOS' table (as present in some OpenType fonts).

FT_FACE_FLAG_MULTIPLE_MASTERS

Indicates that the font contains multiple masters and is capable of interpolating between them. See the multiple-masters specific API for details.

FT_FACE_FLAG_GLYPH_NAMES

Indicates that the font contains glyph names that can be retrieved through `FT_Get_Glyph_Name`. Note that some TrueType fonts contain broken glyph name tables. Use the function `FT_Has_PS_Glyph_Names` when needed.

FT_FACE_FLAG_EXTERNAL_STREAM

Used internally by FreeType to indicate that a face's stream was provided by the client application and should not be destroyed when `FT_Done_Face` is called. Don't read or test this flag.

FT_FACE_FLAG_HINTER

Set if the font driver has a hinting machine of its own. For example, with TrueType fonts, it makes sense to use data from the SFNT 'gasp' table only if the native TrueType hinting engine (with the bytecode interpreter) is available and active.

FT_FACE_FLAG_CID_KEYED

Set if the font is CID-keyed. In that case, the font is not accessed by glyph indices but by CID values. For subsetted CID-keyed fonts this has the consequence that not all index values are a valid argument to `FT_Load_Glyph`. Only the CID values for which corresponding glyphs in the subsetted font exist make `FT_Load_Glyph` return successfully; in all other cases you get an 'FT_Err_Invalid_Argument' error.

Note that CID-keyed fonts which are in an SFNT wrapper don't have this flag set since the glyphs are accessed in the normal way (using contiguous indices); the 'CID-ness' isn't visible to the application.

FT_FACE_FLAG_TRICKY

Set if the font is 'tricky', this is, it always needs the font format's native hinting engine to get a reasonable result. A typical example is the Chinese font 'mingli.ttf' which uses TrueType bytecode instructions to move and scale all of its subglyphs.

It is not possible to autohint such fonts using `FT_LOAD_FORCE_AUTOHINT`; it will also ignore `FT_LOAD_NO_HINTING`. You have to set both `FT_LOAD_NO_HINTING` and `FT_LOAD_NO_AUTOHINT` to really disable hinting; however, you probably never want this except for demonstration purposes.

Currently, there are six TrueType fonts in the list of tricky fonts; they are hard-coded in file 'ttobjs.c'.

FT_FSTYPES

A list of bit flags that inform client applications of embedding and subsetting restrictions associated with a font.

FT_FSTYPE_INSTALLABLE_EMBEDDING

Fonts with no fsType bit set may be embedded and permanently installed on the remote system by an application.

FT_FSTYPE_RESTRICTED_LICENSE_EMBEDDING

Fonts that have only this bit set must not be modified, embedded or exchanged in any manner without first obtaining permission of the font software copyright owner.

FT_FSTYPE_PREVIEW_AND_PRINT_EMBEDDING

If this bit is set, the font may be embedded and temporarily loaded on the remote system. Documents containing Preview & Print fonts must be opened 'read-only'; no edits can be applied to the document.

FT_FSTYPE_EDITABLE_EMBEDDING

If this bit is set, the font may be embedded but must only be installed temporarily on other systems. In contrast to Preview & Print fonts, documents containing editable fonts may be opened for reading, editing is permitted, and changes may be saved.

FT_FSTYPE_NO_SUBSETTING

If this bit is set, the font may not be subsetted prior to embedding.

FT_FSTYPE_BITMAP_EMBEDDING_ONLY

If this bit is set, only bitmaps contained in the font may be embedded; no outline data may be embedded. If there are no bitmaps available in the font, then the font is unembeddable.

FT_GLYPH_BBOX_MODES

The mode how the values of FT_Glyph_Get_CBox are returned.

FT_GLYPH_BBOX_UNSCALED

Return unscaled font units.

FT_GLYPH_BBOX_SUBPIXELS

Return unfitted 26.6 coordinates.

FT_GLYPH_BBOX_GRIDFIT

Return grid-fitted 26.6 coordinates.

FT_GLYPH_BBOX_TRUNCATE

Return coordinates in integer pixels.

FT_GLYPH_BBOX_PIXELS

Return grid-fitted pixel coordinates.

FT_GLYPH_FORMATS

An enumeration type used to describe the format of a given glyph image. Note that this version of FreeType only supports two image formats, even though future font drivers will be able to register their own format.

FT_GLYPH_FORMAT_NONE

The value 0 is reserved.

FT_GLYPH_FORMAT_COMPOSITE

The glyph image is a composite of several other images. This format is only used with FT_LOAD_NO_RECURSE, and is used to report compound glyphs (like accented characters).

FT_GLYPH_FORMAT_BITMAP

The glyph image is a bitmap, and can be described as an FT_Bitmap. You generally need to access the ‘bitmap’ field of the FT_GlyphSlotRec structure to read it.

FT_GLYPH_FORMAT_OUTLINE

The glyph image is a vectorial outline made of line segments and Bezier arcs; it can be described as an FT_Outline; you generally want to access the ‘outline’ field of the FT_GlyphSlotRec structure to read it.

FT_GLYPH_FORMAT_PLOTTER

The glyph image is a vectorial path with no inside and outside contours. Some Type 1 fonts, like those in the Hershey family, contain glyphs in this format. These are described as FT_Outline, but FreeType isn’t currently capable of rendering them correctly.

FT_KERNING_MODES

An enumeration used to specify which kerning values to return in `.. data:: FT_Get_Kerning`.

FT_KERNING_DEFAULT

Return scaled and grid-fitted kerning distances (value is 0).

FT_KERNING_UNFITTED

Return scaled but un-grid-fitted kerning distances.

FT_KERNING_UNSCALED

Return the kerning vector in original font units.

FT_LCD_FILTERS

A list of values to identify various types of LCD filters.

FT_LCD_FILTER_NONE

Do not perform filtering. When used with subpixel rendering, this results in sometimes severe color fringes.

FT_LCD_FILTER_DEFAULT

The default filter reduces color fringes considerably, at the cost of a slight blurriness in the output.

FT_LCD_FILTER_LIGHT

The light filter is a variant that produces less blurriness at the cost of slightly more color fringes than the default one. It might be better, depending on taste, your monitor, or your personal vision.

FT_LCD_FILTER_LEGACY

This filter corresponds to the original libXft color filter. It provides high contrast output but can exhibit really bad color fringes if glyphs are not extremely well hinted to the pixel grid. In other words, it only works well if the TrueType bytecode interpreter is enabled and high-quality hinted fonts are used.

This filter is only provided for comparison purposes, and might be disabled or stay unsupported in the future.

FT_LOAD_FLAGS

A list of bit-field constants used with `FT_Load_Glyph` to indicate what kind of operations to perform during glyph loading.

FT_LOAD_DEFAULT

Corresponding to 0, this value is used as the default glyph load operation. In this case, the following happens:

- 1.FreeType looks for a bitmap for the glyph corresponding to the face's current size. If one is found, the function returns. The bitmap data can be accessed from the glyph slot (see note below).
- 2.If no embedded bitmap is searched or found, FreeType looks for a scalable outline. If one is found, it is loaded from the font file, scaled to device pixels, then 'hinted' to the pixel grid in order to optimize it. The outline data can be accessed from the glyph slot (see note below).

Note that by default, the glyph loader doesn't render outlines into bitmaps. The following flags are used to modify this default behaviour to more specific and useful cases.

FT_LOAD_NO_SCALE

Don't scale the outline glyph loaded, but keep it in font units.

This flag implies `FT_LOAD_NO_HINTING` and `FT_LOAD_NO_BITMAP`, and unsets `FT_LOAD_RENDER`.

FT_LOAD_NO_HINTING

Disable hinting. This generally generates 'blurrier' bitmap glyph when the glyph is rendered in any of the anti-aliased modes. See also the note below.

This flag is implied by `FT_LOAD_NO_SCALE`.

`FT_LOAD_RENDER`

Call `FT_Render_Glyph` after the glyph is loaded. By default, the glyph is rendered in `FT_RENDER_MODE_NORMAL` mode. This can be overridden by `FT_LOAD_TARGET_XXX` or `FT_LOAD_MONOCHROME`.

This flag is unset by `FT_LOAD_NO_SCALE`.

`FT_LOAD_NO_BITMAP`

Ignore bitmap strikes when loading. Bitmap-only fonts ignore this flag.

`FT_LOAD_NO_SCALE` always sets this flag.

`FT_LOAD_VERTICAL_LAYOUT`

Load the glyph for vertical text layout. Don't use it as it is problematic currently.

`FT_LOAD_FORCE_AUTOHINT`

Indicates that the auto-hinter is preferred over the font's native hinter. See also the note below.

`FT_LOAD_CROP_BITMAP`

Indicates that the font driver should crop the loaded bitmap glyph (i.e., remove all space around its black bits). Not all drivers implement this.

`FT_LOAD_PEDANTIC`

Indicates that the font driver should perform pedantic verifications during glyph loading. This is mostly used to detect broken glyphs in fonts. By default, FreeType tries to handle broken fonts also.

`FT_LOAD_IGNORE_GLOBAL_ADVANCE_WIDTH`

Indicates that the font driver should ignore the global advance width defined in the font. By default, that value is used as the advance width for all glyphs when the face has `FT_FACE_FLAG_FIXED_WIDTH` set.

This flag exists for historical reasons (to support buggy CJK fonts).

`FT_LOAD_NO_RECURSE`

This flag is only used internally. It merely indicates that the font driver should not load composite glyphs recursively. Instead, it should set the 'num_subglyph' and 'subglyphs' values of the glyph slot accordingly, and set 'glyph->format' to `FT_GLYPH_FORMAT_COMPOSITE`.

The description of sub-glyphs is not available to client applications for now.

This flag implies `FT_LOAD_NO_SCALE` and `FT_LOAD_IGNORE_TRANSFORM`.

`FT_LOAD_IGNORE_TRANSFORM`

Indicates that the transform matrix set by `FT_Set_Transform` should be ignored.

`FT_LOAD_MONOCHROME`

This flag is used with `FT_LOAD_RENDER` to indicate that you want to render an outline glyph to a 1-bit monochrome bitmap glyph, with 8 pixels packed into each byte of the bitmap data.

Note that this has no effect on the hinting algorithm used. You should rather use `FT_LOAD_TARGET_MONO` so that the monochrome-optimized hinting algorithm is used.

`FT_LOAD_LINEAR_DESIGN`

Indicates that the 'linearHoriAdvance' and 'linearVertAdvance' fields of `FT_GlyphSlotRec` should be kept in font units. See `FT_GlyphSlotRec` for details.

`FT_LOAD_NO_AUTOHINT`

Disable auto-hinter. See also the note below.

FT_LOAD_TARGETS

A list of values that are used to select a specific hinting algorithm to use by the hinter. You should OR one of these values to your 'load_flags' when calling FT_Load_Glyph.

Note that font's native hinters may ignore the hinting algorithm you have specified (e.g., the TrueType bytecode interpreter). You can set .. data:: FT_LOAD_FORCE_AUTOHINT to ensure that the auto-hinter is used.

Also note that FT_LOAD_TARGET_LIGHT is an exception, in that it always implies FT_LOAD_FORCE_AUTOHINT.

FT_LOAD_TARGET_NORMAL

This corresponds to the default hinting algorithm, optimized for standard gray-level rendering. For monochrome output, use FT_LOAD_TARGET_MONO instead.

FT_LOAD_TARGET_LIGHT

A lighter hinting algorithm for non-monochrome modes. Many generated glyphs are more fuzzy but better resemble its original shape. A bit like rendering on Mac OS X.

As a special exception, this target implies FT_LOAD_FORCE_AUTOHINT.

FT_LOAD_TARGET_MONO

Strong hinting algorithm that should only be used for monochrome output. The result is probably unpleasant if the glyph is rendered in non-monochrome modes.

FT_LOAD_TARGET_LCD

A variant of FT_LOAD_TARGET_NORMAL optimized for horizontally decimated LCD displays.

FT_LOAD_TARGET_LCD_V

A variant of FT_LOAD_TARGET_NORMAL optimized for vertically decimated LCD displays.

FT_OPEN_MODES

A list of bit-field constants used within the 'flags' field of the .. data:: FT_Open_Args structure.

FT_OPEN_MEMORY

This is a memory-based stream.

FT_OPEN_STREAM

Copy the stream from the 'stream' field.

FT_OPEN_PATHNAME

Create a new input stream from a C path name.

FT_OPEN_DRIVER

Use the 'driver' field.

FT_OPEN_PARAMS

Use the 'num_params' and 'params' fields.

FT_OUTLINE_FLAGS

A list of bit-field constants use for the flags in an outline's 'flags' field.

FT_OUTLINE_NONE

Value 0 is reserved.

FT_OUTLINE_OWNER

If set, this flag indicates that the outline's field arrays (i.e., 'points', 'flags', and 'contours') are 'owned' by the outline object, and should thus be freed when it is destroyed.

FT_OUTLINE_EVEN_ODD_FILL

By default, outlines are filled using the non-zero winding rule. If set to 1, the outline will be filled using the even-odd fill rule (only works with the smooth rasterizer).

FT_OUTLINE_REVERSE_FILL

By default, outside contours of an outline are oriented in clock-wise direction, as defined in the TrueType specification. This flag is set if the outline uses the opposite direction (typically for Type 1 fonts). This flag is ignored by the scan converter.

FT_OUTLINE_IGNORE_DROPOUTS

By default, the scan converter will try to detect drop-outs in an outline and correct the glyph bitmap to ensure consistent shape continuity. If set, this flag hints the scan-line converter to ignore such cases. See below for more information.

FT_OUTLINE_SMART_DROPOUTS

Select smart dropout control. If unset, use simple dropout control. Ignored if FT_OUTLINE_IGNORE_DROPOUTS is set. See below for more information.

FT_OUTLINE_INCLUDE_STUBS

If set, turn pixels on for ‘stubs’, otherwise exclude them. Ignored if FT_OUTLINE_IGNORE_DROPOUTS is set. See below for more information.

FT_OUTLINE_HIGH_PRECISION

This flag indicates that the scan-line converter should try to convert this outline to bitmaps with the highest possible quality. It is typically set for small character sizes. Note that this is only a hint that might be completely ignored by a given scan-converter.

FT_OUTLINE_SINGLE_PASS

This flag is set to force a given scan-converter to only use a single pass over the outline to render a bitmap glyph image. Normally, it is set for very large character sizes. It is only a hint that might be completely ignored by a given scan-converter.

FT_PIXEL_MODES

An enumeration type that lists the render modes supported by FreeType 2. Each mode corresponds to a specific type of scanline conversion performed on the outline.

FT_PIXEL_MODE_NONE

Value 0 is reserved.

FT_PIXEL_MODE_MONO

A monochrome bitmap, using 1 bit per pixel. Note that pixels are stored in most-significant order (MSB), which means that the left-most pixel in a byte has value 128.

FT_PIXEL_MODE_GRAY

An 8-bit bitmap, generally used to represent anti-aliased glyph images. Each pixel is stored in one byte. Note that the number of ‘gray’ levels is stored in the ‘num_grays’ field of the FT_Bitmap structure (it generally is 256).

FT_PIXEL_MODE_GRAY2

A 2-bit per pixel bitmap, used to represent embedded anti-aliased bitmaps in font files according to the OpenType specification. We haven’t found a single font using this format, however.

FT_PIXEL_MODE_GRAY4

A 4-bit per pixel bitmap, representing embedded anti-aliased bitmaps in font files according to the OpenType specification. We haven’t found a single font using this format, however.

FT_PIXEL_MODE_LCD

An 8-bit bitmap, representing RGB or BGR decimated glyph images used for display on LCD displays; the bitmap is three times wider than the original glyph image. See also FT_RENDER_MODE_LCD.

FT_PIXEL_MODE_LCD_V

An 8-bit bitmap, representing RGB or BGR decimated glyph images used for display on rotated LCD displays; the bitmap is three times taller than the original glyph image. See also FT_RENDER_MODE_LCD_V.

FT_RENDER_MODES

An enumeration type that lists the render modes supported by FreeType 2. Each mode corresponds to a specific type of scanline conversion performed on the outline.

For bitmap fonts and embedded bitmaps the ‘bitmap->pixel_mode’ field in the .. data:: FT_GlyphSlotRec structure gives the format of the returned bitmap.

All modes except FT_RENDER_MODE_MONO use 256 levels of opacity.

FT_RENDER_MODE_NORMAL

This is the default render mode; it corresponds to 8-bit anti-aliased bitmaps.

FT_RENDER_MODE_LIGHT

This is equivalent to FT_RENDER_MODE_NORMAL. It is only defined as a separate value because render modes are also used indirectly to define hinting algorithm selectors. See FT_LOAD_TARGET_XXX for details.

FT_RENDER_MODE_MONO

This mode corresponds to 1-bit bitmaps (with 2 levels of opacity).

FT_RENDER_MODE_LCD

This mode corresponds to horizontal RGB and BGR sub-pixel displays like LCD screens. It produces 8-bit bitmaps that are 3 times the width of the original glyph outline in pixels, and which use the FT_PIXEL_MODE_LCD mode.

FT_RENDER_MODE_LCD_V

This mode corresponds to vertical RGB and BGR sub-pixel displays (like PDA screens, rotated LCD displays, etc.). It produces 8-bit bitmaps that are 3 times the height of the original glyph outline in pixels and use the FT_PIXEL_MODE_LCD_V mode.

FT_STROKER_BORDERS

These values are used to select a given stroke border in .. data:: FT_Stroker_GetBorderCounts and FT_Stroker_ExportBorder.

FT_STROKER_BORDER_LEFT

Select the left border, relative to the drawing direction.

FT_STROKER_BORDER_RIGHT

Select the right border, relative to the drawing direction.

Note

Applications are generally interested in the ‘inside’ and ‘outside’ borders. However, there is no direct mapping between these and the ‘left’ and ‘right’ ones, since this really depends on the glyph’s drawing orientation, which varies between font formats.

You can however use FT_Outline_GetInsideBorder and FT_Outline_GetOutsideBorder to get these.

FT_STROKER_LINECAPS

These values determine how the end of opened sub-paths are rendered in a stroke.

FT_STROKER_LINECAP_BUTT

The end of lines is rendered as a full stop on the last point itself.

FT_STROKER_LINECAP_ROUND

The end of lines is rendered as a half-circle around the last point.

FT_STROKER_LINECAP_SQUARE

The end of lines is rendered as a square around the last point.

FT_STROKER_LINEJOINS

These values determine how two joining lines are rendered in a stroker.

FT_STROKER_LINEJOIN_ROUND

Used to render rounded line joins. Circular arcs are used to join two lines smoothly.

FT_STROKER_LINEJOIN_BEVEL

Used to render beveled line joins; i.e., the two joining lines are extended until they intersect.

FT_STROKER_LINEJOIN_MITER

Same as beveled rendering, except that an additional line break is added if the angle between the two joining lines is too closed (this is useful to avoid unpleasant spikes in beveled rendering).

FT_STYLE_FLAGS

A list of bit-flags used to indicate the style of a given face. These are used in the 'style_flags' field of FT_FaceRec.

FT_STYLE_FLAG_ITALIC

Indicates that a given face style is italic or oblique.

FT_STYLE_FLAG_BOLD

Indicates that a given face is bold.

TT_ADOBE_IDS

A list of valid values for the 'encoding_id' for TT_PLATFORM_ADOBE charmaps. This is a FreeType-specific extension!

TT_ADOBE_ID_STANDARD

Adobe standard encoding.

TT_ADOBE_ID_EXPERT

Adobe expert encoding.

TT_ADOBE_ID_CUSTOM

Adobe custom encoding.

TT_ADOBE_ID_LATIN_1

Adobe Latin 1 encoding.

TT_APPLE_IDS

A list of valid values for the ‘encoding_id’ for TT_PLATFORM_APPLE_UNICODE charmaps and name entries.

TT_APPLE_ID_DEFAULT

Unicode version 1.0.

TT_APPLE_ID_UNICODE_1_1

Unicode 1.1; specifies Hangul characters starting at U+34xx.

TT_APPLE_ID_ISO_10646

Deprecated (identical to preceding).

TT_APPLE_ID_UNICODE_2_0

Unicode 2.0 and beyond (UTF-16 BMP only).

TT_APPLE_ID_UNICODE_3_2

Unicode 3.1 and beyond, using UTF-32.

TT_APPLE_ID_VARIANT_SELECTOR

From Adobe, not Apple. Not a normal cmap. Specifies variations on a real cmap.

TT_MAC_IDS

A list of valid values for the ‘encoding_id’ for TT_PLATFORM_MACINTOSH charmaps and name entries.

TT_MAC_ID_ROMAN

TT_MAC_ID_TELUGU

TT_MAC_ID_GURMUKHI

TT_MAC_ID_TIBETAN

TT_MAC_ID_SIMPLIFIED_CHINESE

TT_MAC_ID_SINDHI

TT_MAC_ID_SINHALESE

TT_MAC_ID_RUSSIAN

TT_MAC_ID_KANNADA

TT_MAC_ID_VIETNAMESE

TT_MAC_ID_MONGOLIAN

TT_MAC_ID_DEVANAGARI

TT_MAC_ID_HEBREW

TT_MAC_ID_TAMIL

TT_MAC_ID_THAI

TT_MAC_ID_BURMESE

TT_MAC_ID_MALDIVIAN

TT_MAC_ID_TRADITIONAL_CHINESE

TT_MAC_ID_JAPANESE

TT_MAC_ID_GREEK

`TT_MAC_ID_LAOTIAN`
`TT_MAC_ID_KHMER`
`TT_MAC_ID_UNINTERP`
`TT_MAC_ID_ORIYA`
`TT_MAC_ID_RSYMBOL`
`TT_MAC_ID_MALAYALAM`
`TT_MAC_ID_GEEZ`
`TT_MAC_ID_KOREAN`
`TT_MAC_ID_GUJARATI`
`TT_MAC_ID_BENGALI`
`TT_MAC_ID_ARABIC`
`TT_MAC_ID_GEORGIAN`
`TT_MAC_ID_ARMENIAN`
`TT_MAC_ID_SLAVIC`

TT_MAC_LANGIDS

Possible values of the language identifier field in the name records of the TTF ‘name’ table if the ‘platform’ identifier code is `TT_PLATFORM_MACINTOSH`.

`TT_MAC_LANGID_LATIN`
`TT_MAC_LANGID_MALAY_ARABIC_SCRIPT`
`TT_MAC_LANGID_HINDI`
`TT_MAC_LANGID_CATALAN`
`TT_MAC_LANGID_MARATHI`
`TT_MAC_LANGID_ICELANDIC`
`TT_MAC_LANGID_ARABIC`
`TT_MAC_LANGID_SWAHILI`
`TT_MAC_LANGID_KHMER`
`TT_MAC_LANGID_UKRAINIAN`
`TT_MAC_LANGID_FINNISH`
`TT_MAC_LANGID_POLISH`
`TT_MAC_LANGID_NEPALI`
`TT_MAC_LANGID_UZBEK`
`TT_MAC_LANGID_TELUGU`
`TT_MAC_LANGID_MALTESE`
`TT_MAC_LANGID_AFRIKAANS`
`TT_MAC_LANGID_CHEWA`

TT_MAC_LANGID_BASQUE
TT_MAC_LANGID_CZECH
TT_MAC_LANGID_ROMANIAN
TT_MAC_LANGID_QUECHUA
TT_MAC_LANGID_TAGALOG
TT_MAC_LANGID_HUNGARIAN
TT_MAC_LANGID_AZERBAIJANI_CYRILLIC_SCRIPT
TT_MAC_LANGID_TONGAN
TT_MAC_LANGID_SUNDANESE
TT_MAC_LANGID_JAPANESE
TT_MAC_LANGID_MONGOLIAN
TT_MAC_LANGID_ALBANIAN
TT_MAC_LANGID_NORWEGIAN
TT_MAC_LANGID_SLOVAK
TT_MAC_LANGID_MALAGASY
TT_MAC_LANGID_DZONGKHA
TT_MAC_LANGID_DUTCH
TT_MAC_LANGID_MALAY_ROMAN_SCRIPT
TT_MAC_LANGID_SERBIAN
TT_MAC_LANGID_GERMAN
TT_MAC_LANGID_SOMALI
TT_MAC_LANGID_KOREAN
TT_MAC_LANGID_MONGOLIAN_MONGOLIAN_SCRIPT
TT_MAC_LANGID_CROATIAN
TT_MAC_LANGID_TURKISH
TT_MAC_LANGID_MOLDAVIAN
TT_MAC_LANGID_LAO
TT_MAC_LANGID_ORIYA
TT_MAC_LANGID_BRETON
TT_MAC_LANGID_PASHTO
TT_MAC_LANGID_GUARANI
TT_MAC_LANGID_HEBREW
TT_MAC_LANGID_SLOVENIAN
TT_MAC_LANGID_ESTONIAN
TT_MAC_LANGID_RUNDI
TT_MAC_LANGID_URDU

TT_MAC_LANGID_CHINESE_TRADITIONAL
TT_MAC_LANGID_TATAR
TT_MAC_LANGID_CHINESE_SIMPLIFIED
TT_MAC_LANGID_AZERBAIJANI_ARABIC_SCRIPT
TT_MAC_LANGID_SANSKRIT
TT_MAC_LANGID_KURDISH
TT_MAC_LANGID_FAEROESE
TT_MAC_LANGID_MONGOLIAN_CYRILLIC_SCRIPT
TT_MAC_LANGID_TIGRINYA
TT_MAC_LANGID_THAI
TT_MAC_LANGID_DANISH
TT_MAC_LANGID_KAZAKH
TT_MAC_LANGID_YIDDISH
TT_MAC_LANGID_ESPERANTO
TT_MAC_LANGID_LITHUANIAN
TT_MAC_LANGID_FARSI
TT_MAC_LANGID_LETTISH
TT_MAC_LANGID_VIETNAMESE
TT_MAC_LANGID_PORTUGUESE
TT_MAC_LANGID_IRISH
TT_MAC_LANGID_WELSH
TT_MAC_LANGID_PUNJABI
TT_MAC_LANGID_GREEK
TT_MAC_LANGID_INUKTITUT
TT_MAC_LANGID_FRENCH
TT_MAC_LANGID_GREEK_POLYTONIC
TT_MAC_LANGID_AZERBAIJANI
TT_MAC_LANGID_JAVANESE
TT_MAC_LANGID_SWEDISH
TT_MAC_LANGID_UIGHUR
TT_MAC_LANGID_BENGALI
TT_MAC_LANGID_RUANDA
TT_MAC_LANGID_SINDHI
TT_MAC_LANGID_TIBETAN
TT_MAC_LANGID_ENGLISH
TT_MAC_LANGID_SAAMISK

`TT_MAC_LANGID_INDONESIAN`
`TT_MAC_LANGID_MANX_GAELIC`
`TT_MAC_LANGID_BYELORUSSIAN`
`TT_MAC_LANGID_BULGARIAN`
`TT_MAC_LANGID_GEORGIAN`
`TT_MAC_LANGID_AZERBAIJANI_ROMAN_SCRIPT`
`TT_MAC_LANGID_ITALIAN`
`TT_MAC_LANGID_SCOTTISH_GAELIC`
`TT_MAC_LANGID_ARMENIAN`
`TT_MAC_LANGID_GALLA`
`TT_MAC_LANGID_MACEDONIAN`
`TT_MAC_LANGID_IRISH_GAELIC`
`TT_MAC_LANGID_KIRGHIZ`
`TT_MAC_LANGID_TAMIL`
`TT_MAC_LANGID_SPANISH`
`TT_MAC_LANGID_BURMESE`
`TT_MAC_LANGID_KANNADA`
`TT_MAC_LANGID_GALICIAN`
`TT_MAC_LANGID_FLEMISH`
`TT_MAC_LANGID_TAJIKI`
`TT_MAC_LANGID_ASSAMESE`
`TT_MAC_LANGID_SINHALESE`
`TT_MAC_LANGID_GREELANDIC`
`TT_MAC_LANGID_AMHARIC`
`TT_MAC_LANGID_KASHMIRI`
`TT_MAC_LANGID_AYMARA`
`TT_MAC_LANGID_GUJARATI`
`TT_MAC_LANGID_RUSSIAN`
`TT_MAC_LANGID_TURKMEN`
`TT_MAC_LANGID_MALAYALAM`

TT_MS_IDS

A list of valid values for the ‘encoding_id’ for `TT_PLATFORM_MICROSOFT` charmaps and name entries.

TT_MS_ID_SYMBOL_CS

Corresponds to Microsoft symbol encoding. See `FT_ENCODING_MS_SYMBOL`.

TT_MS_ID_UNICODE_CS

Corresponds to a Microsoft WGL4 charmap, matching Unicode. See `FT_ENCODING_UNICODE`.

TT_MS_ID_SJIS

Corresponds to SJIS Japanese encoding. See FT_ENCODING_SJIS.

TT_MS_ID_GB2312

Corresponds to Simplified Chinese as used in Mainland China. See FT_ENCODING_GB2312.

TT_MS_ID_BIG_5

Corresponds to Traditional Chinese as used in Taiwan and Hong Kong. See FT_ENCODING_BIG5.

TT_MS_ID_WANSUNG

Corresponds to Korean Wansung encoding. See FT_ENCODING_WANSUNG.

TT_MS_ID_JOHAB

Corresponds to Johab encoding. See FT_ENCODING_JOHAB.

TT_MS_ID_UCS_4

Corresponds to UCS-4 or UTF-32 charmaps. This has been added to the OpenType specification version 1.4 (mid-2001.)

TT_MS_LANGIDS

Possible values of the language identifier field in the name records of the TTF ‘name’ table if the ‘platform’ identifier code is TT_PLATFORM_MICROSOFT.

TT_MS_LANGID_SANSKRIT_INDIA

TT_MS_LANGID_ENGLISH_UNITED_KINGDOM

TT_MS_LANGID_ENGLISH_BELIZE

TT_MS_LANGID_ARABIC_LEBANON

TT_MS_LANGID_MOLDAVIAN_MOLDAVIA

TT_MS_LANGID_TURKISH_TURKEY

TT_MS_LANGID_WELSH_WALES

TT_MS_LANGID_GERMAN_AUSTRIA

TT_MS_LANGID_DUTCH_BELGIUM

TT_MS_LANGID_YI_CHINA

TT_MS_LANGID_QUECHUA_ECUADOR

TT_MS_LANGID_SPANISH_EL_SALVADOR

TT_MS_LANGID_SWAHILI_KENYA

TT_MS_LANGID_QUECHUA_BOLIVIA

TT_MS_LANGID_SLOVENE_SLOVENIA

TT_MS_LANGID_ORIYA_INDIA

TT_MS_LANGID_FARSI_IRAN

TT_MS_LANGID_ENGLISH_CANADA

TT_MS_LANGID_NEPALI_NEPAL

TT_MS_LANGID_DHIVEHI_MALDIVES

TT_MS_LANGID_GERMAN_LIECHTENSTEIN

TT_MS_LANGID_TAMIL_INDIA

TT_MS_LANGID_ARABIC_UAE
TT_MS_LANGID_JAPANESE_JAPAN
TT_MS_LANGID_TAMAZIGHT_MOROCCO
TT_MS_LANGID_FRENCH_FRANCE
TT_MS_LANGID_CHINESE_MACAU
TT_MS_LANGID_VIETNAMESE_VIET_NAM
TT_MS_LANGID_HEBREW_ISRAEL
TT_MS_LANGID_SAMI_NORTHERN_SWEDEN
TT_MS_LANGID_PUNJABI_ARABIC_PAKISTAN
TT_MS_LANGID_SWEDISH_SWEDEN
TT_MS_LANGID_FRENCH_REUNION
TT_MS_LANGID_ARABIC_BAHRAIN
TT_MS_LANGID_ENGLISH_INDIA
TT_MS_LANGID_NEPALI_INDIA
TT_MS_LANGID_THAI_THAILAND
TT_MS_LANGID_ENGLISH_GENERAL
TT_MS_LANGID_SAMI_LULE_NORWAY
TT_MS_LANGID_ARABIC_OMAN
TT_MS_LANGID_SPANISH_HONDURAS
TT_MS_LANGID_ENGLISH_JAMAICA
TT_MS_LANGID_ESTONIAN_ESTONIA
TT_MS_LANGID_FRISIAN_NETHERLANDS
TT_MS_LANGID_LATIN
TT_MS_LANGID_ENGLISH_INDONESIA
TT_MS_LANGID_ENGLISH_IRELAND
TT_MS_LANGID_TIBETAN_CHINA
TT_MS_LANGID_PUNJABI_INDIA
TT_MS_LANGID_FRENCH_MALI
TT_MS_LANGID_GERMAN_LUXEMBOURG
TT_MS_LANGID_SUTU_SOUTH_AFRICA
TT_MS_LANGID_FRENCH_CAMEROON
TT_MS_LANGID_FRENCH_CONGO
TT_MS_LANGID_CLASSIC_LITHUANIAN_LITHUANIA
TT_MS_LANGID_MALAYALAM_INDIA
TT_MS_LANGID_SAMI_SOUTHERN_SWEDEN
TT_MS_LANGID_CHEROKEE_UNITED_STATES

TT_MS_LANGID_SPANISH_GUATEMALA
TT_MS_LANGID_CZECH_CZECH_REPUBLIC
TT_MS_LANGID_MANIPURI_INDIA
TT_MS_LANGID_ENGLISH_AUSTRALIA
TT_MS_LANGID_SPANISH_DOMINICAN_REPUBLIC
TT_MS_LANGID_ARABIC_LIBYA
TT_MS_LANGID_FRENCH_WEST_INDIES
TT_MS_LANGID_ENGLISH_TRINIDAD
TT_MS_LANGID_ARABIC_QATAR
TT_MS_LANGID_SPANISH_COLOMBIA
TT_MS_LANGID_GUARANI_PARAGUAY
TT_MS_LANGID_EDO_NIGERIA
TT_MS_LANGID_SEPEDI_SOUTH_AFRICA
TT_MS_LANGID_ENGLISH_HONG_KONG
TT_MS_LANGID_KOREAN_EXTENDED_WANSUNG_KOREA
TT_MS_LANGID_TATAR_TATARSTAN
TT_MS_LANGID_PASHTO_AFGHANISTAN
TT_MS_LANGID_KASHMIRI_PAKISTAN
TT_MS_LANGID_GALICIAN_SPAIN
TT_MS_LANGID_TAJIK_TAJIKISTAN
TT_MS_LANGID_SAMI_INARI_FINLAND
TT_MS_LANGID_KASHMIRI_SASIA
TT_MS_LANGID_SPANISH_ARGENTINA
TT_MS_LANGID_SAMI_SOUTHERN_NORWAY
TT_MS_LANGID_CROATIAN_CROATIA
TT_MS_LANGID_GUJARATI_INDIA
TT_MS_LANGID_TIBETAN_BHUTAN
TT_MS_LANGID_TIGRIGNA_ETHIOPIA
TT_MS_LANGID_FINNISH_FINLAND
TT_MS_LANGID_ENGLISH_UNITED_STATES
TT_MS_LANGID_ITALIAN_SWITZERLAND
TT_MS_LANGID_ARABIC_EGYPT
TT_MS_LANGID_SPANISH_LATIN_AMERICA
TT_MS_LANGID_LITHUANIAN_LITHUANIA
TT_MS_LANGID_ARABIC_ALGERIA
TT_MS_LANGID_MALAY_MALAYSIA

TT_MS_LANGID_ARABIC_GENERAL
TT_MS_LANGID_CHINESE_PRC
TT_MS_LANGID_BENGALI_BANGLADESH
TT_MS_LANGID_SPANISH_PERU
TT_MS_LANGID_SPANISH_SPAIN_INTERNATIONAL_SORT
TT_MS_LANGID_DIVEHI_MALDIVES
TT_MS_LANGID_LATVIAN_LATVIA
TT_MS_LANGID_TURKMEN_TURKMENISTAN
TT_MS_LANGID_XHOSA_SOUTH_AFRICA
TT_MS_LANGID_KHMER_CAMBODIA
TT_MS_LANGID_NORWEGIAN_NORWAY_NYNORSK
TT_MS_LANGID_ARABIC_MOROCCO
TT_MS_LANGID_FRENCH_SENEGAL
TT_MS_LANGID_YORUBA_NIGERIA
TT_MS_LANGID_CATALAN_SPAIN
TT_MS_LANGID_AFRIKAANS_SOUTH_AFRICA
TT_MS_LANGID_ZULU_SOUTH_AFRICA
TT_MS_LANGID_SPANISH_URUGUAY
TT_MS_LANGID_SPANISH_ECUADOR
TT_MS_LANGID_BOSNIAN_BOSNIA_HERZEGOVINA
TT_MS_LANGID_CHINESE_GENERAL
TT_MS_LANGID_SPANISH_PARAGUAY
TT_MS_LANGID_HINDI_INDIA
TT_MS_LANGID_FRENCH_LUXEMBOURG
TT_MS_LANGID_TSWANA_SOUTH_AFRICA
TT_MS_LANGID_HUNGARIAN_HUNGARY
TT_MS_LANGID_CROATIAN_BOSNIA_HERZEGOVINA
TT_MS_LANGID_ENGLISH_SINGAPORE
TT_MS_LANGID_MALTESE_MALTA
TT_MS_LANGID_SAMI_NORTHERN_FINLAND
TT_MS_LANGID_FRENCH_CANADA
TT_MS_LANGID_SAMI_LULE_SWEDEN
TT_MS_LANGID_KANURI_NIGERIA
TT_MS_LANGID_IRISH_GAELIC_IRELAND
TT_MS_LANGID_ARABIC_SAUDI_ARABIA
TT_MS_LANGID_FRENCH_HAITI

TT_MS_LANGID_SPANISH_PUERTO_RICO
TT_MS_LANGID_BURMESE_MYANMAR
TT_MS_LANGID_POLISH_POLAND
TT_MS_LANGID_PORTUGUESE_PORTUGAL
TT_MS_LANGID_ENGLISH_CARIBBEAN
TT_MS_LANGID_KIRGHIZ_KIRGHIZ_REPUBLIC
TT_MS_LANGID_ICELANDIC_ICELAND
TT_MS_LANGID_BENGALI_INDIA
TT_MS_LANGID_HAUSA_NIGERIA
TT_MS_LANGID_BASQUE_SPAIN
TT_MS_LANGID_UIGHUR_CHINA
TT_MS_LANGID_ENGLISH_MALAYSIA
TT_MS_LANGID_FRENCH_MONACO
TT_MS_LANGID_SPANISH_BOLIVIA
TT_MS_LANGID_SORBIAN_GERMANY
TT_MS_LANGID_SINDHI_INDIA
TT_MS_LANGID_CHINESE_SINGAPORE
TT_MS_LANGID_FRENCH_COTE_D_IVOIRE
TT_MS_LANGID_SPANISH_SPAIN_TRADITIONAL_SORT
TT_MS_LANGID_SERBIAN_SERBIA_CYRILLIC
TT_MS_LANGID_SAMI_SKOLT_FINLAND
TT_MS_LANGID_SERBIAN_BOSNIA_HERZ_CYRILLIC
TT_MS_LANGID_MALAY_BRUNEI_DARUSSALAM
TT_MS_LANGID_ARABIC_JORDAN
TT_MS_LANGID_MONGOLIAN_MONGOLIA_MONGOLIAN
TT_MS_LANGID_SERBIAN_SERBIA_LATIN
TT_MS_LANGID_RUSSIAN_RUSSIA
TT_MS_LANGID_ROMANIAN_ROMANIA
TT_MS_LANGID_FRENCH_NORTH_AFRICA
TT_MS_LANGID_MONGOLIAN_MONGOLIA
TT_MS_LANGID_TSONGA_SOUTH_AFRICA
TT_MS_LANGID_SOMALI_SOMALIA
TT_MS_LANGID_SAAMI_LAPONIA
TT_MS_LANGID_SPANISH_COSTA_RICA
TT_MS_LANGID_ARABIC_SYRIA
TT_MS_LANGID_SPANISH_PANAMA

TT_MS_LANGID_PAPIAMENTU_NETHERLANDS_ANTILLES
TT_MS_LANGID_ASSAMESE_INDIA
TT_MS_LANGID_SCOTTISH_GAELIC_UNITED_KINGDOM
TT_MS_LANGID_DUTCH_NETHERLANDS
TT_MS_LANGID_SINDHI_PAKISTAN
TT_MS_LANGID_MACEDONIAN_MACEDONIA
TT_MS_LANGID_KAZAK_KAZAKSTAN
TT_MS_LANGID_AZERI_AZERBAIJAN_LATIN
TT_MS_LANGID_BELARUSIAN_BELARUS
TT_MS_LANGID_FRENCH_MOROCCO
TT_MS_LANGID_SERBIAN_BOSNIA_HERZ_LATIN
TT_MS_LANGID_ALBANIAN_ALBANIA
TT_MS_LANGID_SINHALESE_SRI_LANKA
TT_MS_LANGID_SPANISH_MEXICO
TT_MS_LANGID_ENGLISH_ZIMBABWE
TT_MS_LANGID_OROMO_ETHIOPIA
TT_MS_LANGID_INDONESIAN_INDONESIA
TT_MS_LANGID_SAMI_NORTHERN_NORWAY
TT_MS_LANGID_UZBEK_UZBEKISTAN_LATIN
TT_MS_LANGID_SLOVAK_SLOVAKIA
TT_MS_LANGID_KASHMIRI_INDIA
TT_MS_LANGID_GERMAN_SWITZERLAND
TT_MS_LANGID_URDU_INDIA
TT_MS_LANGID_FAEROESE_FAEROE_ISLANDS
TT_MS_LANGID_SYRIAC_SYRIA
TT_MS_LANGID_SPANISH_CHILE
TT_MS_LANGID_FILIPINO_PHILIPPINES
TT_MS_LANGID_ARABIC_YEMEN
TT_MS_LANGID_KONKANI_INDIA
TT_MS_LANGID_AMHARIC_ETHIOPIA
TT_MS_LANGID_ENGLISH_NEW_ZEALAND
TT_MS_LANGID_RHAETO_ROMANIC_SWITZERLAND
TT_MS_LANGID_ARABIC_TUNISIA
TT_MS_LANGID_SOTHO_SOUTHERN_SOUTH_AFRICA
TT_MS_LANGID_QUECHUA_PERU
TT_MS_LANGID_DANISH_DENMARK

TT_MS_LANGID_ENGLISH_PHILIPPINES
TT_MS_LANGID_SPANISH_NICARAGUA
TT_MS_LANGID_INUKTITUT_CANADA
TT_MS_LANGID_UKRAINIAN_UKRAINE
TT_MS_LANGID_NORWEGIAN_NORWAY_BOKMAL
TT_MS_LANGID_UZBEK_UZBEKISTAN_CYRILLIC
TT_MS_LANGID_FRENCH_BELGIUM
TT_MS_LANGID_ENGLISH_SOUTH_AFRICA
TT_MS_LANGID_HAWAIIAN_UNITED_STATES
TT_MS_LANGID_ARABIC_IRAQ
TT_MS_LANGID_KANNADA_INDIA
TT_MS_LANGID_DZONGHKA_BHUTAN
TT_MS_LANGID_CHINESE_TAIWAN
TT_MS_LANGID_SPANISH_UNITED_STATES
TT_MS_LANGID_ARMENIAN_ARMENIA
TT_MS_LANGID_LAO_LAOS
TT_MS_LANGID_TIGRIGNA_ERYTREA
TT_MS_LANGID_MARATHI_INDIA
TT_MS_LANGID_ARABIC_KUWAIT
TT_MS_LANGID_TAMAZIGHT_MOROCCO_LATIN
TT_MS_LANGID_PORTUGUESE_BRAZIL
TT_MS_LANGID_TIGRIGNA_ERYTHREA
TT_MS_LANGID_GREEK_GREECE
TT_MS_LANGID_URDU_PAKISTAN
TT_MS_LANGID_KIRGHIZ_KIRGHIZSTAN
TT_MS_LANGID_YIDDISH_GERMANY
TT_MS_LANGID_GERMAN_GERMANY
TT_MS_LANGID_TELUGU_INDIA
TT_MS_LANGID_AZERI_AZERBAIJAN_CYRILLIC
TT_MS_LANGID_KOREAN_JOHAB_KOREA
TT_MS_LANGID_ITALIAN_ITALY
TT_MS_LANGID_MAORI_NEW_ZEALAND
TT_MS_LANGID_SPANISH_VENEZUELA
TT_MS_LANGID_IGBO_NIGERIA
TT_MS_LANGID_IBIBIO_NIGERIA
TT_MS_LANGID_CHINESE_HONG_KONG

TT_MS_LANGID_FRENCH_SWITZERLAND
TT_MS_LANGID_BULGARIAN_BULGARIA
TT_MS_LANGID_FULFULDE_NIGERIA
TT_MS_LANGID_RUSSIAN_MOLDAVIA
TT_MS_LANGID_VENDA_SOUTH_AFRICA
TT_MS_LANGID_GEORGIAN_GEORGIA
TT_MS_LANGID_SWEDISH_FINLAND

TT_NAME_IDS

Possible values of the ‘name’ identifier field in the name records of the TTF ‘name’ table. These values are platform independent.

TT_NAME_ID_COPYRIGHT
TT_NAME_ID_FONT_FAMILY
TT_NAME_ID_FONT_SUBFAMILY
TT_NAME_ID_UNIQUE_ID
TT_NAME_ID_FULL_NAME
TT_NAME_ID_VERSION_STRING
TT_NAME_ID_PS_NAME
TT_NAME_ID_TRADEMARK
TT_NAME_ID_MANUFACTURER
TT_NAME_ID_DESIGNER
TT_NAME_ID_DESCRIPTION
TT_NAME_ID_VENDOR_URL
TT_NAME_ID_DESIGNER_URL
TT_NAME_ID_LICENSE
TT_NAME_ID_LICENSE_URL
TT_NAME_ID_PREFERRED_FAMILY
TT_NAME_ID_PREFERRED_SUBFAMILY
TT_NAME_ID_MAC_FULL_NAME
TT_NAME_ID_SAMPLE_TEXT
TT_NAME_ID_CID_FINDFONT_NAME
TT_NAME_ID_WWS_FAMILY
TT_NAME_ID_WWS_SUBFAMILY

TT_PLATFORMS

A list of valid values for the ‘platform_id’ identifier code in FT_CharMapRec and FT_SfntName structures.

TT_PLATFORM_APPLE_UNICODE

Used by Apple to indicate a Unicode character map and/or name entry. See TT_APPLE_ID_XXX for corresponding ‘encoding_id’ values. Note that name entries in this format are coded as big-endian UCS-2 character codes only.

TT_PLATFORM_MACINTOSH

Used by Apple to indicate a MacOS-specific charmap and/or name entry. See TT_MAC_ID_XXX for corresponding ‘encoding_id’ values. Note that most TrueType fonts contain an Apple roman charmap to be usable on MacOS systems (even if they contain a Microsoft charmap as well).

TT_PLATFORM_ISO

This value was used to specify ISO/IEC 10646 charmaps. It is however now deprecated. See TT_ISO_ID_XXX for a list of corresponding ‘encoding_id’ values.

TT_PLATFORM_MICROSOFT

Used by Microsoft to indicate Windows-specific charmaps. See TT_MS_ID_XXX for a list of corresponding ‘encoding_id’ values. Note that most fonts contain a Unicode charmap using (TT_PLATFORM_MICROSOFT, TT_MS_ID_UNICODE_CS).

TT_PLATFORM_CUSTOM

Used to indicate application-specific charmaps.

TT_PLATFORM_ADOBE

This value isn’t part of any font format specification, but is used by FreeType to report Adobe-specific charmaps in an FT_CharMapRec structure. See TT_ADOBE_ID_XXX.

0.4.1

- Fixed a bug in `Face.load_char`
- Added `get_format` and `get_fstype` in `Face` (titusz.pan)

0.3.3

- Fixed a bug in `get_kerning`
- Added test against freetype version for `FT_ReferenceFace` and `FT_Get_FSType_Flags`

0.3.2

- Added `wordle.py` example
- Added `get_bbox` for `Outline` class
- Added `get_cbox` for `Outline` and `Glyph` classes
- Added `__del__` method to `Face` class
- Set encoding (utf-8) to all source files and examples.
- Added test against freetype version for `FT_Library_SetLcdFilterWeights`.

0.3.1

- Added `FT_Stroker` bindings (enums, structs and methods)

- Added ft-outline and ft-color examples
- Fixed first/next char in Face
- Pythonic interface has been documented

0.3.0

- Added ftdump.py demo and necessary functions

0.2.0

- Added sfnt functions
- Added TT_XXX flags in ft_enums
- New examples

0.1.1

- Initial release
- Working examples

Copyright (c) 2011-2014, Nicolas P. Rougier - All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

Neither the name of the freetype-py Development Team nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS “AS IS” AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

CHAPTER 7

Indices and tables

- `genindex`
- `modindex`
- `search`

A

advance (freetype.GlyphSlot attribute), 23
ascender (freetype.Face attribute), 11
ascender (freetype.SizeMetrics attribute), 18
attach_file() (freetype.Face method), 11
available_sizes (freetype.Face attribute), 11

B

BBox (class in freetype), 17
bbox (freetype.Face attribute), 11
begin_subpath() (freetype.Stroker method), 25
Bitmap (class in freetype), 19
bitmap (freetype.BitmapGlyph attribute), 23
bitmap (freetype.GlyphSlot attribute), 23
bitmap_left (freetype.GlyphSlot attribute), 24
bitmap_top (freetype.GlyphSlot attribute), 24
BitmapGlyph (class in freetype), 23
BitmapSize (class in freetype), 19
buffer (freetype.Bitmap attribute), 19

C

Charmap (class in freetype), 20
charmap (freetype.Face attribute), 11
charmaps (freetype.Face attribute), 12
cmap_format (freetype.Charmap attribute), 20
cmap_language_id (freetype.Charmap attribute), 20
conic_to() (freetype.Stroker method), 25
contours (freetype.Outline attribute), 21
cubic_to() (freetype.Stroker method), 25

D

descender (freetype.Face attribute), 12
descender (freetype.SizeMetrics attribute), 18

E

encoding (freetype.Charmap attribute), 20
encoding_id (freetype.Charmap attribute), 20
encoding_id (freetype.SfntName attribute), 24
encoding_name (freetype.Charmap attribute), 20

end_subpath() (freetype.Stroker method), 25
export() (freetype.Stroker method), 26
export_border() (freetype.Stroker method), 26

F

Face (class in freetype), 11
face_flags (freetype.Face attribute), 12
face_index (freetype.Face attribute), 12
family_name (freetype.Face attribute), 12
flags (freetype.Outline attribute), 21
format (freetype.BitmapGlyph attribute), 23
format (freetype.Glyph attribute), 22
format (freetype.GlyphSlot attribute), 24
FT_ENCODING_ADOBE_CUSTOM (built-in variable), 28
FT_ENCODING_ADOBE_EXPERT (built-in variable), 28
FT_ENCODING_ADOBE_LATIN_1 (built-in variable), 28
FT_ENCODING_ADOBE_STANDARD (built-in variable), 28
FT_ENCODING_APPLE_ROMAN (built-in variable), 28
FT_ENCODING_BIG5 (built-in variable), 28
FT_ENCODING_GB2312 (built-in variable), 28
FT_ENCODING_JOHAB (built-in variable), 28
FT_ENCODING_MS_SYMBOL (built-in variable), 27
FT_ENCODING_NONE (built-in variable), 27
FT_ENCODING_OLD_LATIN_2 (built-in variable), 28
FT_ENCODING_SJIS (built-in variable), 27
FT_ENCODING_UNICODE (built-in variable), 27
FT_ENCODING_WANSUNG (built-in variable), 28
FT_FACE_FLAG_CID_KEYED (built-in variable), 29
FT_FACE_FLAG_EXTERNAL_STREAM (built-in variable), 29
FT_FACE_FLAG_FIXED_SIZES (built-in variable), 28
FT_FACE_FLAG_FIXED_WIDTH (built-in variable), 28
FT_FACE_FLAG_GLYPH_NAMES (built-in variable), 29

- FT_FACE_FLAG_HINTER (built-in variable), 29
- FT_FACE_FLAG_HORIZONTAL (built-in variable), 28
- FT_FACE_FLAG_KERNING (built-in variable), 29
- FT_FACE_FLAG_MULTIPLE_MASTERS (built-in variable), 29
- FT_FACE_FLAG_SCALABLE (built-in variable), 28
- FT_FACE_FLAG_SFNT (built-in variable), 28
- FT_FACE_FLAG_TRICKY (built-in variable), 29
- FT_FACE_FLAG_VERTICAL (built-in variable), 29
- FT_FSTYPE_BITMAP_EMBEDDING_ONLY (built-in variable), 30
- FT_FSTYPE_EDITABLE_EMBEDDING (built-in variable), 30
- FT_FSTYPE_INSTALLABLE_EMBEDDING (built-in variable), 29
- FT_FSTYPE_NO_SUBSETTING (built-in variable), 30
- FT_FSTYPE_PREVIEW_AND_PRINT_EMBEDDING (built-in variable), 29
- FT_FSTYPE_RESTRICTED_LICENSE_EMBEDDING (built-in variable), 29
- FT_GLYPH_BBOX_GRIDFIT (built-in variable), 30
- FT_GLYPH_BBOX_PIXELS (built-in variable), 30
- FT_GLYPH_BBOX_SUBPIXELS (built-in variable), 30
- FT_GLYPH_BBOX_TRUNCATE (built-in variable), 30
- FT_GLYPH_BBOX_UNSCALED (built-in variable), 30
- FT_GLYPH_FORMAT_BITMAP (built-in variable), 30
- FT_GLYPH_FORMAT_COMPOSITE (built-in variable), 30
- FT_GLYPH_FORMAT_NONE (built-in variable), 30
- FT_GLYPH_FORMAT_OUTLINE (built-in variable), 30
- FT_GLYPH_FORMAT_PLOTTER (built-in variable), 30
- FT_KERNING_DEFAULT (built-in variable), 31
- FT_KERNING_UNFITTED (built-in variable), 31
- FT_KERNING_UNSCALED (built-in variable), 31
- FT_LCD_FILTER_DEFAULT (built-in variable), 31
- FT_LCD_FILTER_LEGACY (built-in variable), 31
- FT_LCD_FILTER_LIGHT (built-in variable), 31
- FT_LCD_FILTER_NONE (built-in variable), 31
- FT_LOAD_CROP_BITMAP (built-in variable), 32
- FT_LOAD_DEFAULT (built-in variable), 31
- FT_LOAD_FORCE_AUTOHINT (built-in variable), 32
- FT_LOAD_IGNORE_GLOBAL_ADVANCE_WIDTH (built-in variable), 32
- FT_LOAD_IGNORE_TRANSFORM (built-in variable), 32
- FT_LOAD_LINEAR_DESIGN (built-in variable), 32
- FT_LOAD_MONOCHROME (built-in variable), 32
- FT_LOAD_NO_AUTOHINT (built-in variable), 32
- FT_LOAD_NO_BITMAP (built-in variable), 32
- FT_LOAD_NO_HINTING (built-in variable), 31
- FT_LOAD_NO_RECURSE (built-in variable), 32
- FT_LOAD_NO_SCALE (built-in variable), 31
- FT_LOAD_PEDANTIC (built-in variable), 32
- FT_LOAD_RENDER (built-in variable), 32
- FT_LOAD_TARGET_LCD (built-in variable), 33
- FT_LOAD_TARGET_LCD_V (built-in variable), 33
- FT_LOAD_TARGET_LIGHT (built-in variable), 33
- FT_LOAD_TARGET_MONO (built-in variable), 33
- FT_LOAD_TARGET_NORMAL (built-in variable), 33
- FT_LOAD_VERTICAL_LAYOUT (built-in variable), 32
- FT_OPEN_DRIVER (built-in variable), 33
- FT_OPEN_MEMORY (built-in variable), 33
- FT_OPEN_PARAMS (built-in variable), 33
- FT_OPEN_PATHNAME (built-in variable), 33
- FT_OPEN_STREAM (built-in variable), 33
- FT_OUTLINE_EVEN_ODD_FILL (built-in variable), 34
- FT_OUTLINE_HIGH_PRECISION (built-in variable), 34
- FT_OUTLINE_IGNORE_DROPOUTS (built-in variable), 34
- FT_OUTLINE_INCLUDE_STUBS (built-in variable), 34
- FT_OUTLINE_NONE (built-in variable), 33
- FT_OUTLINE_OWNER (built-in variable), 33
- FT_OUTLINE_REVERSE_FILL (built-in variable), 34
- FT_OUTLINE_SINGLE_PASS (built-in variable), 34
- FT_OUTLINE_SMART_DROPOUTS (built-in variable), 34
- FT_PIXEL_MODE_GRAY (built-in variable), 34
- FT_PIXEL_MODE_GRAY2 (built-in variable), 34
- FT_PIXEL_MODE_GRAY4 (built-in variable), 34
- FT_PIXEL_MODE_LCD (built-in variable), 34
- FT_PIXEL_MODE_LCD_V (built-in variable), 35
- FT_PIXEL_MODE_MONO (built-in variable), 34
- FT_PIXEL_MODE_NONE (built-in variable), 34
- FT_RENDER_MODE_LCD (built-in variable), 35
- FT_RENDER_MODE_LCD_V (built-in variable), 35
- FT_RENDER_MODE_LIGHT (built-in variable), 35
- FT_RENDER_MODE_MONO (built-in variable), 35
- FT_RENDER_MODE_NORMAL (built-in variable), 35
- FT_STROKER_BORDER_LEFT (built-in variable), 35
- FT_STROKER_BORDER_RIGHT (built-in variable), 35
- FT_STROKER_LINECAP_BUTT (built-in variable), 36
- FT_STROKER_LINECAP_ROUND (built-in variable), 36
- FT_STROKER_LINECAP_SQUARE (built-in variable), 36
- FT_STROKER_LINEJOIN_BEVEL (built-in variable), 36
- FT_STROKER_LINEJOIN_MITER (built-in variable), 36
- FT_STROKER_LINEJOIN_ROUND (built-in variable), 36
- FT_STYLE_FLAG_BOLD (built-in variable), 36
- FT_STYLE_FLAG_ITALIC (built-in variable), 36

G

[get_advance\(\)](#) (freetype.Face method), 12
[get_bbox\(\)](#) (freetype.Outline method), 21
[get_border_counts\(\)](#) (freetype.Stroker method), 26
[get_cbox\(\)](#) (freetype.Glyph method), 22
[get_cbox2\(\)](#) (freetype.Outline method), 21
[get_char_index\(\)](#) (freetype.Face method), 12
[get_chars\(\)](#) (freetype.Face method), 12
[get_counts\(\)](#) (freetype.Stroker method), 26
[get_first_char\(\)](#) (freetype.Face method), 12
[get_format\(\)](#) (freetype.Face method), 13
[get_fstype\(\)](#) (freetype.Face method), 13
[get_glyph\(\)](#) (freetype.GlyphSlot method), 24
[get_glyph_name\(\)](#) (freetype.Face method), 13
[get_inside_border\(\)](#) (freetype.Outline method), 21
[get_kerning\(\)](#) (freetype.Face method), 13
[get_name_index\(\)](#) (freetype.Face method), 13
[get_next_char\(\)](#) (freetype.Face method), 13
[get_outside_border\(\)](#) (freetype.Outline method), 21
[get_sfnt_name\(\)](#) (freetype.Face method), 14
[Glyph](#) (class in freetype), 22
[glyph](#) (freetype.Face attribute), 14
[GlyphSlot](#) (class in freetype), 23

H

[has_fixed_sizes](#) (freetype.Face attribute), 14
[has_glyph_names](#) (freetype.Face attribute), 14
[has_horizontal](#) (freetype.Face attribute), 14
[has_kerning](#) (freetype.Face attribute), 14
[has_multiple_masters](#) (freetype.Face attribute), 14
[has_vertical](#) (freetype.Face attribute), 14
[height](#) (freetype.BitmapSize attribute), 19
[height](#) (freetype.Face attribute), 14
[height](#) (freetype.SizeMetrics attribute), 18

I

[index](#) (freetype.Charmap attribute), 21
[is_cid_keyed](#) (freetype.Face attribute), 14
[is_fixed_width](#) (freetype.Face attribute), 14
[is_scalable](#) (freetype.Face attribute), 14
[is_sfnt](#) (freetype.Face attribute), 14
[is_tricky](#) (freetype.Face attribute), 15

L

[language_id](#) (freetype.SfntName attribute), 24
[left](#) (freetype.BitmapGlyph attribute), 23
[line_to\(\)](#) (freetype.Stroker method), 26
[linearHoriAdvance](#) (freetype.GlyphSlot attribute), 24
[linearVertAdvance](#) (freetype.GlyphSlot attribute), 24
[load_char\(\)](#) (freetype.Face method), 15
[load_glyph\(\)](#) (freetype.Face method), 15

M

[max_advance](#) (freetype.SizeMetrics attribute), 18

[max_advance_height](#) (freetype.Face attribute), 15
[max_advance_width](#) (freetype.Face attribute), 15
[metrics](#) (freetype.GlyphSlot attribute), 24

N

[name_id](#) (freetype.SfntName attribute), 24
[next](#) (freetype.GlyphSlot attribute), 24
[num_faces](#) (freetype.Face attribute), 15
[num_fixed_sizes](#) (freetype.Face attribute), 15
[num_glyphs](#) (freetype.Face attribute), 15
[num_grays](#) (freetype.Bitmap attribute), 19

O

[Outline](#) (class in freetype), 21
[outline](#) (freetype.GlyphSlot attribute), 24

P

[palette](#) (freetype.Bitmap attribute), 19
[palette_mode](#) (freetype.Bitmap attribute), 19
[parse_outline\(\)](#) (freetype.Stroker method), 26
[pitch](#) (freetype.Bitmap attribute), 19
[pixel_mode](#) (freetype.Bitmap attribute), 20
[platform_id](#) (freetype.Charmap attribute), 21
[platform_id](#) (freetype.SfntName attribute), 25
[points](#) (freetype.Outline attribute), 21
[postscript_name](#) (freetype.Face attribute), 16

R

[rewind\(\)](#) (freetype.Stroker method), 27
[rows](#) (freetype.Bitmap attribute), 20

S

[select_charmap\(\)](#) (freetype.Face method), 16
[select_size\(\)](#) (freetype.Face method), 16
[set\(\)](#) (freetype.Stroker method), 27
[set_char_size\(\)](#) (freetype.Face method), 16
[set_charmap\(\)](#) (freetype.Face method), 16
[set_pixel_sizes\(\)](#) (freetype.Face method), 16
[set_transform\(\)](#) (freetype.Face method), 16
[sfnt_name_count](#) (freetype.Face attribute), 17
[SfntName](#) (class in freetype), 24
[size](#) (freetype.BitmapSize attribute), 19
[size](#) (freetype.Face attribute), 17
[SizeMetrics](#) (class in freetype), 18
[string](#) (freetype.SfntName attribute), 25
[string_len](#) (freetype.SfntName attribute), 25
[stroke\(\)](#) (freetype.Glyph method), 22
[Stroker](#) (class in freetype), 25
[style_flags](#) (freetype.Face attribute), 17
[style_name](#) (freetype.Face attribute), 17

T

[tags](#) (freetype.Outline attribute), 21

- to_bitmap() (freetype.Glyph method), 22
- top (freetype.BitmapGlyph attribute), 23
- TT_ADOBE_ID_CUSTOM (built-in variable), 36
- TT_ADOBE_ID_EXPERT (built-in variable), 36
- TT_ADOBE_ID_LATIN_1 (built-in variable), 36
- TT_ADOBE_ID_STANDARD (built-in variable), 36
- TT_APPLE_ID_DEFAULT (built-in variable), 37
- TT_APPLE_ID_ISO_10646 (built-in variable), 37
- TT_APPLE_ID_UNICODE_1_1 (built-in variable), 37
- TT_APPLE_ID_UNICODE_2_0 (built-in variable), 37
- TT_APPLE_ID_UNICODE_32 (built-in variable), 37
- TT_APPLE_ID_VARIANT_SELECTOR (built-in variable), 37
- TT_MAC_ID_ARABIC (built-in variable), 38
- TT_MAC_ID_ARMENIAN (built-in variable), 38
- TT_MAC_ID_BENGALI (built-in variable), 38
- TT_MAC_ID_BURMESE (built-in variable), 37
- TT_MAC_ID_DEVANAGARI (built-in variable), 37
- TT_MAC_ID_GEEZ (built-in variable), 38
- TT_MAC_ID_GEORGIAN (built-in variable), 38
- TT_MAC_ID_GREEK (built-in variable), 37
- TT_MAC_ID_GUJARATI (built-in variable), 38
- TT_MAC_ID_GURMUKHI (built-in variable), 37
- TT_MAC_ID_HEBREW (built-in variable), 37
- TT_MAC_ID_JAPANESE (built-in variable), 37
- TT_MAC_ID_KANNADA (built-in variable), 37
- TT_MAC_ID_KHMER (built-in variable), 38
- TT_MAC_ID_KOREAN (built-in variable), 38
- TT_MAC_ID_LAOTIAN (built-in variable), 37
- TT_MAC_ID_MALAYALAM (built-in variable), 38
- TT_MAC_ID_MALDIVIAN (built-in variable), 37
- TT_MAC_ID_MONGOLIAN (built-in variable), 37
- TT_MAC_ID_ORIYA (built-in variable), 38
- TT_MAC_ID_ROMAN (built-in variable), 37
- TT_MAC_ID_RSYMBOL (built-in variable), 38
- TT_MAC_ID_RUSSIAN (built-in variable), 37
- TT_MAC_ID_SIMPLIFIED_CHINESE (built-in variable), 37
- TT_MAC_ID_SINDHI (built-in variable), 37
- TT_MAC_ID_SINHALESE (built-in variable), 37
- TT_MAC_ID_SLAVIC (built-in variable), 38
- TT_MAC_ID_TAMIL (built-in variable), 37
- TT_MAC_ID_TELUGU (built-in variable), 37
- TT_MAC_ID_THAI (built-in variable), 37
- TT_MAC_ID_TIBETAN (built-in variable), 37
- TT_MAC_ID_TRADITIONAL_CHINESE (built-in variable), 37
- TT_MAC_ID_UNINTERP (built-in variable), 38
- TT_MAC_ID_VIETNAMESE (built-in variable), 37
- TT_MAC_LANGID_AFRIKAANS (built-in variable), 38
- TT_MAC_LANGID_ALBANIAN (built-in variable), 39
- TT_MAC_LANGID_AMHARIC (built-in variable), 41
- TT_MAC_LANGID_ARABIC (built-in variable), 38
- TT_MAC_LANGID_ARMENIAN (built-in variable), 41
- TT_MAC_LANGID_ASSAMESE (built-in variable), 41
- TT_MAC_LANGID_AYMARA (built-in variable), 41
- TT_MAC_LANGID_AZERBAIJANI (built-in variable), 40
- TT_MAC_LANGID_AZERBAIJANI_ARABIC_SCRIPT (built-in variable), 40
- TT_MAC_LANGID_AZERBAIJANI_CYRILLIC_SCRIPT (built-in variable), 39
- TT_MAC_LANGID_AZERBAIJANI_ROMAN_SCRIPT (built-in variable), 41
- TT_MAC_LANGID_BASQUE (built-in variable), 38
- TT_MAC_LANGID_BENGALI (built-in variable), 40
- TT_MAC_LANGID_BRETON (built-in variable), 39
- TT_MAC_LANGID_BULGARIAN (built-in variable), 41
- TT_MAC_LANGID_BURMESE (built-in variable), 41
- TT_MAC_LANGID_BYELORUSSIAN (built-in variable), 41
- TT_MAC_LANGID_CATALAN (built-in variable), 38
- TT_MAC_LANGID_CHEWA (built-in variable), 38
- TT_MAC_LANGID_CHINESE_SIMPLIFIED (built-in variable), 40
- TT_MAC_LANGID_CHINESE_TRADITIONAL (built-in variable), 39
- TT_MAC_LANGID_CROATIAN (built-in variable), 39
- TT_MAC_LANGID_CZECH (built-in variable), 39
- TT_MAC_LANGID_DANISH (built-in variable), 40
- TT_MAC_LANGID_DUTCH (built-in variable), 39
- TT_MAC_LANGID_DZONGKHA (built-in variable), 39
- TT_MAC_LANGID_ENGLISH (built-in variable), 40
- TT_MAC_LANGID_ESPERANTO (built-in variable), 40
- TT_MAC_LANGID_ESTONIAN (built-in variable), 39
- TT_MAC_LANGID_FAEROESE (built-in variable), 40
- TT_MAC_LANGID_FARSI (built-in variable), 40
- TT_MAC_LANGID_FINNISH (built-in variable), 38
- TT_MAC_LANGID_FLEMISH (built-in variable), 41
- TT_MAC_LANGID_FRENCH (built-in variable), 40
- TT_MAC_LANGID_GALICIAN (built-in variable), 41
- TT_MAC_LANGID_GALLA (built-in variable), 41
- TT_MAC_LANGID_GEORGIAN (built-in variable), 41
- TT_MAC_LANGID_GERMAN (built-in variable), 39
- TT_MAC_LANGID_GREEK (built-in variable), 40
- TT_MAC_LANGID_GREEK_POLYTONIC (built-in variable), 40
- TT_MAC_LANGID_GREELANDIC (built-in variable), 41
- TT_MAC_LANGID_GUARANI (built-in variable), 39
- TT_MAC_LANGID_GUJARATI (built-in variable), 41
- TT_MAC_LANGID_HEBREW (built-in variable), 39
- TT_MAC_LANGID_HINDI (built-in variable), 38

- TT_MAC_LANGID_HUNGARIAN (built-in variable), 39
- TT_MAC_LANGID_ICELANDIC (built-in variable), 38
- TT_MAC_LANGID_INDONESIAN (built-in variable), 40
- TT_MAC_LANGID_INUKTITUT (built-in variable), 40
- TT_MAC_LANGID_IRISH (built-in variable), 40
- TT_MAC_LANGID_IRISH_GAELIC (built-in variable), 41
- TT_MAC_LANGID_ITALIAN (built-in variable), 41
- TT_MAC_LANGID_JAPANESE (built-in variable), 39
- TT_MAC_LANGID_JAVANESE (built-in variable), 40
- TT_MAC_LANGID_KANNADA (built-in variable), 41
- TT_MAC_LANGID_KASHMIRI (built-in variable), 41
- TT_MAC_LANGID_KAZAKH (built-in variable), 40
- TT_MAC_LANGID_KHMER (built-in variable), 38
- TT_MAC_LANGID_KIRGHIZ (built-in variable), 41
- TT_MAC_LANGID_KOREAN (built-in variable), 39
- TT_MAC_LANGID_KURDISH (built-in variable), 40
- TT_MAC_LANGID_LAO (built-in variable), 39
- TT_MAC_LANGID_LATIN (built-in variable), 38
- TT_MAC_LANGID_LETTISH (built-in variable), 40
- TT_MAC_LANGID_LITHUANIAN (built-in variable), 40
- TT_MAC_LANGID_MACEDONIAN (built-in variable), 41
- TT_MAC_LANGID_MALAGASY (built-in variable), 39
- TT_MAC_LANGID_MALAY_ARABIC_SCRIPT (built-in variable), 38
- TT_MAC_LANGID_MALAY_ROMAN_SCRIPT (built-in variable), 39
- TT_MAC_LANGID_MALAYALAM (built-in variable), 41
- TT_MAC_LANGID_MALTESE (built-in variable), 38
- TT_MAC_LANGID_MANX_GAELIC (built-in variable), 41
- TT_MAC_LANGID_MARATHI (built-in variable), 38
- TT_MAC_LANGID_MOLDAVIAN (built-in variable), 39
- TT_MAC_LANGID_MONGOLIAN (built-in variable), 39
- TT_MAC_LANGID_MONGOLIAN_CYRILLIC_SCRIPT (built-in variable), 40
- TT_MAC_LANGID_MONGOLIAN_MONGOLIAN_SCRIPT (built-in variable), 39
- TT_MAC_LANGID_NEPALI (built-in variable), 38
- TT_MAC_LANGID_NORWEGIAN (built-in variable), 39
- TT_MAC_LANGID_ORIYA (built-in variable), 39
- TT_MAC_LANGID_PASHTO (built-in variable), 39
- TT_MAC_LANGID_POLISH (built-in variable), 38
- TT_MAC_LANGID_PORTUGUESE (built-in variable), 40
- TT_MAC_LANGID_PUNJABI (built-in variable), 40
- TT_MAC_LANGID_QUECHUA (built-in variable), 39
- TT_MAC_LANGID_ROMANIAN (built-in variable), 39
- TT_MAC_LANGID_RUANDA (built-in variable), 40
- TT_MAC_LANGID_RUNDI (built-in variable), 39
- TT_MAC_LANGID_RUSSIAN (built-in variable), 41
- TT_MAC_LANGID_SAAMISK (built-in variable), 40
- TT_MAC_LANGID_SANSKRIT (built-in variable), 40
- TT_MAC_LANGID_SCOTTISH_GAELIC (built-in variable), 41
- TT_MAC_LANGID_SERBIAN (built-in variable), 39
- TT_MAC_LANGID_SINDHI (built-in variable), 40
- TT_MAC_LANGID_SINHALESE (built-in variable), 41
- TT_MAC_LANGID_SLOVAK (built-in variable), 39
- TT_MAC_LANGID_SLOVENIAN (built-in variable), 39
- TT_MAC_LANGID_SOMALI (built-in variable), 39
- TT_MAC_LANGID_SPANISH (built-in variable), 41
- TT_MAC_LANGID_SUNDANESE (built-in variable), 39
- TT_MAC_LANGID_SWAHILI (built-in variable), 38
- TT_MAC_LANGID_SWEDISH (built-in variable), 40
- TT_MAC_LANGID_TAGALOG (built-in variable), 39
- TT_MAC_LANGID_TAJIKI (built-in variable), 41
- TT_MAC_LANGID_TAMIL (built-in variable), 41
- TT_MAC_LANGID_TATAR (built-in variable), 40
- TT_MAC_LANGID_TELUGU (built-in variable), 38
- TT_MAC_LANGID_THAI (built-in variable), 40
- TT_MAC_LANGID_TIBETAN (built-in variable), 40
- TT_MAC_LANGID_TIGRINYA (built-in variable), 40
- TT_MAC_LANGID_TONGAN (built-in variable), 39
- TT_MAC_LANGID_TURKISH (built-in variable), 39
- TT_MAC_LANGID_TURKMEN (built-in variable), 41
- TT_MAC_LANGID_UGHUR (built-in variable), 40
- TT_MAC_LANGID_UKRAINIAN (built-in variable), 38
- TT_MAC_LANGID_URDU (built-in variable), 39
- TT_MAC_LANGID_UZBEK (built-in variable), 38
- TT_MAC_LANGID_VIETNAMESE (built-in variable), 40
- TT_MAC_LANGID_WELSH (built-in variable), 40
- TT_MAC_LANGID_YIDDISH (built-in variable), 40
- TT_MS_ID_BIG_5 (built-in variable), 42
- TT_MS_ID_GB2312 (built-in variable), 42
- TT_MS_ID_JOHAB (built-in variable), 42
- TT_MS_ID_SJIS (built-in variable), 42
- TT_MS_ID_SYMBOL_CS (built-in variable), 41
- TT_MS_ID_UCS_4 (built-in variable), 42
- TT_MS_ID_UNICODE_CS (built-in variable), 41
- TT_MS_ID_WANSUNG (built-in variable), 42
- TT_MS_LANGID_AFRIKAANS_SOUTH_AFRICA (built-in variable), 45
- TT_MS_LANGID_ALBANIAN_ALBANIA (built-in variable), 47

TT_MS_LANGID_AMHARIC_ETHIOPIA (built-in variable), 47
 TT_MS_LANGID_ARABIC_ALGERIA (built-in variable), 44
 TT_MS_LANGID_ARABIC_BAHRAIN (built-in variable), 43
 TT_MS_LANGID_ARABIC_EGYPT (built-in variable), 44
 TT_MS_LANGID_ARABIC_GENERAL (built-in variable), 44
 TT_MS_LANGID_ARABIC_IRAQ (built-in variable), 48
 TT_MS_LANGID_ARABIC_JORDAN (built-in variable), 46
 TT_MS_LANGID_ARABIC_KUWAIT (built-in variable), 48
 TT_MS_LANGID_ARABIC_LEBANON (built-in variable), 42
 TT_MS_LANGID_ARABIC_LIBYA (built-in variable), 44
 TT_MS_LANGID_ARABIC_MOROCCO (built-in variable), 45
 TT_MS_LANGID_ARABIC_OMAN (built-in variable), 43
 TT_MS_LANGID_ARABIC_QATAR (built-in variable), 44
 TT_MS_LANGID_ARABIC_SAUDI_ARABIA (built-in variable), 45
 TT_MS_LANGID_ARABIC_SYRIA (built-in variable), 46
 TT_MS_LANGID_ARABIC_TUNISIA (built-in variable), 47
 TT_MS_LANGID_ARABIC_UAE (built-in variable), 43
 TT_MS_LANGID_ARABIC_YEMEN (built-in variable), 47
 TT_MS_LANGID_ARMENIAN_ARMENIA (built-in variable), 48
 TT_MS_LANGID_ASSAMESE_INDIA (built-in variable), 47
 TT_MS_LANGID_AZERI_AZERBAIJAN_CYRILLIC (built-in variable), 48
 TT_MS_LANGID_AZERI_AZERBAIJAN_LATIN (built-in variable), 47
 TT_MS_LANGID_BASQUE_SPAIN (built-in variable), 46
 TT_MS_LANGID_BELARUSIAN_BELARUS (built-in variable), 47
 TT_MS_LANGID_BENGALI_BANGLADESH (built-in variable), 45
 TT_MS_LANGID_BENGALI_INDIA (built-in variable), 46
 TT_MS_LANGID_BOSNIAN_BOSNIA_HERZEGOVINA (built-in variable), 45
 TT_MS_LANGID_BULGARIAN_BULGARIA (built-in variable), 49
 TT_MS_LANGID_BURMESE_MYANMAR (built-in variable), 46
 TT_MS_LANGID_CATALAN_SPAIN (built-in variable), 45
 TT_MS_LANGID_CHEROKEE_UNITED_STATES (built-in variable), 43
 TT_MS_LANGID_CHINESE_GENERAL (built-in variable), 45
 TT_MS_LANGID_CHINESE_HONG_KONG (built-in variable), 48
 TT_MS_LANGID_CHINESE_MACAU (built-in variable), 43
 TT_MS_LANGID_CHINESE_PRC (built-in variable), 45
 TT_MS_LANGID_CHINESE_SINGAPORE (built-in variable), 46
 TT_MS_LANGID_CHINESE_TAIWAN (built-in variable), 48
 TT_MS_LANGID_CLASSIC_LITHUANIAN_LITHUANIA (built-in variable), 43
 TT_MS_LANGID_CROATIAN_BOSNIA_HERZEGOVINA (built-in variable), 45
 TT_MS_LANGID_CROATIAN_CROATIA (built-in variable), 44
 TT_MS_LANGID_CZECH_CZECH_REPUBLIC (built-in variable), 44
 TT_MS_LANGID_DANISH_DENMARK (built-in variable), 47
 TT_MS_LANGID_DHIVEHI_MALDIVES (built-in variable), 42
 TT_MS_LANGID_DIVEHI_MALDIVES (built-in variable), 45
 TT_MS_LANGID_DUTCH_BELGIUM (built-in variable), 42
 TT_MS_LANGID_DUTCH_NETHERLANDS (built-in variable), 47
 TT_MS_LANGID_DZONGHKA_BHUTAN (built-in variable), 48
 TT_MS_LANGID_EDO_NIGERIA (built-in variable), 44
 TT_MS_LANGID_ENGLISH_AUSTRALIA (built-in variable), 44
 TT_MS_LANGID_ENGLISH_BELIZE (built-in variable), 42
 TT_MS_LANGID_ENGLISH_CANADA (built-in variable), 42
 TT_MS_LANGID_ENGLISH_CARIBBEAN (built-in variable), 46
 TT_MS_LANGID_ENGLISH_GENERAL (built-in variable), 43
 TT_MS_LANGID_ENGLISH_HONG_KONG (built-in variable), 44
 TT_MS_LANGID_ENGLISH_INDIA (built-in variable), 44

able), 43

TT_MS_LANGID_ENGLISH_INDONESIA (built-in variable), 43

TT_MS_LANGID_ENGLISH_IRELAND (built-in variable), 43

TT_MS_LANGID_ENGLISH_JAMAICA (built-in variable), 43

TT_MS_LANGID_ENGLISH_MALAYSIA (built-in variable), 46

TT_MS_LANGID_ENGLISH_NEW_ZEALAND (built-in variable), 47

TT_MS_LANGID_ENGLISH_PHILIPPINES (built-in variable), 47

TT_MS_LANGID_ENGLISH_SINGAPORE (built-in variable), 45

TT_MS_LANGID_ENGLISH_SOUTH_AFRICA (built-in variable), 48

TT_MS_LANGID_ENGLISH_TRINIDAD (built-in variable), 44

TT_MS_LANGID_ENGLISH_UNITED_KINGDOM (built-in variable), 42

TT_MS_LANGID_ENGLISH_UNITED_STATES (built-in variable), 44

TT_MS_LANGID_ENGLISH_ZIMBABWE (built-in variable), 47

TT_MS_LANGID_ESTONIAN_ESTONIA (built-in variable), 43

TT_MS_LANGID_FAEROESE_FAEROE_ISLANDS (built-in variable), 47

TT_MS_LANGID_FARSI_IRAN (built-in variable), 42

TT_MS_LANGID_FILIPINO_PHILIPPINES (built-in variable), 47

TT_MS_LANGID_FINNISH_FINLAND (built-in variable), 44

TT_MS_LANGID_FRENCH_BELGIUM (built-in variable), 48

TT_MS_LANGID_FRENCH_CAMEROON (built-in variable), 43

TT_MS_LANGID_FRENCH_CANADA (built-in variable), 45

TT_MS_LANGID_FRENCH_CONGO (built-in variable), 43

TT_MS_LANGID_FRENCH_COTE_D_IVOIRE (built-in variable), 46

TT_MS_LANGID_FRENCH_FRANCE (built-in variable), 43

TT_MS_LANGID_FRENCH_HAITI (built-in variable), 45

TT_MS_LANGID_FRENCH_LUXEMBOURG (built-in variable), 45

TT_MS_LANGID_FRENCH_MALI (built-in variable), 43

TT_MS_LANGID_FRENCH_MONACO (built-in variable), 46

TT_MS_LANGID_FRENCH_MOROCCO (built-in variable), 47

TT_MS_LANGID_FRENCH_NORTH_AFRICA (built-in variable), 46

TT_MS_LANGID_FRENCH_REUNION (built-in variable), 43

TT_MS_LANGID_FRENCH_SENEGAL (built-in variable), 45

TT_MS_LANGID_FRENCH_SWITZERLAND (built-in variable), 48

TT_MS_LANGID_FRENCH_WEST_INDIES (built-in variable), 44

TT_MS_LANGID_FRISIAN_NETHERLANDS (built-in variable), 43

TT_MS_LANGID_FULFULDE_NIGERIA (built-in variable), 49

TT_MS_LANGID_GALICIAN_SPAIN (built-in variable), 44

TT_MS_LANGID_GEORGIAN_GEORGIA (built-in variable), 49

TT_MS_LANGID_GERMAN_AUSTRIA (built-in variable), 42

TT_MS_LANGID_GERMAN_GERMANY (built-in variable), 48

TT_MS_LANGID_GERMAN_LIECHTENSTEIN (built-in variable), 42

TT_MS_LANGID_GERMAN_LUXEMBOURG (built-in variable), 43

TT_MS_LANGID_GERMAN_SWITZERLAND (built-in variable), 47

TT_MS_LANGID_GREEK_GREECE (built-in variable), 48

TT_MS_LANGID_GUARANI_PARAGUAY (built-in variable), 44

TT_MS_LANGID_GUJARATI_INDIA (built-in variable), 44

TT_MS_LANGID_HAUSA_NIGERIA (built-in variable), 46

TT_MS_LANGID_HAWAIIAN_UNITED_STATES (built-in variable), 48

TT_MS_LANGID_HEBREW_ISRAEL (built-in variable), 43

TT_MS_LANGID_HINDI_INDIA (built-in variable), 45

TT_MS_LANGID_HUNGARIAN_HUNGARY (built-in variable), 45

TT_MS_LANGID_IBIBIO_NIGERIA (built-in variable), 48

TT_MS_LANGID_ICELANDIC_ICELAND (built-in variable), 46

TT_MS_LANGID_IGBO_NIGERIA (built-in variable), 48

TT_MS_LANGID_INDONESIAN_INDONESIA (built-in variable), 47

TT_MS_LANGID_INUKTITUT_CANADA (built-in

variable), 48

TT_MS_LANGID_IRISH_GAELIC_IRELAND (built-in variable), 45

TT_MS_LANGID_ITALIAN_ITALY (built-in variable), 48

TT_MS_LANGID_ITALIAN_SWITZERLAND (built-in variable), 44

TT_MS_LANGID_JAPANESE_JAPAN (built-in variable), 43

TT_MS_LANGID_KANNADA_INDIA (built-in variable), 48

TT_MS_LANGID_KANURI_NIGERIA (built-in variable), 45

TT_MS_LANGID_KASHMIRI_INDIA (built-in variable), 47

TT_MS_LANGID_KASHMIRI_PAKISTAN (built-in variable), 44

TT_MS_LANGID_KASHMIRI_SASIA (built-in variable), 44

TT_MS_LANGID_KAZAK_KAZAKSTAN (built-in variable), 47

TT_MS_LANGID_KHMER_CAMBODIA (built-in variable), 45

TT_MS_LANGID_KIRGHIZ_KIRGHIZ_REPUBLIC (built-in variable), 46

TT_MS_LANGID_KIRGHIZ_KIRGHIZSTAN (built-in variable), 48

TT_MS_LANGID_KONKANI_INDIA (built-in variable), 47

TT_MS_LANGID_KOREAN_EXTENDED_WANSUNG_KOREA (built-in variable), 44

TT_MS_LANGID_KOREAN_JOHAB_KOREA (built-in variable), 48

TT_MS_LANGID_LAO_LAOS (built-in variable), 48

TT_MS_LANGID_LATIN (built-in variable), 43

TT_MS_LANGID_LATVIAN_LATVIA (built-in variable), 45

TT_MS_LANGID_LITHUANIAN_LITHUANIA (built-in variable), 44

TT_MS_LANGID_MACEDONIAN_MACEDONIA (built-in variable), 47

TT_MS_LANGID_MALAY_BRUNEI_DARUSSALAM (built-in variable), 46

TT_MS_LANGID_MALAY_MALAYSIA (built-in variable), 44

TT_MS_LANGID_MALAYALAM_INDIA (built-in variable), 43

TT_MS_LANGID_MALTESE_MALTA (built-in variable), 45

TT_MS_LANGID_MANIPURI_INDIA (built-in variable), 44

TT_MS_LANGID_MAORI_NEW_ZEALAND (built-in variable), 48

TT_MS_LANGID_MARATHI_INDIA (built-in variable), 48

TT_MS_LANGID_MOLDAVIAN_MOLDAVIA (built-in variable), 42

TT_MS_LANGID_MONGOLIAN_MONGOLIA (built-in variable), 46

TT_MS_LANGID_MONGOLIAN_MONGOLIA_MONGOLIAN (built-in variable), 46

TT_MS_LANGID_NEPALI_INDIA (built-in variable), 43

TT_MS_LANGID_NEPALI_NEPAL (built-in variable), 42

TT_MS_LANGID_NORWEGIAN_NORWAY_BOKMAL (built-in variable), 48

TT_MS_LANGID_NORWEGIAN_NORWAY_NYNORSK (built-in variable), 45

TT_MS_LANGID_ORIYA_INDIA (built-in variable), 42

TT_MS_LANGID_OROMO_ETHIOPIA (built-in variable), 47

TT_MS_LANGID_PAPIAMENTU_NETHERLANDS_ANTILLES (built-in variable), 46

TT_MS_LANGID_PASHTO_AFGHANISTAN (built-in variable), 44

TT_MS_LANGID_POLISH_POLAND (built-in variable), 46

TT_MS_LANGID_PORTUGUESE_BRAZIL (built-in variable), 48

TT_MS_LANGID_PORTUGUESE_PORTUGAL (built-in variable), 46

TT_MS_LANGID_PUNJABI_ARABIC_PAKISTAN (built-in variable), 43

TT_MS_LANGID_PUNJABI_INDIA (built-in variable), 43

TT_MS_LANGID_QUECHUA_BOLIVIA (built-in variable), 42

TT_MS_LANGID_QUECHUA_ECUADOR (built-in variable), 42

TT_MS_LANGID_QUECHUA_PERU (built-in variable), 47

TT_MS_LANGID_RHAETO_ROMANIC_SWITZERLAND (built-in variable), 47

TT_MS_LANGID_ROMANIAN_ROMANIA (built-in variable), 46

TT_MS_LANGID_RUSSIAN_MOLDAVIA (built-in variable), 49

TT_MS_LANGID_RUSSIAN_RUSSIA (built-in variable), 46

TT_MS_LANGID_SAAMI_LAPONIA (built-in variable), 46

TT_MS_LANGID_SAMI_INARI_FINLAND (built-in variable), 44

TT_MS_LANGID_SAMI_LULE_NORWAY (built-in variable), 43

TT_MS_LANGID_SAMI_LULE_SWEDEN (built-in

variable), 45

TT_MS_LANGID_SAMI_NORTHERN_FINLAND (built-in variable), 45

TT_MS_LANGID_SAMI_NORTHERN_NORWAY (built-in variable), 47

TT_MS_LANGID_SAMI_NORTHERN_SWEDEN (built-in variable), 43

TT_MS_LANGID_SAMI_SKOLT_FINLAND (built-in variable), 46

TT_MS_LANGID_SAMI_SOUTHERN_NORWAY (built-in variable), 44

TT_MS_LANGID_SAMI_SOUTHERN_SWEDEN (built-in variable), 43

TT_MS_LANGID_SANSKRIT_INDIA (built-in variable), 42

TT_MS_LANGID_SCOTTISH_GAELIC_UNITED_KINGDOMS (built-in variable), 47

TT_MS_LANGID_SEPEDI_SOUTH_AFRICA (built-in variable), 44

TT_MS_LANGID_SERBIAN_BOSNIA_HERZ_CYRILLIC (built-in variable), 46

TT_MS_LANGID_SERBIAN_BOSNIA_HERZ_LATIN (built-in variable), 47

TT_MS_LANGID_SERBIAN_SERBIA_CYRILLIC (built-in variable), 46

TT_MS_LANGID_SERBIAN_SERBIA_LATIN (built-in variable), 46

TT_MS_LANGID_SINDHI_INDIA (built-in variable), 46

TT_MS_LANGID_SINDHI_PAKISTAN (built-in variable), 47

TT_MS_LANGID_SINHALESE_SRI_LANKA (built-in variable), 47

TT_MS_LANGID_SLOVAK_SLOVAKIA (built-in variable), 47

TT_MS_LANGID_SLOVENE_SLOVENIA (built-in variable), 42

TT_MS_LANGID_SOMALI_SOMALIA (built-in variable), 46

TT_MS_LANGID_SORBIAN_GERMANY (built-in variable), 46

TT_MS_LANGID_SOTHO_SOUTHERN_SOUTH_AFRICA (built-in variable), 47

TT_MS_LANGID_SPANISH_ARGENTINA (built-in variable), 44

TT_MS_LANGID_SPANISH_BOLIVIA (built-in variable), 46

TT_MS_LANGID_SPANISH_CHILE (built-in variable), 47

TT_MS_LANGID_SPANISH_COLOMBIA (built-in variable), 44

TT_MS_LANGID_SPANISH_COSTA_RICA (built-in variable), 46

TT_MS_LANGID_SPANISH_DOMINICAN_REPUBLIC (built-in variable), 44

TT_MS_LANGID_SPANISH_ECUADOR (built-in variable), 45

TT_MS_LANGID_SPANISH_EL_SALVADOR (built-in variable), 42

TT_MS_LANGID_SPANISH_GUATEMALA (built-in variable), 43

TT_MS_LANGID_SPANISH_HONDURAS (built-in variable), 43

TT_MS_LANGID_SPANISH_LATIN_AMERICA (built-in variable), 44

TT_MS_LANGID_SPANISH_MEXICO (built-in variable), 47

TT_MS_LANGID_SPANISH_NICARAGUA (built-in variable), 48

TT_MS_LANGID_SPANISH_PANAMA (built-in variable), 46

TT_MS_LANGID_SPANISH_PARAGUAY (built-in variable), 45

TT_MS_LANGID_SPANISH_PERU (built-in variable), 45

TT_MS_LANGID_SPANISH_PUERTO_RICO (built-in variable), 45

TT_MS_LANGID_SPANISH_SPAIN_INTERNATIONAL_SORT (built-in variable), 45

TT_MS_LANGID_SPANISH_SPAIN_TRADITIONAL_SORT (built-in variable), 46

TT_MS_LANGID_SPANISH_UNITED_STATES (built-in variable), 48

TT_MS_LANGID_SPANISH_URUGUAY (built-in variable), 45

TT_MS_LANGID_SPANISH_VENEZUELA (built-in variable), 48

TT_MS_LANGID_SUTU_SOUTH_AFRICA (built-in variable), 43

TT_MS_LANGID_SWAHILI_KENYA (built-in variable), 42

TT_MS_LANGID_SWEDISH_FINLAND (built-in variable), 49

TT_MS_LANGID_SWEDISH_SWEDEN (built-in variable), 43

TT_MS_LANGID_SYRIAC_SYRIA (built-in variable), 47

TT_MS_LANGID_TAJIK_TAJIKISTAN (built-in variable), 44

TT_MS_LANGID_TAMAZIGHT_MOROCCO (built-in variable), 43

TT_MS_LANGID_TAMAZIGHT_MOROCCO_LATIN (built-in variable), 48

TT_MS_LANGID_TAMIL_INDIA (built-in variable), 42

TT_MS_LANGID_TATAR_TATARSTAN (built-in variable), 44

TT_MS_LANGID_TELUGU_INDIA (built-in variable),

- 48
 - TT_MS_LANGID_THAI_THAILAND (built-in variable), 43
 - TT_MS_LANGID_TIBETAN_BHUTAN (built-in variable), 44
 - TT_MS_LANGID_TIBETAN_CHINA (built-in variable), 43
 - TT_MS_LANGID_TIGRIGNA_ERYTHREA (built-in variable), 48
 - TT_MS_LANGID_TIGRIGNA_ERYTREA (built-in variable), 48
 - TT_MS_LANGID_TIGRIGNA_ETHIOPIA (built-in variable), 44
 - TT_MS_LANGID_TSONGA_SOUTH_AFRICA (built-in variable), 46
 - TT_MS_LANGID_TSWANA_SOUTH_AFRICA (built-in variable), 45
 - TT_MS_LANGID_TURKISH_TURKEY (built-in variable), 42
 - TT_MS_LANGID_TURKMEN_TURKMENISTAN (built-in variable), 45
 - TT_MS_LANGID_UIGHUR_CHINA (built-in variable), 46
 - TT_MS_LANGID_UKRAINIAN_UKRAINE (built-in variable), 48
 - TT_MS_LANGID_URDU_INDIA (built-in variable), 47
 - TT_MS_LANGID_URDU_PAKISTAN (built-in variable), 48
 - TT_MS_LANGID_UZBEK_UZBEKISTAN_CYRILLIC (built-in variable), 48
 - TT_MS_LANGID_UZBEK_UZBEKISTAN_LATIN (built-in variable), 47
 - TT_MS_LANGID_VENDA_SOUTH_AFRICA (built-in variable), 49
 - TT_MS_LANGID_VIETNAMESE_VIET_NAM (built-in variable), 43
 - TT_MS_LANGID_WELSH_WALES (built-in variable), 42
 - TT_MS_LANGID_XHOSA_SOUTH_AFRICA (built-in variable), 45
 - TT_MS_LANGID_YI_CHINA (built-in variable), 42
 - TT_MS_LANGID_YIDDISH_GERMANY (built-in variable), 48
 - TT_MS_LANGID_YORUBA_NIGERIA (built-in variable), 45
 - TT_MS_LANGID_ZULU_SOUTH_AFRICA (built-in variable), 45
 - TT_NAME_ID_CID_FINDFONT_NAME (built-in variable), 49
 - TT_NAME_ID_COPYRIGHT (built-in variable), 49
 - TT_NAME_ID_DESCRIPTION (built-in variable), 49
 - TT_NAME_ID_DESIGNER (built-in variable), 49
 - TT_NAME_ID_DESIGNER_URL (built-in variable), 49
 - TT_NAME_ID_FONT_FAMILY (built-in variable), 49
 - TT_NAME_ID_FONT_SUBFAMILY (built-in variable), 49
 - TT_NAME_ID_FULL_NAME (built-in variable), 49
 - TT_NAME_ID_LICENSE (built-in variable), 49
 - TT_NAME_ID_LICENSE_URL (built-in variable), 49
 - TT_NAME_ID_MAC_FULL_NAME (built-in variable), 49
 - TT_NAME_ID_MANUFACTURER (built-in variable), 49
 - TT_NAME_ID_PREFERRED_FAMILY (built-in variable), 49
 - TT_NAME_ID_PREFERRED_SUBFAMILY (built-in variable), 49
 - TT_NAME_ID_PS_NAME (built-in variable), 49
 - TT_NAME_ID_SAMPLE_TEXT (built-in variable), 49
 - TT_NAME_ID_TRADEMARK (built-in variable), 49
 - TT_NAME_ID_UNIQUE_ID (built-in variable), 49
 - TT_NAME_ID_VENDOR_URL (built-in variable), 49
 - TT_NAME_ID_VERSION_STRING (built-in variable), 49
 - TT_NAME_ID_WWS_FAMILY (built-in variable), 49
 - TT_NAME_ID_WWS_SUBFAMILY (built-in variable), 49
 - TT_PLATFORM_ADOBE (built-in variable), 50
 - TT_PLATFORM_APPLE_UNICODE (built-in variable), 50
 - TT_PLATFORM_CUSTOM (built-in variable), 50
 - TT_PLATFORM_ISO (built-in variable), 50
 - TT_PLATFORM_MACINTOSH (built-in variable), 50
 - TT_PLATFORM_MICROSOFT (built-in variable), 50
- ## U
- underline_position (freetype.Face attribute), 17
 - underline_thickness (freetype.Face attribute), 17
 - units_per_EM (freetype.Face attribute), 17
- ## W
- width (freetype.Bitmap attribute), 20
 - width (freetype.BitmapSize attribute), 19
- ## X
- x_ppem (freetype.BitmapSize attribute), 19
 - x_ppem (freetype.SizeMetrics attribute), 18
 - x_scale (freetype.SizeMetrics attribute), 18
 - xMax (freetype.BBox attribute), 17
 - xMin (freetype.BBox attribute), 17
- ## Y
- y_ppem (freetype.BitmapSize attribute), 19
 - y_ppem (freetype.SizeMetrics attribute), 18
 - y_scale (freetype.SizeMetrics attribute), 18
 - yMax (freetype.BBox attribute), 18
 - yMin (freetype.BBox attribute), 18