pangocairocffi

Release 0.7.0

Oct 07, 2022

Contents

pangocairocffi is a CFFI-based set of Python bindings for the cairo rendering methods with pango. It is meant to be used in conjunction with cairocffi and pangocffi.

CHAPTER 1

Documentation

1.1 Overview

1.1.1 Installing

To get started, there are multiple dependencies that require installation.

Installing cairocffi and pangocffi

Follow the instructions as provided by these dependencies:

- Installing cairocffi
- Installing pangocffi

Installing pangocairocffi

Install with pip:

```
pip install pangocairocffi
```

Note: Python versions < 3.6 are not supported.

1.1.2 Importing pangocairocffi

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

```
import pangocairocffi as pangocairo
```

pangocairocffi will dynamically load PangoCairo as a shared library upon importing. If it fails to find it, you will see an exception like this:

```
OSError: dlopen() failed to load pangocairo: pangocairo-1.0 / pangocairo-1.0.0
```

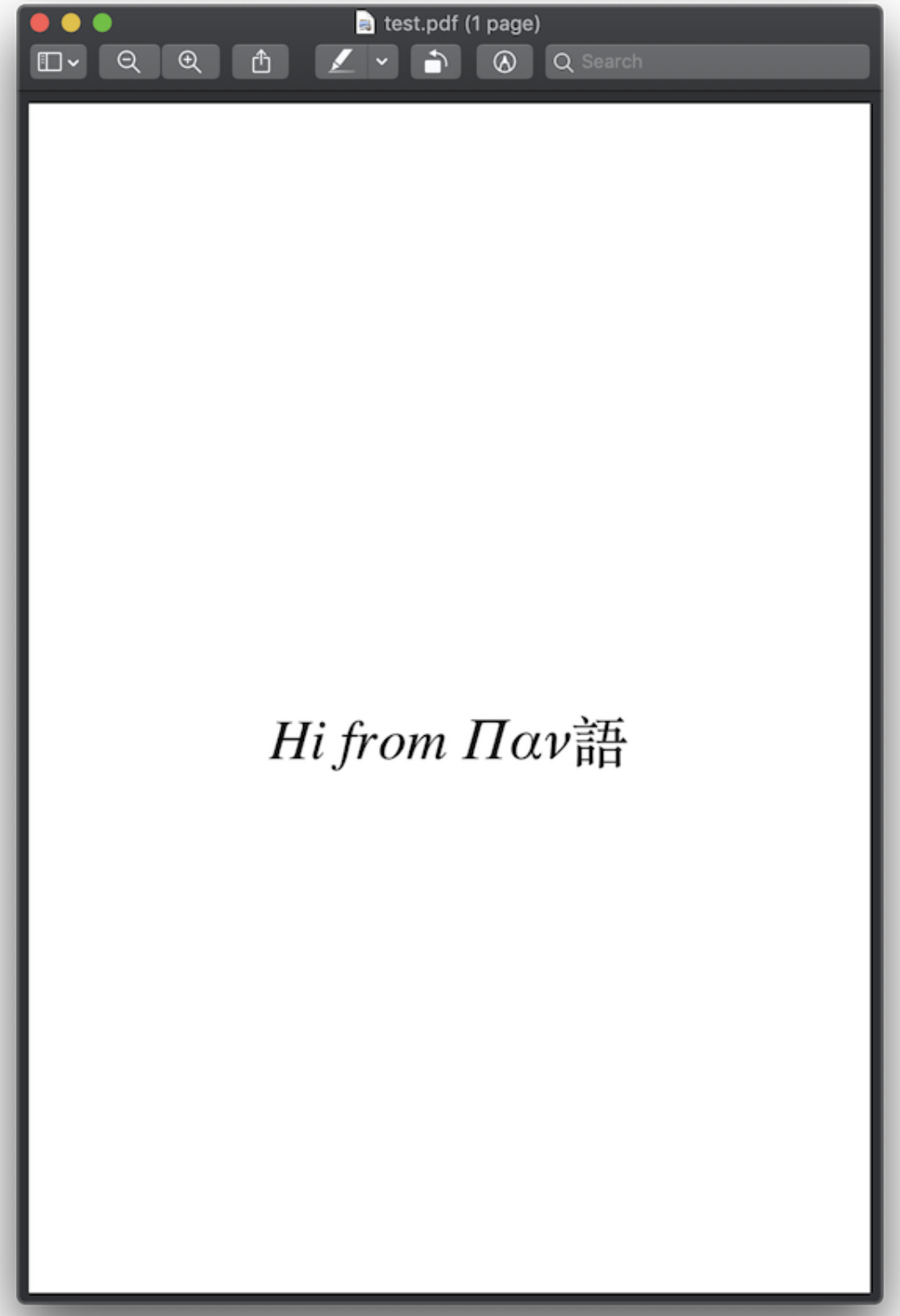
If PangoCairo is not installed as a shared library, pangocairocffi supports specifying a path via an environment variable: `PANGOCAIRO_LOCATION`. Note that the loading of dynamic libraries also applies to pangocffi, so be sure to check Importing pangocffi as well for information on how to specify paths to Pango, GLib, and GObject.

1.1.3 Basic usage and example

Below is a rough example of how to use pangocairocffi together with pangocffi and cairocffi:

```
import cairocffi
import pangocffi
import pangocairocffi

# Create the surface and get the context
filename = 'test.pdf'
pt_per_mm = 72 / 25.4
width, height = 210 * pt_per_mm, 297 * pt_per_mm  # A4 portrait
surface = cairocffi.PDFSurface(filename, width, height)
context = cairocffi.Context(surface)

context.translate(0, height / 2)

# Build the layout
layout = pangocairocffi.create_layout(context)
layout.set_width(pangocffi.units_from_double(width))
layout.set_alignment(pangocffi.Alignment.CENTER)
layout.set_markup('<span font="italic 30">Hi from Παν</span>')

# Render the layout
pangocairocffi.show_layout(context, layout)

# Output the surface
surface.finish()
```

Which produces the following output:


test.pdf (1 page)
Search
Hi from Παν語

1.2 Python API reference

1.2.1 Creating and updating Pango objects from Cairo

Contexts

pangocairocffi.**create_context**(*cairo_context: cairocffi.context.Context*) → pangocffi.context.Context

Creates a context object set up to match the current transformation and target surface of the Cairo context. This context can then be used to create a layout using `pangocffi.Layout()`.

This function is a convenience function that creates a context using the default font map, then updates it to `cairo_context`. If you just need to create a layout for use with `cairo_context` and do not need to access PangoContext directly, you can use *`create_layout()`* instead.

Parameters **cairo_context** – a Cairo context

Returns a Pango context

pangocairocffi.**update_context**(*cairo_context: cairocffi.context.Context*, *pango_context: pangocffi.context.Context*) → None

Updates a PangoContext previously created for use with Cairo to match the current transformation and target surface of a Cairo context. If any layouts have been created for the context, it's necessary to call pango_layout_context_changed() on those layouts.

Parameters

- **cairo_context** – a Cairo context
- **pango_context** – a Pango context, from a pango-cairo font map

Layouts

pangocairocffi.**create_layout**(*cairo_context: cairocffi.context.Context*) → pangocffi.layout.Layout

Creates a layout object set up to match the current transformation and target surface of the Cairo context. This layout can then be used for text measurement with functions like `get_size()` or drawing with functions like *`show_layout()`*. If you change the transformation or target surface for `cairo_context`, you need to call *`update_layout()`*

This function is the most convenient way to use Cairo with Pango, however it is slightly inefficient since it creates a separate PangoContext object for each layout. This might matter in an application that was laying out large amounts of text.

Parameters **cairo_context** – a Cairo context

Returns a Pango layout

pangocairocffi.**update_layout**(*cairo_context: cairocffi.context.Context*, *layout: pangocffi.layout.Layout*) → None

Updates the private Pango `Context` of a Pango `Layout` created with *`create_layout()`* to match the current transformation and target surface of a Cairo context.

Parameters

- **cairo_context** – a Cairo context
- **layout** – a Pango layout

1.2.2 Rendering Pango objects with Cairo

Drawing on the cairo context

pangocairocffi.**show_layout**(*cairo_context: cairocffi.context.Context*, *layout: pangocffi.layout.Layout*) → None

Draws a Pango Layout in the specified cairo context. The top-left corner of the PangoLayout will be drawn at the current point of the cairo context.

Parameters

- **cairo_context** – a Cairo context
- **layout** – a Pango layout

pangocairocffi.**show_error_underline**(*cairo_context: cairocffi.context.Context*, *x: float*, *y: float*, *width: float*, *height: float*) → None

Draw a squiggly line in the specified cairo context that approximately covers the given rectangle in the style of an underline used to indicate a spelling error. (The width of the underline is rounded to an integer number of up/down segments and the resulting rectangle is centered in the original rectangle)

Parameters

- **cairo_context** – a Cairo context
- **x** – The X coordinate of one corner of the rectangle
- **y** – The Y coordinate of one corner of the rectangle
- **width** – Non-negative width of the rectangle
- **height** – Non-negative height of the rectangle

pangocairocffi.**show_glyph_item**(*cairo_context: cairocffi.context.Context*, *text: str*, *glyph_item: pangocffi.glyph_item.GlyphItem*) → None

Draws the glyphs in `glyph_item` in the specified cairo context, embedding the text associated with the glyphs in the output if the output format supports it (PDF for example), otherwise it acts similar to `show_glyph_string()`.

The origin of the glyphs (the left edge of the baseline) will be drawn at the current point of the cairo context.

Note that `text` is the start of the text for layout, which is then indexed by `glyph_item->item->offset`.

Parameters

- **cairo_context** – a Cairo context
- **text** – the UTF-8 text that `glyph_item` refers to
- **glyph_item** – a Pango glyph item

Adding text to cairo's current path

pangocairocffi.**layout_path**(*cairo_context: cairocffi.context.Context*, *layout: pangocffi.layout.Layout*) → None

Adds the text in a `Pango.Layout` to the current path in the specified cairo context. The top-left corner of the `Pango.Layout` will be at the current point of the cairo context.

Parameters

- **cairo_context** – a Cairo context
- **layout** – a Pango layout

pangocairocffi.**error_underline_path**(*cairo_context: cairocffi.context.Context*, *x: float*, *y: float*, *width: float*, *height: float*) → None

Add a squiggly line to the current path in the specified cairo context that approximately covers the given rectangle in the style of an underline used to indicate a spelling error. (The width of the underline is rounded to an integer number of up/down segments and the resulting rectangle is centered in the original rectangle)

Parameters

- **cairo_context** – a Cairo context
- **x** – The X coordinate of one corner of the rectangle
- **y** – The Y coordinate of one corner of the rectangle
- **width** – Non-negative width of the rectangle
- **height** – Non-negative height of the rectangle

1.2.3 PangoCairo Fonts

PangoCairo Font Functions

pangocairocffi.**set_resolution**(*context: pangocffi.context.Context*, *dpi: float*) → None

Sets the resolution for the context. This is a scale factor between points specified in a PangoFontDescription and Cairo units. The default value is 96, meaning that a 10 point font will be 13 units high. (10 * 96. / 72. = 13.3).

Parameters

- **context** – a Pango context
- **dpi** – the resolution in "dots per inch". (Physical inches aren't actually involved; the terminology is conventional.) A 0 or negative value means to use the resolution from the font map.

pangocairocffi.**get_resolution**(*context: pangocffi.context.Context*) → float

Returns the resolution for the Pango context.

Parameters **context** – a Pango context

Returns the resolution in "dots per inch". A negative value will be returned if no resolution has previously been set.

pangocairocffi.**set_font_options**(*context: pangocffi.context.Context*, *options: Optional[_cffi_backend._CDataBase]*) → None

Sets the font options used when rendering text with this context. These options override any options that pango_cairo_update_context() derives from the target surface.

Parameters

- **context** – a Pango context
- **options** – a cairo_font_options_t, or `None` to unset any previously set options.

pangocairocffi.**get_font_options**(*context: pangocffi.context.Context*) → Optional[_cffi_backend._CDataBase]

Retrieves any font rendering options previously set with pango_cairo_context_set_font_options(). This function does not report options that are derived from the target surface by pango_cairo_update_context()

Parameters **context** – a Pango Context

Returns a cairo_font_options_t pointer previously set on the context, otherwise `None`.

PangoCairo Font Map

class pangocairocffi.**PangoCairoFontMap**

API not *fully* implemented yet.

Todo: This class should extend FontMap from pangocffi once it is implemented. This class in theory should be able to inherit the functions listed here: https://developer.gnome.org/pango/stable/pango-Fonts.html, with the prefix `pango_font_map_X`. For example: `create_context`, `load_font`, `load_fontset`, `list_families`.

`PangoCairoFontMap` is an interface exported by font maps for use with Cairo. The actual type of the font map will depend on the particular font technology Cairo was compiled to use.

`__init__` Creates a new PangoCairoFontMap object; a `fontmap` is used to cache information about available fonts, and holds certain global parameters such as the resolution. In most cases, you can use *`PangoCairoFontMap.get_default()`* instead.

Note that the type of the returned object will depend on the particular font backend Cairo was compiled to use; You generally should only use the `PangoFontMap` and `PangoCairoFontMap` interfaces on the returned object.

You can override the type of backend returned by using an environment variable `PANGOCAIRO_BACKEND`. Supported types, based on your build, are fc (fontconfig), win32, and coretext. If requested type is not available, NULL is returned. I.E. this is only useful for testing, when at least two backends are compiled in.

pointer

Returns the pointer to the font map

Returns the pointer to the font map.

classmethod from_pointer(*pointer: _cffi_backend._CDataBase*) → pangocairocffi.font_map.PangoCairoFontMap

Instantiates a *`PangoCairoFontMap`* from a pointer.

Returns the pango-cairo font map.

classmethod get_default() → pangocairocffi.font_map.PangoCairoFontMap

Gets a default `PangoCairoFontMap` to use with Cairo.

Note that the type of the returned object will depend on the particular font backend Cairo was compiled to use; You generally should only use the PangoFontMap and PangoCairoFontMap interfaces on the returned object.

The default Cairo fontmap can be changed by using *`PangoCairoFontMap.set_default()`*. This can be used to change the Cairo font backend that the default fontmap uses for example.

Note that since Pango 1.32.6, the default fontmap is per-thread. Each thread gets its own default fontmap. In this way, PangoCairo can be used safely from multiple threads.

Returns the default PangoCairo fontmap for the current thread. This object is owned by Pango and must not be freed.

classmethod set_default(*fontmap: Optional[PangoCairoFontMap] = None*) → None

Sets a default PangoCairoFontMap to use with Cairo.

This can be used to change the Cairo font backend that the default fontmap uses for example. The old default font map is unreffed and the new font map referenced.

Note that since Pango 1.32.6, the default fontmap is per-thread. This function only changes the default fontmap for the current thread. Default fontmaps of existing threads are not changed. Default fontmaps of any new threads will still be created using pango_cairo_font_map_new().

A value of `None` for fontmap will cause the current default font map to be released and a new default font map to be created on demand, using `pango_cairo_font_map_new()`.

Returns the pango-cairo font map.

classmethod from_cairo_font_type(*cairo_font_type_pointer: _cffi_backend._CDataBase*) → pangocairocffi.font_map.PangoCairoFontMap
Instantiates a *PangoCairoFontMap* from a Cairo context.

Returns the pango-cairo font map.

get_cairo_font_type_pointer() → _cffi_backend._CDataBase
Returns the pointer to the type of Cairo font backend that `fontmap` uses

Returns the pointer to the `cairo_font_type_t`.

resolution
The resolution for the `fontmap` in "dots per inch". This is a scale factor between points specified in a Pango FontDescription and Cairo units. The default value is 96, meaning that a 10 point font will be 13 units high. (10 * 96. / 72. = 13.3). (Physical inches aren't actually involved; the terminology is conventional.)

create_context() → pangocffi.context.Context
Creates a Pango `Context` connected to `fontmap`. This is equivalent to `pango.Context()` followed by `context.set_font_map()`.

Returns the newly allocated Pango `Context`.

1.3 Tests

1.3.1 test_end_to_end.py

1.3.2 test_error_underline.py

1.3.3 test_extents.py

1.3.4 test_glyph_item.py

1.4 Changelog

1.4.1 Version 0.6.0

Released on 2020-12-30.

- Added two new static methods to `PangoCairoFontMap`:
  - `set_default`
  - `get_default`

1.4.2 Version 0.5.0

Released on 2020-12-19.

- Added the ability to configure loading a specific library via the environment variable `PANGOCAIRO_LOCATION`.

1.4.3 Version 0.4.0

Released on 2020-11-13.

Breaking Changes

- C-FFI bindings are now generated at runtime, rather than at installation. This was done to avoid common installation issues like `ModuleNotFoundError: No module named 'pangocairocffi._generated'`. This change does mean the bindings are re-compiled the first time `import pangocairocffi` is called in a python session, but it takes less than a second to do this. If there are any issues with this change, please raise an issue in the issue tracker.
- pangocairocffi now depends on pangocffi v0.8.0. Older versions are not compatible.
- Support for Python 3.5 has been dropped because it has reached end-of-life.

1.4.4 Version 0.3.2

Released on 2020-10-06.

- Reverted previous unsuccessful patch.

1.4.5 Version 0.3.1

Released on 2020-10-06.

- Hopefully fixed issue with `pangocairocffi` not being cache-able as a wheel by pip.

1.4.6 Version 0.3.0

Released on 2019-10-08.

- Extended library names to include current Win64 library names.

1.4.7 Version 0.2.2 - 0.2.6

Released on 2019-03-23.

These were a series of changes that happened rapidly since the issues were not easy to replicate manually.

- Version 0.2.2

  - Upgraded dependency of `pangocffi` from 0.3.0 to 0.4.0.
  - Possible fix for SandboxViolation when installing via easy_install/ setuptools
- Version 0.2.3
  - Possible fix for installation issues related to `ffi_pango.py` not being installed in the correct directory, and also the `pangocffi` dependency not being installed in the setup process.
- Version 0.2.4
  - `ffi_pango.py` still was not being installed correctly. So now rather than relying on the file being generated, the file exists hardcoded in the repository.
- Version 0.2.5
  - setuptools apparently does not follow the `package_data` rule. A `MANIFEST.in` file has been added to fix this. (see https://stackoverflow.com/a/14159430)
- Version 0.2.6
  - `include_package_data` and `zip_safe` need to be set.

1.4.8 Version 0.2.2

Released on 2019-03-23.

1.4.9 Version 0.2.1

Released on 2019-03-21.

- Improved linting coverage by refactoring the cffi build script.

1.4.10 Version 0.2.0

Released on 2019-03-19.

- Upgraded dependency of `pangocffi` from 0.1.1 to 0.3.0. This gave us the ability to implement `show_glyph_item()`
- Replaced incorrect type hinting of `ctypes` with `ffi.CData`
- Added new tests, including examples in the documentation:
  - `test_error_underline.py`
  - `test_extents.py`
  - `test_glyph_item.py`

1.4.11 Version 0.1.0

Released on 2019-03-09.

First PyPI release.

Python Module Index

p

Index