# pangocffi

***Release 0.11.0***

**Oct 07, 2022**

# Contents

pangocffi is a CFFI-based set of Python bindings for pango.

pangocffi on its own is not that useful, since it depends on a PangoFontMap being declared against the PangoContext. PangoFontMap instances can easily be retrieved from libraries such as PangoCairo, PangoXft, PangoFT2, and PangoWin32 (See gnome's documentation 'Rendering with Pango' for a list of rendering engines).

See pangocairocffi for bindings that allow you to render pango objects with cairo.

The bindings are currently not fully implemented. Feel free to make a pull request to contribute!

CHAPTER 1

# Documentation

## 1.1 Overview

### 1.1.1 Installing

#### Installing Pango, GLib, FFI

pangocffi depends on Pango, GLib, and libffi being installed.

- On Mac OS, this is as easy as installing homebrew and then running:

```
brew install pkg-config
brew install libffi
brew install pango
brew install glib
```

- On Linux, these packages are usually installed as part of the base OS.
- On Windows, if you are using a 64-bit Python, you can use this binary installer to install GTK 3. Use `python --version --version` to check whether you're running 32-bit or 64-bit Python.

Feel free to contribute a guide for how to install these dependencies on other systems.

#### Installing pangocffi

Install with pip:

```
pip install pangocffi
```

Note: Python versions < 3.6 are not supported.

### 1.1.2 Importing pangocffi

The module to import is named `pangocffi`, however you are welcome to alias the module as `pango`:

```
import pangocffi as pango
```

pangocffi will dynamically load Pango, GLib, and GObject as a shared library upon importing. If it fails to find it, you will see an exception like this:

```
OSError: dlopen() failed to load pango: pango / pango-1 / ...
```

If Pango, GLib, or GObject are not installed as a shared library, pangocffi supports specifying a path via environment variables respectively: `PANGO_LOCATION`, `GLIB_LOCATION`, `GOBJECT_LOCATION`.

### 1.1.3 Basic usage

pangocffi on its own is not that useful, since it depends on a Pango `FontMap` being declared against the Pango `Context`. `FontMap` instances can only be retrieved from libraries such as PangoCairo, PangoXft, PangoFT2, PangoWin32 (See gnome's documentation Rendering with Pango for a list of rendering engines).

See pangocairocffi for bindings that allow you to render pango objects with cairo.

## 1.2 Python API reference

### 1.2.1 Rendering

Functions to run the rendering pipeline.

**class** pangocffi.**Context**

The *Context* structure stores global information used to control the itemization process.

**font_description**

The default font description for the context.

**Parameters** **desc** – the new Pango font description.

**base_gravity**

The base gravity for the context, which is used for laying out vertical text.

**gravity**

The gravity for the context. This is similar to *base_gravity*, unless it's *Gravity.AUTO()*; `get_for_matrix()` is used to retrieve the appropriate gravity from the current context matrix.

**gravity_hint**

The gravity hint for the context, which is used when laying out vertical text, and is only relevant if the gravity of the context is *Gravity.EAST()* or *Gravity.WEST()*.

### 1.2.2 Glyph Storage

Structures for storing information about glyphs.

### Units and conversion

pangocffi.**units_to_double**(*i: int*) → float
Converts a number in Pango units to floating-point: divides it by PANGO_SCALE.

**Parameters** **i** – value in Pango units

pangocffi.**units_from_double**(*d: float*) → int
Converts a floating-point number to Pango units: multiplies it by PANGO_SCALE and rounds to nearest integer.

**Parameters** **d** – double floating-point value

### Rectangle

**class** pangocffi.**Rectangle**(*pointer: Optional[_cffi_backend._CDataBase] = None*, *width: Optional[int] = None*, *height: Optional[int] = None*, *x: Optional[int] = None*, *y: Optional[int] = None*)
The *Rectangle* structure represents a rectangle. It is frequently used to represent the logical or ink extents of a single glyph or section of text.

**x**
X coordinate of the left side of the rectangle.

**y**
Y coordinate of the the top side of the rectangle.

**width**
The width of the rectangle.

**height**
The height of the rectangle.

### Matrix

API not implemented yet.

### Glyph String

API not implemented yet.

### Glyph Item

**class** pangocffi.**GlyphItem**(*pointer: _cffi_backend._CDataBase = None*, **init_args*)
A *GlyphItem* is a pair of a *Item* and the glyphs resulting from shaping the text corresponding to an item. As an example of the usage of *GlyphItem*, the results of shaping text with *Layout* is a list of LayoutLine, each of which contains a list of *GlyphItem*.

**item**
Corresponding *Item*.

**split**(*text: str*, *split_index: int*) → pangocffi.glyph_item.GlyphItem
Modifies orig to cover only the text after split_index, and returns a new item that covers the text before split_index that used to be in orig. You can think of split_index as the length of the returned item. split_index may not be 0, and it may not be greater than or equal to the length of orig (that is, there must be at least one byte assigned to each item, you can't create a zero-length item).

This function is similar in function to `Item.split()` (and uses it internally.)

**Parameters**

- **text** – text to which positions in `orig` apply
- **split_index** – byte index of position to split item, relative to the start of the item

**Returns** a new *GlyphItem* representing text before `split_index`.

**get_logical_widths**(*text: str*) → List[int]

Given a *GlyphItem* and the corresponding text, determine the screen width corresponding to each character. When multiple characters compose a single cluster, the width of the entire cluster is divided equally among the characters.

See also `GlyphString.get_logical_widths()`.

**Parameters** **text** – text that `glyph_item` corresponds to ( `glyph_item->item->offset` is an offset from the start of `text`)

**Returns** an array whose length is the number of characters in *glyph_item* (equal to `glyph_item->item->num_chars`) to be filled in with the resulting character widths.)

### Glyph Item Iterator

**class** pangocffi.**GlyphItemIter**(*pointer: _cffi_backend._CDataBase = None*, **init_args*)

A *GlyphItemIter* is an iterator over the clusters in a *GlyphItem*. The forward direction of the iterator is the logical direction of text. That is, with increasing `start_index` and `start_char` values. If `glyph_item` is right-to-left (that is, if `glyph_item->item->analysis.level` is odd), then `start_glyph` decreases as the iterator moves forward. Moreover, in right-to-left cases, `start_glyph` is greater than `end_glyph`.

Note that `text` is the start of the text for layout, which is then indexed by `glyph_item->item->offset` to get to the text of `glyph_item`. The `start_index` and `end_index` values can directly index into `text`. The `start_glyph`, `end_glyph`, `start_char`, and `end_char` values however are zero-based for the `glyph_item`. For each cluster, the item pointed at by the start variables is included in the cluster while the one pointed at by end variables is not.

None of the members of a *GlyphItemIter* should be modified manually.

**init_start**(*glyph_item: pangocffi.glyph_item.GlyphItem*, *text: str*) → bool

Initializes the *GlyphItemIter* structure to point to the first cluster in a glyph item. See *GlyphItemIter* for details of cluster orders.

**Returns** `False` if there are no clusters in the glyph item

**init_end**(*glyph_item: pangocffi.glyph_item.GlyphItem*, *text: str*) → bool

Initializes the *GlyphItemIter* structure to point to the last cluster in a glyph item. See *GlyphItemIter* for details of cluster orders.

**Returns** `False` if there are no clusters in the glyph item

**next_cluster**() → bool

Moves the iterator forward to the next run in visual order. If the iterator was already at the end of the layout, it will return `False`.

**Returns** `True` if the iterator was advanced, `False` if we were already on the last cluster.

**prev_cluster**() → bool

Moves the iterator to the preceding cluster in the glyph item. See *GlyphItemIter* for details of cluster orders.

**Returns** `True` if the iterator was moved, `False` if we were already on the first cluster.

**glyph_item**
The glyph item being iterated over.

**text**
The text being iterated over.

**start_index**
The index at which this cluster starts within the text.

**end_index**
The character at which this cluster ends within the text.

#### Item

**class** pangocffi.**Item**(*pointer: _cffi_backend._CDataBase = None*, **init_args*)
The *Item* structure stores information about a segment of text.

**offset**
Byte offset of the start of this item in text.

**length**
Length of this item in bytes.

**num_chars**
Length of this item in characters.

### 1.2.3 Fonts

Structures representing abstract fonts.

#### Font Description

**class** pangocffi.**FontDescription**(*pointer: _cffi_backend._CDataBase = None*, **init_args*)
The *FontDescription* represents the description of an ideal font. These structures are used both to list what fonts are available on the system and also for specifying the characteristics of a font to load.

**family**
The family name of the font description. This represents a font family of related font styles, and will resolve to a particular `FontFamily`. In some uses of *FontDescription*, it is also possible to use a comma-separated list of family names for this field.

**style**
The slant style of the font description.

Most fonts will either have an italic style or an oblique style, but not both, and font matching in Pango will match italic specifications with oblique fonts and vice-versa if an exact match is not found.

**variant**
The variant of the font description, which describes whether the font is normal or small caps.

**stretch**
The stretch of the font description, which specifies how narrow or wide the font should be.

**size**
The size field of a font description in fractional points, scaled by `PANGO_SCALE`. (That is, a size value of `10 * PANGO_SCALE` is a 10 point font). The conversion factor between points and device units

depends on system configuration and the output device. For screen display, a logical DPI of 96 is common, in which case a 10 point font corresponds to a `10 * (96 / 72) = 13.3` pixel font. Use *set_absolute_size()* if you need a particular size in device units.

**size_is_absolute**
Determines whether the size of the font is in points (not absolute) or device units (absolute). See *size* and *set_absolute_size()*.

**Returns** whether the size for the font description is in device units.

**gravity**
Sets the gravity of the font description, which specifies how the glyphs should be rotated. If gravity is `Gravity.AUTO`, this actually unsets the gravity mask on the font description.

**set_absolute_size**(*size: float*) → None
Sets the size field of a font description, in device units. There are `PANGO_SCALE` Pango units in one device unit. For an output backend where a device unit is a pixel, a size value of `10 * PANGO_SCALE` gives a 10 pixel font. See *size*.

### Font Metrics

API not implemented yet.

### Font Map

API not implemented yet.

### Font Set

API not implemented yet.

### Font Fields

API not implemented yet.

### Style

**class** pangocffi.**Style**
An enumeration specifying the various slant styles possible for a font.

**NORMAL = 0**
the font is upright.

**OBLIQUE = 1**
the font is slanted, but in a roman style.

**ITALIC = 2**
the font is slanted in an italic style.

### Weight

**class** pangocffi.**Weight**

An enumeration specifying the weight (boldness) of a font. This is a numerical value ranging from 100 to 1000, but there are some predefined values.

**THIN = 100**
the thin weight( = 100; Since: 1.24)

**ULTRALIGHT = 200**
the ultralight weight( = 200)

**LIGHT = 300**
the light weight( = 300)

**SEMILIGHT = 350**
the semilight weight( = 350; Since: 1.36.7)

**BOOK = 380**
the book weight( = 380; Since: 1.24)

**NORMAL = 400**
the default weight( = 400)

**MEDIUM = 500**
the normal weight( = 500; Since: 1.24)

**SEMIBOLD = 600**
the semibold weight( = 600)

**BOLD = 700**
the bold weight( = 700)

**ULTRABOLD = 800**
the ultrabold weight( = 800)

**HEAVY = 900**
the heavy weight( = 900)

**ULTRAHEAVY = 1000**
the ultraheavy weight( = 1000; Since: 1.24)

### Variant

**class** pangocffi.**Variant**

An enumeration specifying capitalization variant of the font.

**NORMAL = 0**
A normal font.

**SMALL_CAPS = 1**
A font with the lower case characters replaced by smaller variants of the capital characters.

### Stretch

**class** pangocffi.**Stretch**

An enumeration specifying the width of the font relative to other designs within a family.

**ULTRA_CONDENSED = 0**
Ultra Condensed width

**EXTRA_CONDENSED = 1**
Extra Condensed width

**CONDENSED = 2**
Condensed width

**SEMI_CONDENSED = 3**
Semi condensed width

**NORMAL = 4**
The normal width

**SEMI_EXPANDED = 5**
Semi expanded width

**EXPANDED = 6**
Expanded width

**EXTRA_EXPANDED = 7**
Extra Expanded width

**ULTRA_EXPANDED = 8**
Ultra Expanded width

#### FontMask

**class** pangocffi.**FontMask**
The bits in a *FontMask* correspond to fields in a *FontDescription* that have been set.

**FAMILY = 1**
the font family is specified.

**STYLE = 2**
the font style is specified.

**VARIANT = 4**
the font variant is specified.

**WEIGHT = 8**
the font weight is specified.

**STRETCH = 16**
the font stretch is specified.

**SIZE = 32**
the font size is specified.

**GRAVITY = 64**
the font gravity is specified(Since: 1.16.)

### 1.2.4 Text Attributes

Font and other attributes for annotating text.

### Attribute

**class** pangocffi.**Attribute**(*pointer: _cffi_backend._CDataBase = None*, **init_args*)

The Attributes — Font and other attributes for annotating text. Attributed text is used in a number of places in Pango. It is used as the input to the itemization process and also when creating a PangoLayout. The data types and functions in this section are used to represent and manipulate sets of attributes applied to a portion of text.

**start_index**

The index at which this attribute starts.

**end_index**

The index at which this attriute ends.

**classmethod from_family**(*family: str*, *start_index: int = 0*, *end_index: int = 1*) → pangocffi.attribute.Attribute

Create a new font family attribute.

**Parameters**

- **family** – the font family
- **start_index** – the start index of the range. Should be >=0.
- **end_index** – end index of the range. The character at this index is not included in the range.

**Returns** the Attribute.

**classmethod from_style**(*style: pangocffi.enums.Style*, *start_index: int = 0*, *end_index: int = 1*) → pangocffi.attribute.Attribute

Create a new font slant style attribute.

**Parameters**

- **style** – the slant style
- **start_index** – the start index of the range. Should be >=0.
- **end_index** – end index of the range. The character at this index is not included in the range.

**Returns** the Attribute.

**Raises** AssertionError When `style` isn't a *Style*.

**classmethod from_variant**(*variant: pangocffi.enums.Variant*, *start_index: int = 0*, *end_index: int = 1*) → pangocffi.attribute.Attribute

Create a new font variant attribute (normal or small caps)

**Parameters**

- **variant** – the variant
- **start_index** – the start index of the range. Should be >=0.
- **end_index** – end index of the range. The character at this index is not included in the range.

**Returns** the Attribute.

**Raises** AssertionError When `variant` isn't a *Variant*.

**classmethod from_stretch**(*stretch: pangocffi.enums.Stretch*, *start_index: int = 0*, *end_index: int = 1*) → pangocffi.attribute.Attribute

Create a new font stretch attribute

**Parameters**

- **stretch** – the stretch
- **start_index** – the start index of the range. Should be >=0.
- **end_index** – end index of the range. The character at this index is not included in the range.

**Returns** the Attribute.

**Raises** AssertionError When `stretch` isn't a *Stretch*.

**classmethod from_weight**(*weight: pangocffi.enums.Weight*, *start_index: int = 0*, *end_index: int = 1*) → pangocffi.attribute.Attribute

Create a new font weight attribute.

**Parameters**

- **weight** – the weight
- **start_index** – the start index of the range. Should be >=0.
- **end_index** – end index of the range. The character at this index is not included in the range.

**Returns** the Attribute.

**Raises** AssertionError When `weight` isn't a *Weight*.

**classmethod from_size**(*size: int*, *start_index: int = 0*, *end_index: int = 1*) → pangocffi.attribute.Attribute

Create a new font-size attribute in fractional points.

**Parameters**

- **size** – the font size, in PANGO_SCALEths of a point.
- **start_index** – the start index of the range. Should be >=0.
- **end_index** – end index of the range. The character at this index is not included in the range.

**Returns** the Attribute.

**Raises** AssertionError When `size` isn't a `int`.

**classmethod from_size_absolute**(*size: int*, *start_index: int = 0*, *end_index: int = 1*)

Create a new font-size attribute in device units.

**Parameters**

- **size** – the font size, in PANGO_SCALEths of a device unit.
- **start_index** – the start index of the range. Should be >=0.
- **end_index** – end index of the range. The character at this index is not included in the range.

**Returns** the Attribute.

**Raises** AssertionError When `absolute_size` isn't a `int`.

**classmethod from_font_desc**(*font_desc: pangocffi.font_description.FontDescription*, *start_index: int = 0*, *end_index: int = 1*) → pangocffi.attribute.Attribute

Create a new font description attribute. This attribute allows setting family, style, weight, variant, stretch, and size simultaneously.

**Parameters**

- **font_desc** – the font description
- **start_index** – the start index of the range. Should be >=0.
- **end_index** – end index of the range. The character at this index is not included in the range.

**Returns** the Attribute.

**Raises** AssertionError When `font_desc` isn't a *FontDescription*.

**classmethod from_foreground_color**(*red: int*, *green: int*, *blue: int*, *start_index: int = 0*, *end_index: int = 1*) → pangocffi.attribute.Attribute

Create a new foreground color attribute.

**Parameters**

- **red** – the red value (ranging from 0 to 65535)
- **green** – the green value (ranging from 0 to 65535)
- **blue** – the blue value (ranging from 0 to 65535)
- **start_index** – the start index of the range. Should be >=0.
- **end_index** – end index of the range. The character at this index is not included in the range.

**Returns** the Attribute.

**Raises** AssertionError When `red` or `blue` or `green` isn't a `int`.

**classmethod from_background_color**(*red: int*, *green: int*, *blue: int*, *start_index: int = 0*, *end_index: int = 1*)

Create a new background color attribute.

**Parameters**

- **red** – the red value (ranging from 0 to 65535)
- **green** – the green value (ranging from 0 to 65535)
- **blue** – the blue value (ranging from 0 to 65535)
- **start_index** – the start index of the range. Should be >=0.
- **end_index** – end index of the range. The character at this index is not included in the range.

**Returns** the Attribute.

**Raises** AssertionError When `red` or `blue` or `green` isn't a `int`.

**classmethod from_strikethrough**(*strikethrough: bool*, *start_index: int = 0*, *end_index: int = 1*) → pangocffi.attribute.Attribute

Create a new strike-through attribute.

**Parameters**

- **strikethrough** – True if the text should be struck-through.
- **start_index** – the start index of the range. Should be >=0.
- **end_index** – end index of the range. The character at this index is not included in the range.

**Returns** the Attribute.

**Raises** AssertionError When `strikethrough` isn't a `bool`.

**classmethod from_strikethrough_color**(*red: int*, *green: int*, *blue: int*, *start_index: int = 0*, *end_index: int = 1*) → pangocffi.attribute.Attribute

Create a new strikethrough color attribute. This attribute modifies the color of strikethrough lines. If not set, strikethrough lines will use the foreground color.

**Parameters**

- **red** – the red value (ranging from 0 to 65535)
- **green** – the green value (ranging from 0 to 65535)
- **blue** – the blue value (ranging from 0 to 65535)
- **start_index** – the start index of the range. Should be >=0.
- **end_index** – end index of the range. The character at this index is not included in the range.

**Returns** the Attribute.

**Raises** AssertionError When `red` or `blue` or `green` isn't a `int`.

**classmethod from_underline**(*underline: pangocffi.enums.Underline*, *start_index: int = 0*, *end_index: int = 1*) → pangocffi.attribute.Attribute

Create a new underline-style attribute.

**Parameters**

- **underline** – the underline style.
- **start_index** – the start index of the range. Should be >=0.
- **end_index** – end index of the range. The character at this index is not included in the range.

**Returns** the Attribute.

**Raises** AssertionError When `underline` isn't a *`Underline`*.

**classmethod from_underline_color**(*red: int*, *green: int*, *blue: int*, *start_index: int = 0*, *end_index: int = 1*) → pangocffi.attribute.Attribute

Create a new underline color attribute. This attribute modifies the color of underlines. If not set, underlines will use the foreground color.

**Parameters**

- **red** – the red value (ranging from 0 to 65535)
- **green** – the green value (ranging from 0 to 65535)
- **blue** – the blue value (ranging from 0 to 65535)
- **start_index** – the start index of the range. Should be >=0.
- **end_index** – end index of the range. The character at this index is not included in the range.

**Returns** the Attribute.

**Raises** AssertionError When `red` or `blue` or `green` isn't a `int`.

**classmethod from_shape**(*ink_rect: pangocffi.rectangle.Rectangle*, *logical_rectangle: pangocffi.rectangle.Rectangle*, *start_index: int = 0*, *end_index: int = 1*) → pangocffi.attribute.Attribute

Create a new shape attribute. A shape is used to impose a particular ink and logical rectangle on the result

of shaping a particular glyph. This might be used, for instance, for embedding a picture or a widget inside a PangoLayout.

**Parameters**

- **ink_rect** – ink rectangle to assign to each character
- **logical_rectangle** – logical rectangle to assign to each character
- **start_index** – the start index of the range. Should be >=0.
- **end_index** – end index of the range. The character at this index is not included in the range.

**Returns** the Attribute.

**Raises** AssertionError When `ink_rect` or `logical_rectangle` isn't a *Rectangle*.

**classmethod from_scale**(*scale_factor: int*, *start_index: int = 0*, *end_index: int = 1*) → pangocffi.attribute.Attribute

Create a new font size scale attribute. The base font for the affected text will have its size multiplied by `scale_factor`.

**Parameters**

- **scale_factor** – factor to scale the font
- **start_index** – the start index of the range. Should be >=0.
- **end_index** – end index of the range. The character at this index is not included in the range.

**Returns** the Attribute.

**Raises** AssertionError When `scale_factor` isn't a `int`.

**classmethod from_rise**(*rise: int*, *start_index: int = 0*, *end_index: int = 1*) → pangocffi.attribute.Attribute

Create a new baseline displacement attribute.

**Parameters**

- **rise** – the amount that the text should be displaced vertically, in Pango units. Positive values displace the text upwards.
- **start_index** – the start index of the range. Should be >=0.
- **end_index** – end index of the range. The character at this index is not included in the range.

**Returns** the Attribute.

**Raises** AssertionError When `rise` isn't a `int`.

**classmethod from_letter_spacing**(*letter_spacing: int*, *start_index: int = 0*, *end_index: int = 1*) → pangocffi.attribute.Attribute

Create a new letter-spacing attribute.

**Parameters**

- **letter_spacing** – amount of extra space to add between graphemes of the text, in Pango units.
- **start_index** – the start index of the range. Should be >=0.
- **end_index** – end index of the range. The character at this index is not included in the range.

**Returns** the Attribute.

**Raises** AssertionError When `letter_spacing` isn't a `int`.

**classmethod from_fallback**(*enable_fallback: bool*, *start_index: int = 0*, *end_index: int = 1*) → pangocffi.attribute.Attribute

Create a new font fallback attribute.

If fallback is disabled, characters will only be used from the closest matching font on the system. No fallback will be done to other fonts on the system that might contain the characters in the text.

**Parameters**

- **enable_fallback** – True if we should fall back on other fonts for characters the active font is missing.
- **start_index** – the start index of the range. Should be >=0.
- **end_index** – end index of the range. The character at this index is not included in the range.

**Returns** the Attribute.

**Raises** AssertionError When `enable_fallback` isn't a `bool`.

**classmethod from_gravity**(*gravity: pangocffi.enums.Gravity*, *start_index: int = 0*, *end_index: int = 1*) → pangocffi.attribute.Attribute

Create a new gravity attribute.

**Parameters**

- **gravity** – the gravity value; should not be `PANGO_GRAVITY_AUTO`.
- **start_index** – the start index of the range. Should be >=0.
- **end_index** – end index of the range. The character at this index is not included in the range.

**Returns** the Attribute.

**Raises** AssertionError When `gravity` isn't a *Gravity*.

**classmethod from_gravity_hints**(*hint: pangocffi.enums.GravityHint*, *start_index: int = 0*, *end_index: int = 1*) → pangocffi.attribute.Attribute

Create a new gravity hint attribute.

**Parameters**

- **hint** – the gravity hint value from *GravityHint*
- **start_index** – the start index of the range. Should be >=0.
- **end_index** – end index of the range. The character at this index is not included in the range.

**Returns** the Attribute.

**Raises** AssertionError When `hint` isn't a *GravityHint*.

**classmethod from_font_features**(*features: str*, *start_index: int = 0*, *end_index: int = 1*) → pangocffi.attribute.Attribute

Create a new font features tag attribute.

**Parameters**

- **features** – a string with OpenType font features, in CSS syntax
- **start_index** – the start index of the range. Should be >=0.

- **end_index** – end index of the range. The character at this index is not included in the range.

**Returns** the Attribute.

**Raises** AssertionError When `features` isn't a `str`.

**classmethod from_foreground_alpha**(*alpha: int*, *start_index: int = 0*, *end_index: int = 1*) → pangocffi.attribute.Attribute

Create a new foreground alpha attribute.

**Parameters**

- **alpha** – the alpha value, between 1 and 65536
- **start_index** – the start index of the range. Should be >=0.
- **end_index** – end index of the range. The character at this index is not included in the range.

**Returns** the Attribute.

**Raises** AssertionError When `alpha` isn't a `int`.

**classmethod from_background_alpha**(*alpha: int*, *start_index: int = 0*, *end_index: int = 1*) → pangocffi.attribute.Attribute

Create a new background alpha attribute.

**Parameters**

- **alpha** – the alpha value, between 1 and 65536
- **start_index** – the start index of the range. Should be >=0.
- **end_index** – end index of the range. The character at this index is not included in the range.

**Returns** the Attribute.

**Raises** AssertionError When `alpha` isn't a `int`.

### AttrList

**class** pangocffi.**AttrList**(*pointer: _cffi_backend._CDataBase = None*, **init_args*)

The *AttrList* represents a list of attributes(*Attribute*) that apply to a section of text. The attributes are, in general, allowed to overlap in an arbitrary fashion, however, if the attributes are manipulated only through *AttrList.change()*, the overlap between properties will meet stricter criteria.

In general, you should not use a single *AttrList* for more than one paragraph of text due to internal structures.

**insert**(*attr: pangocffi.attribute.Attribute*) → None

Insert the given attribute into the PangoAttrList. It will be inserted after all other attributes with a matching `start_index`.

**Parameters** **attr** – The *Attribute* to insert.

**Raises** AssertionError When `attr` isn't a *Attribute*.

**insert_before**(*attr: pangocffi.attribute.Attribute*) → None

Insert the given attribute into the PangoAttrList. It will be inserted before all other attributes with a matching `start_index`.

**Parameters** **attr** – The *Attribute* to insert.

**Raises** AssertionError When `attr` isn't a *Attribute*.

**change** (*attr: pangocffi.attribute.Attribute*) → None
Insert the given attribute into the *AttrList*. It will replace any attributes of the same type on that segment and be merged with any adjoining attributes that are identical.

This function is slower than *AttrList.insert()* for creating an attribute list in order (potentially much slower for large lists). However, *AttrList.insert()* is not suitable for continually changing a set of attributes since it never removes or combines existing attributes.

**Parameters** **attr** – The *Attribute* to insert.

**Raises** AssertionError When `attr` isn't a *Attribute*.

**splice** (*attr_list: pangocffi.attr_list.AttrList*, *pos: int*, *length: int*)
This function opens up a hole in `self`, fills it in with attributes from the left, and then merges other on top of the hole. This operation is equivalent to stretching every attribute that applies at position pos in list by an amount len , and then calling *AttrList.change()* with a copy of each attribute in other in sequence (offset in position by pos ).

This operation proves useful for, for instance, inserting a pre-edit string in the middle of an edit buffer.

**Parameters**

- **attr_list** (AttrList) – another *AttrList*
- **pos** (*int*) – the position in `self` at which to insert other
- **length** (*int*) – the length of the spliced segment. (Note that this must be specified since the attributes in other may only be present at some subsection of this range)

**Raises** AssertionError When `attr_list` isn't a *AttrList*. When `pos` isn't a `int` When `length` isn't a `int`

### 1.2.5 Tab Stops

Structures for storing tab stops.

API not implemented yet.

### 1.2.6 Text Attribute Markup

See Gnome's documentation for details on how to use the Pango Text Attribute Markup Language.

### 1.2.7 Layout Objects

High-level layout driver objects.

#### Layout

**class** pangocffi.**Layout** (*context: pangocffi.context.Context*)
A Pango *Layout* represents an entire paragraph of text. It is initialized with a Pango *Context*, UTF-8 string and set of attributes for that string. Once that is done, the set of formatted lines can be extracted from the object, the layout can be rendered, and conversion between logical character positions within the layout's text, and the physical position of the resulting glyphs can be made.

**context**
The *Context* used for this layout.

**text**
The text contained in the layout.

Note that if you have used *apply_markup()* on the layout before, you may want to clear the *attributes* from the markup when setting this property, as attributes are not cleared automatically.

**font_description**
The default font description for the layout. If no font description is set, the font description from the layout's context is used.

**width**
The width to which the lines of the layout should wrap or ellipsize.

**height**
The height to which the layout should be ellipsized at.

If the height is positive, it will be the maximum height of the layout. Only lines which fit will be shown, and any omitted text is replaced by an ellipsis. At least one line is included in each paragraph regardless of how small the height value is; a value of zero will render exactly one line for the entire layout.

If height is negative, it will be the (negative of) maximum number of lines per paragraph. That is, the total number of lines shown may well be more than this value if the layout contains multiple paragraphs of text. The default value of -1 means that first line of each paragraph is ellipsized.

This property only has effect if a positive width is set on layout and its ellipsization mode is not NONE. The behavior is undefined if a height other than -1 is set and ellipsization mode is set to NONE.

**spacing**
The amount of spacing, in Pango units, between the lines of the layout.

**alignment**
The alignment of the layout: how partial lines are positioned within the horizontal space available.

**ellipsize**
The ellipsize mode of the layout.

**wrap**
The wrap mode of the layout.

**attributes**
The text attributes for this layout.

**Parameters** **attrs** – a *AttrList*

**apply_markup**(*markup: str*) → None
Sets the layout text and attribute list from marked-up text.

**Parameters** **markup** – marked-up text

**get_extents**() → Tuple[pangocffi.rectangle.Rectangle, pangocffi.rectangle.Rectangle]
Computes the logical and ink extents of the layout. Logical extents are usually what you want for positioning things. Note that both extents may have non-zero x and y. You may want to use those to offset where you render the layout. Not doing that is a very typical bug that shows up as right-to-left layouts not being correctly positioned in a layout with a set width.

The extents are given in layout coordinates and in Pango units; layout coordinates begin at the top left corner of the layout.

**Returns** a tuple containing two *Rectangle* objects. The first is the extent of the layout as drawn. The second is the logical extent of the layout.

**get_size**() → Tuple[int, int]
Determines the logical width and height of the layout in Pango units. This is simply a convenience function around *get_extents()*.

**Returns** a tuple containing the logical width and height, respectively.

**get_baseline**() → int
Returns the Y coordinate of the first line's baseline in the layout.

**Returns** baseline of the *Layout*'s first, line from the top

**get_line_count**() → int
Returns the number of lines.

**Returns** the line count

**get_iter**() → pangocffi.layout_iter.LayoutIter
Returns an iterator to iterate over the visual extents of the layout.

**Returns** the layout iterator

### Layout Iterator

**class** pangocffi.**LayoutIter**(*pointer: _cffi_backend._CDataBase = None*, **init_args*)
A *LayoutIter* can be used to iterate over the visual extents of a Pango *Layout*.

To obtain a *LayoutIter*, use *Layout.get_iter()*.

**next_run**() → bool
Moves the iterator forward to the next run in visual order. If the iterator was already at the end of the layout, it will return `False`.

**Returns** whether motion was possible

**next_char**() → bool
Moves the iterator forward to the next character in visual order. If the iterator was already at the end of the layout, it will return `False`.

**Returns** whether motion was possible

**next_cluster**() → bool
Moves the iterator forward to the next cluster in visual order. If the iterator was already at the end of the layout, it will return `False`.

**Returns** whether motion was possible

**next_line**() → bool
Moves the iterator forward to the start of the next line. If the iterator was already at the end of the layout, it will return `False`.

**Returns** whether motion was possible.

**at_last_line**() → bool
Determines whether the iterator is on the last line of the layout.

**Returns** whether the iterator is on the last line

**get_index**() → int
Returns the current byte index. Note that iterating forward by char moves in visual order, not logical order, so indexes may not be sequential. Also, the index may be equal to the length of the text in the layout, if on the NULL run (see *get_run()*).

**Returns** the current byte index

**get_baseline**() → int
Returns the Y position of the current line's baseline, in layout coordinates (origin at top left of the entire layout).

**Returns** the baseline of the current line

**get_run**() → Optional[pangocffi.layout_run.LayoutRun]
Returns the current run. When iterating by run, at the end of each line, there's a position with a NULL run, so this function can return `None`. The NULL run at the end of each line ensures that all lines have at least one run, even lines consisting of only a newline.

Use the faster `get_run_readonly()` if you do not plan to modify the contents of the run (glyphs, glyph widths, etc.).

**Returns** the current run

**get_char_extents**() → pangocffi.rectangle.Rectangle
Returns the extents of the current character, in layout coordinates (origin is the top left of the entire layout). Only logical extents can sensibly be obtained for characters; ink extents make sense only down to the level of clusters.

**Returns** a rectangle representing the logical extent of a character.

**get_cluster_extents**() → Tuple[pangocffi.rectangle.Rectangle, pangocffi.rectangle.Rectangle]
Returns the extents of the current cluster, in layout coordinates (origin is the top left of the entire layout).

**Returns** a tuple containing two *Rectangle* objects. The first is the extent of the cluster as drawn. The second is the logical extent.

**get_run_extents**() → Tuple[pangocffi.rectangle.Rectangle, pangocffi.rectangle.Rectangle]
Returns the extents of the current run, in layout coordinates (origin is the top left of the entire layout).

**Returns** a tuple containing two *Rectangle* objects. The first is the extent of the run as drawn. The second is the logical extent.

**get_line_yrange**() → Tuple[int, int]
Divides the vertical space in the *Layout* being iterated over between the lines in the layout, and returns the space belonging to the current line. A line's range includes the line's logical extents, plus half of the spacing above and below the line, if `Layout.set_spacing()` has been called to set layout spacing. The Y positions are in layout coordinates (origin at top left of the entire layout).

**Returns** a tuple containing two integers. The first is the y position of the start of the line. The second is the y position of the end of the line

**get_line_extents**() → Tuple[pangocffi.rectangle.Rectangle, pangocffi.rectangle.Rectangle]
Obtains the extents of the current line. Extents are in layout coordinates (origin is the top-left corner of the entire *Layout*). Thus the extents returned by this function will be the same width/height but not at the same x/y as the extents returned from `LayoutLine.get_extents()`.

**Returns** a tuple containing two *Rectangle* objects. The first is the extent of the line as drawn. The second is the logical extent.

**get_layout_extents**() → Tuple[pangocffi.rectangle.Rectangle, pangocffi.rectangle.Rectangle]
Returns the extents of the *Layout* being iterated over.

**Returns** a tuple containing two *Rectangle* objects. The first is the extent of the run as drawn. The second is the logical extent.

### Layout Line

API not implemented yet.

### Layout Run

**class** pangocffi.**LayoutRun**(*pointer: _cffi_backend._CDataBase = None*, **init_args*)

The *LayoutRun* structure represents a single run within a LayoutLine; it is simply an alternate name for *GlyphItem*. See the *GlyphItem* docs for details on the fields.

### Layout Modes

API not implemented yet.

### Wrap Mode

**class** pangocffi.**WrapMode**

*WrapMode* describes how to wrap the lines of a Pango layout to the desired width.

**WORD = 0**

Wrap lines at word boundaries

**CHAR = 1**

Wrap lines at character boundaries

**WORD_CHAR = 2**

Wrap lines at word boundaries, but fall back to character boundaries if there is not enough space for a full word.

### Ellipsize Mode

**class** pangocffi.**EllipsizeMode**

*EllipsizeMode* describes what sort of (if any) ellipsization should be applied to a line of text. In the ellipsization process characters are removed from the text in order to make it fit to a given width and replaced with an ellipsis.

**NONE = 0**

No ellipsization

**START = 1**

Omit characters at the start of the text

**MIDDLE = 2**

Omit characters in the middle of the text

**END = 3**

Omit characters at the end of the text

### Alignment

**class** pangocffi.**Alignment**

*Alignment* describes how to align the lines of a *Layout* within the available space. If the *Layout* is set to justify using set_justify(), this only has effect for partial lines.

**LEFT = 0**

Put all available space on the right

**CENTER = 1**

Center the line within the available space

**RIGHT = 2**
Put all available space on the left

### 1.2.8 Scripts and Languages

Identifying writing systems and languages.

#### Script

API not implemented yet.

#### Language

API not implemented yet.

### 1.2.9 Underlined Text

Enum used by Text Attributes for underlining text.

**class** pangocffi.**Underline**
*Underline* is used to specify whether text should be underlined, and if so, the type of underlining.

**NONE = 0**
no underline should be drawn

**SINGLE = 1**
a single underline should be drawn

**DOUBLE = 2**
a double underline should be drawn

**LOW = 3**
A single underline should be drawn at a position beneath the ink extents of the text being underlined. This should be used only for underlining single characters, such as for keyboard accelerators. PANGO_UNDERLINE_SINGLE should be used for extended portions of text.

**ERROR = 4**
a wavy underline should be drawn below. This underline is typically used to indicate an error such as a possible mispelling; in some cases a contrasting color may automatically be used. This type of underlining is available since Pango 1.4.

### 1.2.10 Bidirectional Text

Types and functions to help with handling bidirectional text.

API not implemented yet.

### 1.2.11 Vertical Text

Laying text out in vertical directions.

### Gravity

**class** pangocffi.**Gravity**

*Gravity* represents the orientation of glyphs in a segment of text. This is useful when rendering vertical text layouts. In those situations, the layout is rotated using a non-identity `Matrix`, and then glyph orientation is controlled using *Gravity*. Not every value in this enumeration makes sense for every usage of *Gravity*; for example, `Gravity.AUTO` only can be passed to `Context.set_base_gravity()` and can only be returned by `Context.get_base_gravity()`.

**SOUTH = 0**
Glyphs stand upright (default)

**EAST = 1**
Glyphs are rotated 90 degrees clockwise

**NORTH = 2**
Glyphs are upside-down

**WEST = 3**
Glyphs are rotated 90 degrees counter-clockwise

**AUTO = 4**
Gravity is resolved from the context matrix

### Gravity Hints

**class** pangocffi.**GravityHint**

*GravityHint* defines how horizontal scripts should behave in a vertical context. That is, English excerpt in a vertical paragraph for example.

See *Gravity*.

**NATURAL = 0**
scripts will take their natural gravity based on the base gravity and the script. This is the default.

**STRONG = 1**
always use the base gravity set, regardless of the script.

**LINE = 2**
for scripts not in their natural direction (eg. Latin in East gravity), choose per-script gravity such that every script respects the line progression. This means, Latin and Arabic will take opposite gravities and both flow top-to-bottom for example.

## 1.2.12 Low Level Functionality

### Version Checking

pangocffi.**pango_version**() → int

Return the pango version number as a single integer, such as 14204 for `1.42.4`. Major, minor and micro versions are "worth" 10000, 100 and 1 respectively.

Can be useful as a guard for method not available in older pango versions:

```
if pango_version() >= 11000:
    ...
```

pangocffi.**pango_version_string**() → str

Return the pango version number as a string, such as `1.42.4`.

#### Pango Object

**class** pangocffi.**PangoObject**(*pointer: _cffi_backend._CDataBase = None*, **init_args*)

An **:external:class:'AbstractBaseClass <abc.ABC>'** for every object used by Pango.

**pointer**

The C pointer to this object.

**classmethod from_pointer**(*pointer: _cffi_backend._CDataBase*, *gc: bool = False*) → pangocffi.pango_object.PangoObject

Instantiates an object from a C pointer.

**Parameters**

- **pointer** – a C pointer to the object.
- **gc** – whether to garbage collect the pointer. Defaults to `False`.

**Returns** the object.

## 1.3 CFFI API

pangocffi's *API* is made of a number of wrapper classes that provide a more Pythonic interface for various pango objects. Functions that take a pointer as their first argument become methods.

In order to use other C libraries that integrate with pango, or if pangocffi's API is not sufficient, you can access pango's lower level C pointers and API through CFFI.

### 1.3.1 Module-level objects

pangocffi.**ffi**

A `cffi.FFI` instance with all of the pango C API declared.

pangocffi.**pango**

The libpango library, pre-loaded with `ffi.dlopen()`. All pango functions are accessible as attributes of this object:

```
from pangocffi import Context, Layout
from pangocffi import pango as pango_c

context = Context()
layout = Layout(context)

pango_c.pango_renderer_draw_layout(..., layout.get_pointer(), 0, 0)
```

See the pango reference manual for details.

## 1.4 Binding progress

Below is a list of bindings that have (denoted with a ✓) and have not been implemented.

### 1.4.1 Basic Pango Interfaces

- pango_itemize ()
- pango_itemize_with_base_dir ()
- ✓ pango_item_free ()
- ✓ pango_item_copy ()
- pango_item_new ()
- pango_item_split ()
- pango_reorder_items ()
- ✓ pango_context_new ()
- pango_context_changed ()
- pango_context_get_serial ()
- pango_context_set_font_map ()
- pango_context_get_font_map ()
- ✓ pango_context_get_font_description ()
- ✓ pango_context_set_font_description ()
- pango_context_get_language ()
- pango_context_set_language ()
- pango_context_get_base_dir ()
- pango_context_set_base_dir ()
- ✓ pango_context_get_base_gravity ()
- ✓ pango_context_set_base_gravity ()
- ✓ pango_context_get_gravity ()
- ✓ pango_context_get_gravity_hint ()
- ✓ pango_context_set_gravity_hint ()
- pango_context_get_matrix ()
- pango_context_set_matrix ()
- pango_context_load_font ()
- pango_context_load_fontset ()
- pango_context_get_metrics ()
- pango_context_list_families ()
- pango_break ()
- pango_get_log_attrs ()
- pango_find_paragraph_boundary ()
- pango_default_break ()
- pango_shape ()
- pango_shape_full ()

- ✓ PangoContext
- ✓ PangoItem
- PangoAnalysis
- PANGO_ANALYSIS_FLAG_CENTERED_BASELINE
- PANGO_ANALYSIS_FLAG_IS_ELLIPSIS
- PANGO_TYPE_DIRECTION
- PangoLogAttr
- PANGO_PIXELS()
- PANGO_PIXELS_FLOOR()
- PANGO_PIXELS_CEIL()
- PANGO_UNITS_ROUND()
- ✓ pango_units_to_double ()
- ✓ pango_units_from_double ()
- PANGO_ASCENT()
- PANGO_DESCENT()
- PANGO_LBEARING()
- PANGO_RBEARING()
- pango_extents_to_pixels ()
- pango_matrix_copy ()
- pango_matrix_free ()
- pango_matrix_translate ()
- pango_matrix_scale ()
- pango_matrix_rotate ()
- pango_matrix_concat ()
- pango_matrix_transform_point ()
- pango_matrix_transform_distance ()
- pango_matrix_transform_rectangle ()
- pango_matrix_transform_pixel_rectangle ()
- pango_matrix_get_font_scale_factor ()
- pango_matrix_get_font_scale_factors ()
- PANGO_GET_UNKNOWN_GLYPH()
- pango_glyph_string_new ()
- pango_glyph_string_copy ()
- pango_glyph_string_set_size ()
- pango_glyph_string_free ()
- pango_glyph_string_extents ()

- pango_glyph_string_extents_range ()
- pango_glyph_string_get_width ()
- pango_glyph_string_index_to_x ()
- pango_glyph_string_x_to_index ()
- pango_glyph_string_get_logical_widths ()
- ✓ pango_glyph_item_copy ()
- ✓ pango_glyph_item_free ()
- ✓ pango_glyph_item_split ()
- pango_glyph_item_apply_attrs ()
- pango_glyph_item_letter_space ()
- ✓ pango_glyph_item_get_logical_widths ()
- pango_glyph_item_iter_copy ()
- pango_glyph_item_iter_free ()
- ✓ pango_glyph_item_iter_init_start ()
- ✓ pango_glyph_item_iter_init_end ()
- ✓ pango_glyph_item_iter_next_cluster ()
- ✓ pango_glyph_item_iter_prev_cluster ()
- PANGO_SCALE
- ✓ PangoRectangle
- PangoMatrix
- PANGO_TYPE_MATRIX
- PANGO_MATRIX_INIT
- PangoGlyph
- PANGO_GLYPH_EMPTY
- PANGO_GLYPH_INVALID_INPUT
- PANGO_GLYPH_UNKNOWN_FLAG
- PangoGlyphInfo
- PangoGlyphGeometry
- PangoGlyphUnit
- PangoGlyphVisAttr
- PangoGlyphString
- PangoGlyphItem
- PangoGlyphItemIter
- PANGO_TYPE_GLYPH_STRING
- PANGO_TYPE_GLYPH_ITEM
- PANGO_TYPE_GLYPH_ITEM_ITER

- ✓ pango_font_description_new ()
- ✓ pango_font_description_copy ()
- pango_font_description_copy_static ()
- pango_font_description_hash ()
- pango_font_description_equal ()
- ✓ pango_font_description_free ()
- pango_font_descriptions_free ()
- ✓ pango_font_description_set_family ()
- pango_font_description_set_family_static ()
- ✓ pango_font_description_get_family ()
- ✓ pango_font_description_set_style ()
- ✓ pango_font_description_get_style ()
- ✓ pango_font_description_set_variant ()
- ✓ pango_font_description_get_variant ()
- ✓ pango_font_description_set_weight ()
- ✓ pango_font_description_get_weight ()
- ✓ pango_font_description_set_stretch ()
- ✓ pango_font_description_get_stretch ()
- ✓ pango_font_description_set_size ()
- ✓ pango_font_description_get_size ()
- ✓ pango_font_description_set_absolute_size ()
- ✓ pango_font_description_get_size_is_absolute ()
- ✓ pango_font_description_set_gravity ()
- ✓ pango_font_description_get_gravity ()
- pango_font_description_get_set_fields ()
- pango_font_description_unset_fields ()
- pango_font_description_merge ()
- pango_font_description_merge_static ()
- pango_font_description_better_match ()
- pango_font_description_from_string ()
- pango_font_description_to_string ()
- pango_font_description_to_filename ()
- pango_font_metrics_ref ()
- pango_font_metrics_unref ()
- pango_font_metrics_get_ascent ()
- pango_font_metrics_get_descent ()

- pango_font_metrics_get_approximate_char_width ()
- pango_font_metrics_get_approximate_digit_width ()
- pango_font_metrics_get_underline_thickness ()
- pango_font_metrics_get_underline_position ()
- pango_font_metrics_get_strikethrough_thickness ()
- pango_font_metrics_get_strikethrough_position ()
- PANGO_FONT()
- PANGO_IS_FONT()
- pango_font_find_shaper ()
- pango_font_describe ()
- pango_font_describe_with_absolute_size ()
- pango_font_get_coverage ()
- pango_font_get_glyph_extents ()
- pango_font_get_metrics ()
- pango_font_get_font_map ()
- PANGO_FONT_FAMILY()
- PANGO_IS_FONT_FAMILY()
- pango_font_family_get_name ()
- pango_font_family_is_monospace ()
- pango_font_family_list_faces ()
- PANGO_FONT_FACE()
- PANGO_IS_FONT_FACE()
- pango_font_face_get_face_name ()
- pango_font_face_list_sizes ()
- pango_font_face_describe ()
- pango_font_face_is_synthesized ()
- PANGO_FONT_MAP()
- PANGO_IS_FONT_MAP()
- PANGO_FONT_MAP_CLASS()
- PANGO_IS_FONT_MAP_CLASS()
- PANGO_FONT_MAP_GET_CLASS()
- pango_font_map_create_context ()
- pango_font_map_load_font ()
- pango_font_map_load_fontset ()
- pango_font_map_list_families ()
- pango_font_map_get_shape_engine_type ()

- pango_font_map_get_serial ()
- pango_font_map_changed ()
- pango_fontset_get_font ()
- pango_fontset_get_metrics ()
- (* PangoFontsetForeachFunc) ()
- pango_fontset_foreach ()
- pango_fontset_simple_new ()
- pango_fontset_simple_append ()
- pango_fontset_simple_size ()
- PangoFontDescription
- PANGO_TYPE_FONT_DESCRIPTION
- ✓ PangoStyle
- PANGO_TYPE_STYLE
- ✓ PangoWeight
- PANGO_TYPE_WEIGHT
- ✓ PangoVariant
- PANGO_TYPE_VARIANT
- ✓ PangoStretch
- PANGO_TYPE_STRETCH
- ✓ PangoFontMask
- PANGO_TYPE_FONT_MASK
- PangoFontMetrics
- PANGO_TYPE_FONT_METRICS
- PangoFont
- PANGO_TYPE_FONT
- PangoFontFamily
- PANGO_TYPE_FONT_FAMILY
- PangoFontFace
- PANGO_TYPE_FONT_FACE
- PangoFontMap
- PANGO_TYPE_FONT_MAP
- PangoFontMapClass
- PangoFontset
- PANGO_TYPE_FONTSET
- PangoFontsetClass
- PangoFontsetSimple

- PANGO_TYPE_FONTSET_SIMPLE
- pango_parse_markup ()
- pango_markup_parser_new ()
- pango_markup_parser_finish ()
- pango_attr_type_register ()
- pango_attr_type_get_name ()
- pango_attribute_init ()
- ✓ pango_attribute_copy ()
- ✓ pango_attribute_equal ()
- ✓ pango_attribute_destroy ()
- pango_attr_language_new ()
- ✓ pango_attr_family_new ()
- ✓ pango_attr_style_new ()
- ✓ pango_attr_variant_new ()
- ✓ pango_attr_stretch_new ()
- ✓ pango_attr_weight_new ()
- ✓ pango_attr_size_new ()
- ✓ pango_attr_size_new_absolute ()
- ✓ pango_attr_font_desc_new ()
- ✓ pango_attr_foreground_new ()
- ✓ pango_attr_background_new ()
- ✓ pango_attr_strikethrough_new ()
- ✓ pango_attr_strikethrough_color_new ()
- ✓ pango_attr_underline_new ()
- ✓ pango_attr_underline_color_new ()
- ✓ pango_attr_shape_new ()
- pango_attr_shape_new_with_data ()
- (* PangoAttrDataCopyFunc) ()
- ✓ pango_attr_scale_new ()
- ✓ pango_attr_rise_new ()
- ✓ pango_attr_letter_spacing_new ()
- ✓ pango_attr_fallback_new ()
- ✓ pango_attr_gravity_new ()
- ✓ pango_attr_gravity_hint_new ()
- ✓ pango_attr_font_features_new ()
- ✓ pango_attr_foreground_alpha_new ()

- ✓ pango_attr_background_alpha_new ()
- ✓ pango_color_parse ()
- ✓ pango_color_copy ()
- ✓ pango_color_free ()
- ✓ pango_color_to_string ()
- ✓ pango_attr_list_new ()
- ✓ pango_attr_list_ref ()
- ✓ pango_attr_list_unref ()
- ✓ pango_attr_list_copy ()
- ✓ pango_attr_list_insert ()
- ✓ pango_attr_list_insert_before ()
- ✓ pango_attr_list_change ()
- ✓ pango_attr_list_splice ()
- pango_attr_list_filter ()
- (* PangoAttrFilterFunc) ()
- pango_attr_list_get_iterator ()
- pango_attr_iterator_copy ()
- pango_attr_iterator_next ()
- pango_attr_iterator_range ()
- pango_attr_iterator_get ()
- pango_attr_iterator_get_font ()
- pango_attr_iterator_get_attrs ()
- pango_attr_iterator_destroy ()
- PangoAttrType
- PANGO_TYPE_ATTR_TYPE
- PangoAttrClass
- PangoAttribute
- PANGO_ATTR_INDEX_FROM_TEXT_BEGINNING
- PANGO_ATTR_INDEX_TO_TEXT_END
- PangoAttrString
- PangoAttrLanguage
- PangoAttrColor
- PangoAttrInt
- PangoAttrFloat
- PangoAttrFontDesc
- PangoAttrShape

- PangoAttrSize
- PangoAttrFontFeatures
- PangoUnderline
- PANGO_TYPE_UNDERLINE
- PANGO_SCALE_XX_SMALL
- PANGO_SCALE_X_SMALL
- PANGO_SCALE_SMALL
- PANGO_SCALE_MEDIUM
- PANGO_SCALE_LARGE
- PANGO_SCALE_X_LARGE
- PANGO_SCALE_XX_LARGE
- PangoColor
- PANGO_TYPE_COLOR
- PangoAttrList
- PANGO_TYPE_ATTR_LIST
- PangoAttrIterator
- pango_tab_array_new ()
- pango_tab_array_new_with_positions ()
- pango_tab_array_copy ()
- pango_tab_array_free ()
- pango_tab_array_get_size ()
- pango_tab_array_resize ()
- pango_tab_array_set_tab ()
- pango_tab_array_get_tab ()
- pango_tab_array_get_tabs ()
- pango_tab_array_get_positions_in_pixels ()
- PangoTabArray
- PANGO_TYPE_TAB_ARRAY
- PangoTabAlign
- PANGO_TYPE_TAB_ALIGN
- ✓ pango_layout_new ()
- pango_layout_copy ()
- ✓ pango_layout_get_context ()
- pango_layout_context_changed ()
- pango_layout_get_serial ()
- ✓ pango_layout_set_text ()

- ✓ pango_layout_get_text ()
- pango_layout_get_character_count ()
- ✓ pango_layout_set_markup ()
- pango_layout_set_markup_with_accel ()
- ✓ pango_layout_set_attributes ()
- ✓ pango_layout_get_attributes ()
- ✓ pango_layout_set_font_description ()
- ✓ pango_layout_get_font_description ()
- ✓ pango_layout_set_width ()
- ✓ pango_layout_get_width ()
- ✓ pango_layout_set_height ()
- ✓ pango_layout_get_height ()
- ✓ pango_layout_set_wrap ()
- ✓ pango_layout_get_wrap ()
- pango_layout_is_wrapped ()
- ✓ pango_layout_set_ellipsize ()
- ✓ pango_layout_get_ellipsize ()
- pango_layout_is_ellipsized ()
- pango_layout_set_indent ()
- pango_layout_get_indent ()
- ✓ pango_layout_get_spacing ()
- ✓ pango_layout_set_spacing ()
- pango_layout_set_justify ()
- pango_layout_get_justify ()
- pango_layout_set_auto_dir ()
- pango_layout_get_auto_dir ()
- ✓ pango_layout_set_alignment ()
- ✓ pango_layout_get_alignment ()
- pango_layout_set_tabs ()
- pango_layout_get_tabs ()
- pango_layout_set_single_paragraph_mode ()
- pango_layout_get_single_paragraph_mode ()
- pango_layout_get_unknown_glyphs_count ()
- pango_layout_get_log_attrs ()
- pango_layout_get_log_attrs_readonly ()
- pango_layout_index_to_pos ()

- pango_layout_index_to_line_x ()
- pango_layout_xy_to_index ()
- pango_layout_get_cursor_pos ()
- pango_layout_move_cursor_visually ()
- pango_layout_get_extents ()
- pango_layout_get_pixel_extents ()
- ✓ pango_layout_get_size ()
- pango_layout_get_pixel_size ()
- ✓ pango_layout_get_baseline ()
- ✓ pango_layout_get_line_count ()
- pango_layout_get_line ()
- pango_layout_get_line_readonly ()
- pango_layout_get_lines ()
- pango_layout_get_lines_readonly ()
- ✓ pango_layout_get_iter ()
- ✓ pango_layout_iter_copy ()
- ✓ pango_layout_iter_free ()
- ✓ pango_layout_iter_next_run ()
- ✓ pango_layout_iter_next_char ()
- ✓ pango_layout_iter_next_cluster ()
- ✓ pango_layout_iter_next_line ()
- ✓ pango_layout_iter_at_last_line ()
- ✓ pango_layout_iter_get_index ()
- ✓ pango_layout_iter_get_baseline ()
- ✓ pango_layout_iter_get_run ()
- pango_layout_iter_get_run_readonly ()
- pango_layout_iter_get_line ()
- pango_layout_iter_get_line_readonly ()
- pango_layout_iter_get_layout ()
- ✓ pango_layout_iter_get_char_extents ()
- ✓ pango_layout_iter_get_cluster_extents ()
- ✓ pango_layout_iter_get_run_extents ()
- ✓ pango_layout_iter_get_line_yrange ()
- ✓ pango_layout_iter_get_line_extents ()
- ✓ pango_layout_iter_get_layout_extents ()
- pango_layout_line_ref ()

- pango_layout_line_unref ()
- pango_layout_line_get_extents ()
- pango_layout_line_get_pixel_extents ()
- pango_layout_line_index_to_x ()
- pango_layout_line_x_to_index ()
- pango_layout_line_get_x_ranges ()
- PangoLayout
- PangoLayoutIter
- ✓ PangoWrapMode
- ✓ PANGO_TYPE_WRAP_MODE
- ✓ PangoEllipsizeMode
- PANGO_TYPE_ELLIPSIZE_MODE
- ✓ PangoAlignment
- PANGO_TYPE_ALIGNMENT
- PangoLayoutLine
- PangoLayoutRun
- pango_script_for_unichar ()
- pango_script_get_sample_language ()
- pango_script_iter_new ()
- pango_script_iter_get_range ()
- pango_script_iter_next ()
- pango_script_iter_free ()
- pango_language_from_string ()
- pango_language_to_string ()
- pango_language_matches ()
- pango_language_includes_script ()
- pango_language_get_scripts ()
- pango_language_get_default ()
- pango_language_get_sample_string ()
- PangoScript
- PANGO_TYPE_SCRIPT
- PangoScriptIter
- PangoLanguage
- PANGO_TYPE_LANGUAGE
- pango_unichar_direction ()
- pango_find_base_dir ()

- pango_get_mirror_char ()
- pango_bidi_type_for_unichar ()
- PangoDirection
- PangoBidiType
- PANGO_GRAVITY_IS_IMPROPER()
- PANGO_GRAVITY_IS_VERTICAL()
- pango_gravity_get_for_matrix ()
- pango_gravity_get_for_script ()
- pango_gravity_get_for_script_and_width ()
- pango_gravity_to_rotation ()
- ✓ PangoGravity
- ✓ PangoGravityHint

### 1.4.2 Low Level functionality

- PANGO_VERSION_ENCODE()
- PANGO_VERSION_CHECK()
- ✓ pango_version ()
- ✓ pango_version_string ()
- pango_version_check ()
- PANGO_VERSION
- PANGO_VERSION_MAJOR
- PANGO_VERSION_MINOR
- PANGO_VERSION_MICRO
- PANGO_VERSION_STRING

## 1.5 Changelog

See the Releases section on GitHub for a description of changes between each version.

## 1.6 Contributing

See CONTRIBUTING.md on GitHub for information on how to contribute!

# Python Module Index

## p

# Index

## G



## H



## I



## L



## M



## N



## O



## P



## R