ManimPango

Release 0.4.1

The Manim Community Dev Team

May 03, 2022

CONTENTS:

CHAPTER

ONE

MANIMPANGO REFERENCE

manimpango.TextSetting(start, end, font, ...)	Formatting for slices of a `manim.mobject.svg.text_mobject.Text` object.
manimpango.PangoUtils()	
manimpango.text2svg	Render an SVG file from a `manim.mobject.svg.text_mobject.Text` object.
manimpango.MarkupUtils()	
manimpango.register_font	This function registers the font file using `fontconfig` so that it is available for use by Pango.
manimpango.unregister_font	This function unregisters(removes) the font file using `fontconfig`.
manimpango.list_fonts	Lists the fonts available to Pango.

1.1 TextSetting

Qualified name: `manimpango.TextSetting`

class TextSetting(*start*, *end*, *font*, *slant*, *weight*, *line_num=- 1*, *color=None*)

Formatting for slices of a `manim.mobject.svg.text_mobject.Text` object.

Methods

Parameters

- **start** (*int*) –
- **end** (*int*) –
- **font** (*unicode*) –
- **color** (*unicode*) –

1.2 PangoUtils

Qualified name: `manimpango.PangoUtils`

class PangoUtils

Methods

remove_last_M	Remove element from the SVG file in order to allow comparison.
str2style	Internally used function.
str2weight	Internally used function.

static remove_last_M(*file_name*)

Remove element from the SVG file in order to allow comparison.

Parameters **file_name** (*str*) –

Return type None

static str2style(*string*)

Internally used function. Converts text to Pango Understandable Styles.

Parameters **string** (*str*) –

Return type manimpango.enums.Style

static str2weight(*string*)

Internally used function. Convert text to Pango Understandable Weight

Parameters **string** (*str*) –

Return type manimpango.enums.Weight

1.3 manimpango.text2svg

text2svg()

Render an SVG file from a `manim.mobject.svg.text_mobject.Text` object.

1.4 MarkupUtils

Qualified name: `manimpango.MarkupUtils`

class MarkupUtils

Methods

text2svg	Render an SVG file from a `manim.mobject.svg.text_mobject.MarkupText` object.
validate	Validates whether markup is a valid Markup and return the error's if any.

static text2svg(*text*, *font*, *slant*, *weight*, *size*, *_*, *disable_liga*, *file_name*, *START_X*, *START_Y*, *width*, *height*, ***, *justify=None*, *indent=None*, *line_spacing=None*, *alignment=None*, *pango_width=None*)

Render an SVG file from a `manim.mobject.svg.text_mobject.MarkupText` object.

Parameters

- **text** (*unicode*) –
- **font** (*unicode*) –
- **slant** (*unicode*) –
- **weight** (*unicode*) –
- **size** (*int*) –
- **_** (*int*) –
- **disable_liga** (*bool*) –
- **file_name** (*unicode*) –
- **START_X** (*int*) –
- **START_Y** (*int*) –
- **width** (*int*) –
- **height** (*int*) –
- **justify** (*bool*) –
- **indent** (*float*) –
- **line_spacing** (*float*) –
- **alignment** (*manimpango.enums.Alignment*) –
- **pango_width** (*Optional[int]*) –

Return type int

static validate(*markup*)

Validates whether markup is a valid Markup and return the error's if any.

Parameters **markup** (`str`) – The markup which should be checked.

Returns Returns empty string if markup is valid. If markup contains error it return the error message.

Return type `str`

1.5 manimpango.register_font

register_font()

This function registers the font file using `fontconfig` so that it is available for use by Pango. On Linux it is aliased to *register_font()* and on Windows and macOS this would work only when using `fontconfig` backend.

Parameters **font_path** (`str`) – Relative or absolute path to font file.

Returns True means it worked without any error. False means there was an unknown error

Return type `bool`

Examples

```
>>> register_font("/home/roboto.tff")
True
```

Raises `AssertionError` – Font is missing.

1.6 manimpango.unregister_font

unregister_font()

This function unregisters(removes) the font file using `fontconfig`. It is mostly optional to call this. Mainly used in tests. On Linux it is aliased to *unregister_font()* and on Windows and macOS this would work only when using `fontconfig` backend.

Parameters **font_path** (`str`) – For compatibility with the windows function.

Returns True means it worked without any error. False means there was an unknown error

Return type `bool`

1.7 manimpango.list_fonts

list_fonts()

Lists the fonts available to Pango. This is usually same as system fonts but it also includes the fonts added through *register_font()*.

Returns List of fonts sorted alphabetically.

Return type `list`

CHAPTER

TWO

TEXT ATTRIBUTES

manimpango.attributes

2.1 attributes

Classes

TextAttribute	*TextAttribute* defines the properties/attributes of the text within a specific range of the text.

2.1.1 TextAttribute

Qualified name: `manimpango.attributes.TextAttribute`

class TextAttribute(*start_index=0*, *end_index=- 1*)

TextAttribute defines the properties/attributes of the text within a specific range of the text.

A *TextAttribute* object can define multiple properties at the same time, for example, it can change the *background_color*, as well as, `foreground_color`. Also, a *TextAttribute* can be used for multiple times for different texts. By default, an attribute has an inclusive range from `0` to the end of the text `-1`, ie. `[0, -1]`.

Initialize *TextAttribute*.

Parameters

- **start_index** (*int, optional*) – The start index of the range, by default 0 (start of the string).
- **end_index** (*int, optional*) – End index of the range. The character at this index is not included in the range, by default -1 (end of the string).

Return type None

Methods

property allow_breaks: Optional[bool]

Whether to break text or not.

If breaks are disabled, the range will be kept in a single run, as far as possible.

property background_alpha: float

The background_alpha of the text.

Raises ValueError – If the value is not between 0 and 1.

property background_color: Optional[Tuple[int]]

The background color of the region.

If the input is a str the value is considered as string representation of color from CSS Specification. The color is then parsed and ValueError is raised if the color is invalid.

If the input is a collections.abc.Iterable then the items in them are parsed in the order of `red, green, blue` and checked whether they are valid (between 0 and 65535).

Returns either `None` or a tuple with 3 elements representing red, green, blue respectively. The value of each items in that tuple ranges from 0 to 65535.

Raises ValueError – If the value passed isn't a collections.abc.Iterable of 3 elements or a string. Another condition when *ValueError* is raised is when the color passed is invalid.

property end_index: int

It is the start of the range. The character at this index is not included in the range.

Raises ValueError – If the value is not an int.

property fallback: bool

Enable or disable fallbacks.

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.

property family: str

The font family the text should render. Can be a comma seperated list of fonts in a string.

Raises ValueError – If value isn't a str.

property start_index: int

It is the end index of the range.

Raises ValueError – If the value is not an int.

CHAPTER

THREE

FONT DESCRIPTION

manimpango.fonts.enums	Contains Enums which defines text properties from Pango.
manimpango.fonts.FontDescription([family, ...])	A `FontDescription` describes a font.

3.1 enums

Contains Enums which defines text properties from Pango.

Most of these are used in `FontDescription`.

Classes

Style	An enumeration specifying the various slant styles possible for a font.
Variant	An enumeration specifying capitalization variant of the font.
Weight	An enumeration specifying the weight (boldness) of a font.

3.1.1 Style

Qualified name: `manimpango.fonts.enums.Style`

class Style(*value*)

An enumeration specifying the various slant styles possible for a font.

NORMAL

the font is upright.

ITALIC

the font is slanted, but in a roman style.

OBLIQUE

the font is slanted in an italic style.

3.1.2 Variant

Qualified name: `manimpango.fonts.enums.Variant`

class Variant(*value*)

An enumeration specifying capitalization variant of the font.

NORMAL

A normal font.

SMALL_CAPS

A font with the lower case characters replaced by smaller variants of the capital characters.

3.1.3 Weight

Qualified name: `manimpango.fonts.enums.Weight`

class Weight(*value*)

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 Using numerical value other then that defined here is not supported.

NORMAL

the default weight (= 400)

BOLD

the bold weight(= 700)

THIN

the thin weight(= 100; Since: 1.24)

ULTRALIGHT

the ultralight weight(= 200)

LIGHT

the light weight(= 300)

BOOK

the book weight(= 380; Since: 1.24)

MEDIUM

the normal weight(= 500; Since: 1.24)

SEMIBOLD

the semibold weight(= 600)

ULTRABOLD

the ultrabold weight(= 800)

HEAVY

the heavy weight(= 900)

ULTRAHEAVY

the ultraheavy weight(= 1000; Since: 1.24)

3.2 FontDescription

Qualified name: `manimpango.fonts.FontDescription`

class FontDescription(*family=None*, *size=None*, *style=None*, *weight=None*, *variant=None*)

A *FontDescription* describes a font.

This describes the characteristics of a font to load.

Parameters

- **family** (*str*) – Sets *family*.
- **size** (*int*) – Sets *size*.
- **style** (Style) – Sets *style*.
- **weight** (Weight) – Sets *weight*.
- **variant** (Variant) – Sets *variant*.

_font_desc

Reference to the C-implementation of font description.

Type manimpango.fonts._font_desc._FontDescription

Methods

from_string	Parse a string and form *FontDescription* from it.

property family: str

The family name of the font.

The family name represents a family of related font styles, and will resolve to a particular family. It is also possible to use a comma separated list of family names for this field.

classmethod from_string(*string*)

Parse a string and form *FontDescription* from it.

See https://docs.gtk.org/Pango/type_func.FontDescription.from_string.html#description for the syntax of the string.

Parameters **string** (*str*) – The string to be parsed.

Returns The *FontDescription* that is based on the string.

Return type *FontDescription*

property size: int

The size of the font of the text.

property style: *manimpango.fonts.enums.Style*

The style of the font of the text.

It should be one of *Style*. Most fonts will either have a 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.

property variant

The variant of the font of the text.

Should be one of *Variant*.

property weight: *manimpango.fonts.enums.Weight*

The weight of the font of the text.

The weight field specifies how bold or light the font should be. Should be one of *Weight*.

CHAPTER

FOUR

RELEASE PROCEDURE

This is the **maintainer** note on how to Release. All versioning is in accordance to Semantic Versioning 2.0.0. This means older version would have a backport of bugs fixes.

1. Check whether the test suite passes on the main branch.
2. Revert any changes which seems to be not working, and check for the milestone if PR are merged accordingly.
3. Check whether the Wheels Build, against the main branch works as expected.
4. Clone the repository locally.
5. Bump the version in manimpango/_version accordingly.
6. Make a commit with the changes done, as `Release v<version here>`
7. Create a tag, locally with

```
git tag -s v<version-number>
```

Note: Here, `-s` is used to sign the tag with gpg so that users can later verify it, and a tag shouldn't be created with signing because Github shows it unverified.

Important: The message should include the changelog of the release. There is a github actions which will creates a draft release with the changelog. You can edit them and copy it to the tag you create.

8. Push the tag to remote.
9. Go to Github, and draft a new release with the same tag pushed. You can copy the same changelog you copied when you created the tag.

Important: You should actually "draft a new release" instead of just publishing a previously present draft release created by the Github Action. This is important so that the wheels build workflow triggers.

10. Check whether the CI uploads the wheels and the `.tar.gz` file to PyPi.
11. Finally, test the `.tar.gz` which was uploaded to PyPi, and install it in a new virtual environment.

CHAPTER

FIVE

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

m

INDEX

Symbols

A

B

E

F

H

I

L

M

N

O

P

R

S

T

U

V

W