

# How to use rst2pdf

Author: Roberto Alsina <[ralcina@netmanagers.com.ar](mailto:ralcina@netmanagers.com.ar)>  
Version: 0.10.1  
Revision: 676

## Contents

1	Introduction	3
2	Command line options	3
3	Configuration File	4
4	Pipe usage	4
5	Headers and Footers	5
6	Footnotes	5
7	Images	5
7.1	Inline	6
7.2	Supported Image Types	6
7.3	Image Size	6
8	Styles	7
8.1	Included StyleSheets	8
8.2	StyleSheet Syntax	9
8.3	Font Alias	9
8.4	Style Definition	9
8.5	Font Embedding	10
8.5.1	The Easy Way	10
8.5.1.1	Fonty is a True Type font:	11
8.5.1.2	Fonty is a Type 1 font:	11
8.5.2	The Harder Way (True Type)	12
8.5.3	The Harder Way (Type1)	13
8.6	Page Size and Margins	13
8.7	Advanced: the tstyles section	14
8.8	Multiple Stylesheets	14
9	Syntax Highlighting	16
9.1	Inline	16
9.1.1	Examples	16
9.2	File inclusion	17
9.2.1	Include with Boundaries	17
10	Raw Directive	18
10.1	Page Transitions	18
11	Mathematics	19
12	Hyphenation	20

13	Page Layout	21
14	Smart Quotes	23

# 1 Introduction

This document explains how to use rst2pdf. Here is the very short version:

```
rst2pdf.py mydocument.txt -o mydocument.pdf
```

That will, as long as mydocument.txt is a valid Restructured Text (ReST) document, produce a file called mydocument.pdf which is a PDF version of your document.

Of course, that means you just used default styles and settings. If it looks good enough for you, then you may stop reading this document, because you are done with it. If you are reading this in a PDF, it was generated using those default settings.

However, if you want to customize the output, or are just curious to see what can be done, let's continue.

## 2 Command line options

<code>-h, --help</code>	show this help message and exit
<code>-o FILE, --output=FILE</code>	Write the PDF to FILE
<code>-s STYLESHEETS, --stylesheets=STYLESHEETS</code>	A comma-separated list of custom stylesheets. Default=""
<code>--stylesheet-path=FOLDERS</code>	A list of folders to search for stylesheets, separated using ":". Default=""
<code>-c, --compressed</code>	Create a compressed PDF. Default=False
<code>--print-stylesheet</code>	Print the default stylesheet and exit
<code>--font-folder=FOLDER</code>	Search this folder for fonts. (Deprecated)
<code>--font-path=FOLDERS</code>	A list of folders to search for fonts, separated using ":". Default=""
<code>--baseurl=URL</code>	The base URL for relative URLs. Default="None"
<code>-l LANG, --language=LANG</code>	Language to be used for hyphenation and docutils localizations. Default="en_US"
<code>--header=HEADER</code>	Page header if not specified in the document. Default="None"
<code>--footer=FOOTER</code>	Page footer if not specified in the document. Default="None"
<code>--smart-quotes=VALUE</code>	Try to convert ASCII quotes, ellipsis and dashes to the typographically correct equivalent. For details, read the man page or the manual. Default="0"
<code>--fit-literal-mode=MODE</code>	What to do when a literal is too wide. One of error, overflow, shrink, truncate. Default="shrink"
<code>-b LEVEL, --break-level=LEVEL</code>	Maximum section level that starts in a new page. Default: 0
<code>--inline-links</code>	shows target between parenthesis instead of active link
<code>--repeat-table-rows</code>	Repeats header row for each splitted table
<code>-q, --quiet</code>	Print less information.
<code>-v, --verbose</code>	Print debug information.
<code>--very-verbose</code>	Print even more debug information.

<code>--version</code>	Print version number and exit.
<code>--no-footnote-backlinks</code>	Disable footnote backlinks. Default=False
<code>--inline-footnotes</code>	Show footnotes inline. Default=True
<code>--default-dpi=NUMBER</code>	DPI for objects sized in pixels. Default=300
<code>--show-frame-boundary</code>	Show frame borders (only useful for debugging), default=False

For the options that take a folder list, like `--stylesheet-path`, the separator used will depend on your platform. On unix-like OSs, it's ":", on Windows it's ";". Don't blame me, blame DOS.

Some of these options' defaults can be changed by creating a [configuration file](#)

### 3 Configuration File

Since version 0.8, `rst2pdf` will read (if it is available) configuration files in `/etc/rst2pdf.conf` and `~/.rst2pdf/config`.

The user's file at `~/.rst2pdf/config` will have priority over the system's at `/etc/rst2pdf.conf`<sup>1</sup>

Here's an example file showing some of the currently available options:

```
# This is an example config file. Modify and place in ~/.rst2pdf/config

[general]
# A comma-separated list of custom stylesheets. Example:
# stylesheets="fruity.json,a4paper.json,verasans.json"
stylesheets=""

# Create a compressed PDF
# Use true/false (lower case) or 1/0
# Example: compressed=true
compressed=false

# A colon-separated list of folders to search for fonts. Example:
# font_path="/usr/share/fonts:/usr/share/texmf-dist/fonts/"
font_path=""

# Language to be used for hyphenation support
language="en_US"

# Default page header and footer
header=None
footer=None

# What to do if a literal block is too large. Can be
# shrink/truncate/overflow
fit_mode="shrink"

# What is the maximum level of heading that starts in a new page.
# 0 means no level starts in a new page.
break_level=0
```

Included with `rst2pdf` is an example file with every option in it.

## 4 Pipe usage

If no input nor output are provided, stdin and stdout will be used respectively

You may want to use rst2pdf in a linux pipe as such:

```
cat readme.txt | rst2pdf | gzip -c > readme.pdf.gz
```

or:

```
curl http://docutils.sourceforge.net/docs/user/rst/quickstart.txt | rst2pdf > quickstart.pdf
```

If no input argument is provided, stdin will be used:

```
cat readme.txt | rst2pdf -o readme.pdf
```

If outfile is set to dash '-', output goes to stdout:

```
rst2pdf -o - readme.txt > output.pdf
```

## 5 Headers and Footers

ReST supports headers and footers, using the header and footer directive:

```
.. header::  
  
    This will be at the top of every page.
```

Often, you may want to put a page number there, or a section name. The following magic tokens will be replaced (More will be added as rst2pdf evolves):

###Page###

Replaced by the current page number.

###Title###

Replaced by the document title

###Section###

Replaced by the current section title

###SectNum###

Replaced by the current section number. **Important:** You must use the sectnum directive for this to work.

Headers and footers are visible by default but they can be disabled by specific [Page Templates](#) for example, cover pages. You can also set headers and footers via **command line options** or the [configuration file](#).

## 6 Footnotes

Currently rst2pdf doesn't support real footnotes, and converts them to endnotes. There is a real complicated technical reason for this: I can't figure out a clean way to do it right.


You can get the same behaviour as with rst2html by specifying --inline-footnotes, and then the footnotes will appear where you put them (in other words, not footnotes, but "in-the-middle-of-text-notes" or just plain notes.)

## 7 Images

### 7.1 Inline

You can insert images in the middle of your text like this:

```
This |biohazard| means you have to run.  
  
.. |biohazard| image:: ../rst2pdf/tests/input/images/biohazard.png
```

This  means you have to run.

This only works correctly with reportlab 2.2 or later.

### 7.2 Supported Image Types

For raster images, rst2pdf supports anything PIL (The Python Imaging Library) supports. The exact list of supported formats varies according to your PIL version and system.

For vector image support, you need to install Dinu Gherman's svglib (<http://pypi.python.org/pypi/svglib/>) or Uniconvertor from <http://sk1project.org> version 1.1.3 or later.

It provides support for these formats:

- CorelDRAW ver.7-X3,X4 (CDR/CDT/CCX/CDRX/CMX)
- Adobe Illustrator up to 9 ver. (AI postscript based)
- Postscript (PS)
- Encapsulated Postscript (EPS)
- Computer Graphics Metafile (CGM)
- Windows Metafile (WMF)
- XFIG
- Scalable Vector Graphics (SVG)
- Skencil/Sketch/sk1 (SK and SK1)
- Acorn Draw (AFF)

Some features will not work when using these images. For example, gradients will not display, and text may cause problems.

To test suitability of your vector images for use with rst2pdf, try converting them to PDF using uniconvertor. The result should be exactly the way they will look when used in your documents.

If you can choose between raster and vectorial images, for non-photographic images, vector files are usually smaller and look better, specially when printed.

If you want to use PDF files as images, you need to install PythonMagick (<http://www.imagemagick.org>), which will be used to convert it to PNG and then inserted in your document. If the quality is not good enough, try something like `--default-dpi 1200`

This only works for one-page PDF files, and has several drawbacks, such as inability to copy text from the embedded image.

In the future, rst2pdf will support ReportLab's PageCatcher to properly embed PDFs. That is not implemented yet, though.

If there is any other image format you need supported, please report it as a feature request in rst2pdf's site.

## 7.3 Image Size

PDFs are meant to reflect paper. A PDF has a specific size in centimeters or inches.

Images usually are measured in pixels, which are meaningless in a PDF. To convert between pixels and inches or centimeters, we use a DPI (dots-per-inch) value.

For example, 300 pixels, with a 300DPI, are exactly one inch. 300 pixels at 100DPI are 3 inches.

For that reason, to achieve a nice layout of the page, it's usually a good idea to specify the size of your images in those units, or as a percentage of the available width and you can ignore all this DPI nonsense ;-)

The rst2pdf default is 300DPI, but you can change it using the `--default-dpi` option or the `default_dpi` setting in the config file.

Examples of images with specified sizes:

```
.. image:: home.png
   :width: 3in

.. image:: home.png
   :width: 80%

.. image:: home.png
   :width: 7cm
```

The valid units you can use are:

"em" "ex" "px" "in" "cm" "mm" "pt" "pc" "%" "".

- px: Pixels. If you specify the size using this unit, rst2pdf will convert it to inches using the default DPI explained above.
- No unit. If you just use a number, it will be considered as pixels. (**IMPORTANT:** this used to default to points. It was changed to be more compatible with rst2html)
- em: This is the same as your base style's font size. By default: 10 points.
- ex: rst2pdf will use the same broken definition as IE: em/2. In truth this should be the height of the lower-case x character in your base style.
- in: Inches (1 inch = 2.54 cm).
- cm: centimeters (1cm = 0.39 inches)
- mm: millimeters (10mm = 1cm)
- pt: 1/72 inch
- pc: 1/6 inch
- %: percentage of available width in the frame. Setting a percentage as a height does **not** work and probably never will.

If you don't specify a size at all, rst2pdf will do its best to figure out what it should do:

Since there is no specified size, rst2pdf will try to convert the image's pixel size to inches using the DPI information available in the image itself. You can set that value using most image editors. For example, using Gimp, it's in the Image -> Print Size menu.

So, if your image is 6000 pixels wide, and is set to 1200DPI, it will be 5 inches wide.

If your image doesn't have a DPI property set, and doesn't have it's desired size specified, rst2pdf will arbitrarily decide it should use 300DPI (or whatever you choose with the `--default-dpi` option).

As of 0.10.1, images taller than the page will not work (rst2pdf will fail to run), and images wider than the page will be cropped.

## 8 Styles

You can style paragraphs with a style using the class directive:

```
.. class:: special

This paragraph is special.

This one is not.
```

Or inline styles using custom interpreted roles:

```
.. role:: redtext

I like color :redtext:`red`.
```

For more information about this, please check the ReST docs.

The only special thing about using rst2pdf here is the syntax of the stylesheet.

You can make rst2pdf print the default stylesheet:

```
rst2pdf --print-stylesheet
```

If you want to add styles, just create a stylesheet, (or take the standard stylesheet and modify it) and pass it with the -s option:

```
rst2pdf mydoc.txt -s mystyles.txt
```

Those styles will always be searched in these places, in order:

- What you specify using `--stylesheet_path`
- The option `stylesheet_path` in the config file
- The current folder
- `~/rst2pdf/styles`
- The styles folder within rst2pdf's installation folder.

You can use multiple -s options, or pass more than one stylesheet separated with commas. They are processed in the order you give them so the *last* one has priority.

### 8.1 Included StyleSheets

To make some of the more common adjustments easier, rst2pdf includes a collection of stylesheets you can use:

#### Font styles

These stylesheets modify your font settings.

- `serif` uses the PDF serif font (Times) instead of the default Sans Serif (Arial)
- `freetype-sans` uses your system's default TrueType Sans Serif font
- `freetype-serif` uses your system's default TrueType Serif font
- `twelvepoint` makes the base font 12pt (default is 10pt)
- `tenpoint` makes the base font 10pt
- `eightpoint` makes the base font 8pt

#### Page layout styles

These stylesheets modify your page layout.

- `twocolumn` uses the twoColumn layout as the initial page layout.
- `double-sided` adds a gutter margin (margin at the "in side" of the pages)



### Page size styles

Stylesheets that change the paper size.

The usual standard paper sizes are supported:

- A0
- A1
- A2
- A3
- A4 (default)
- A5
- A6
- B0
- B1
- B2
- B3
- B4
- B5
- B6
- Letter
- Legal
- 11x17

The name of the stylesheet is lowercase.

### Code block styles

See [Syntax Highlighting](#)

So, if you want to have a two-column, legal size, serif document with code in murphy style:

```
rst2pdf mydoc.txt -s twocolumn,serif,murphy,legal
```

## 8.2 StyleSheet Syntax

It's a JSON file with several elements in it.

## 8.3 Font Alias

This is the fontsAlias element. By default, it uses some of the standard PDF fonts:

```
"fontsAlias" : {  
  "stdFont": "Helvetica",  
  "stdBold": "Helvetica-Bold",  
  "stdItalic": "Helvetica-Oblique",  
  "stdBoldItalic": "Helvetica-BoldOblique",  
  "stdMono": "Courier"  
},
```

This defines the fonts used in the styles. You can use, for example, Helvetica directly in a style, but if later you want to use another font all through your document, you will have to change it in each style. So, I suggest you use aliases.

The standard PDF fonts are these:

Times_Roman	Times-Bold	Times-Italic	Times-Bold-Italic	Helvetica	Helvetica_Bold
Helvetica-Oblique	Helvetica-Bold-Oblique	Courier	Courier-Bold	Courier-Oblique	
Courier-Bold-Oblique	Symbol	Zapf-Dingbats			

## 8.4 Style Definition

Then you have a 'styles' which is a list of [ stylename, styleproperties ]. For example:

```
[ "normal" , {  
    "parent": "base"  
} ],
```

This means that the style called "normal" inherits style "base". So, each property not defined in the normal style will be taken from the base style.

I suggest you do not remove any style from the default stylesheet. Add or modify at will, though.

If your document requires a style that is not defined in your stylesheet, it will print a warning and use bodytext instead.

Also, the order of the styles is important: if styleA is the parent of styleB, styleA should be earlier in the stylesheet.

These are all the possible attributes for a style and their default values. Some of them, like alignment, apply only when used to paragraphs, and not on inline styles:

```
"fontName": "Times-Roman",  
"fontSize": 10,  
"leading": 12,  
"leftIndent": 0,  
"rightIndent": 0,  
"firstLineIndent": 0,  
"alignment": TA_LEFT,  
"spaceBefore": 0,  
"spaceAfter": 0,  
"bulletFontName": "Times-Roman",  
"bulletFontSize": 10,  
"bulletIndent": 0,  
"textColor": black,  
"backColor": None,  
"wordWrap": None,  
"borderWidth": 0,  
"borderPadding": 0,  
"borderColor": None,  
"borderRadius": None,  
"allowWidows": 1,  
"allowOrphans": 0
```

The following are the only attributes that work on styles when used for interpreted roles (inline styles):

- fontName
- fontSize
- textColor

Notice that backColor is **not** in that list.

## 8.5 Font Embedding

There are thousands of excellent free True Type and Type 1 fonts available on the web, and you can use many of them in your documents by declaring them in your stylesheet.

### 8.5.1 The Easy Way

Just use the font name in your style. For example, you can define this:

```
[ "normal" , {  
  "fontName" : "fonty"  
}]
```

And then it *may* work.

What would need to happen for this to work?

#### 8.5.1.1 Fonty is a True Type font:

1. You need to have it installed in your system, and have the *fc-match* utility available (it's part of [fontconfig](#)). You can test if it is so by running this command:

```
$ fc-match fonty  
fonty.ttf: "Fonty" "Normal"
```

If you are in Windows, I need your help ;- ) or you can use [The Harder Way \(True Type\)](#)

2. The folder where *fonty.ttf* is located needs to be in your font path. You can set it using the `--font-path` option. For example:

```
rst2pdf mydoc.txt -s mystyle.style --font-path /usr/share/fonts
```

You don't need to put the exact folder, just something that is above it. In my own case, fonty is in `/usr/share/fonts/TTF`

Whenever a font is embedded, you can refer to it in a style by its name, and to its variants by the aliases Name-Oblique, Name-Bold, Name-BoldOblique.

#### 8.5.1.2 Fonty is a Type 1 font:

You need it installed, and the folders where its font metric (.afm) and binary (.pfb) files are located need to be in your font path.

For example, the "URW Palladio L" font that came with my installation of TeX consists of the following files:

```
/usr/share/texmf-dist/fonts/type1/urw/palatino/uplb8a.pfb  
/usr/share/texmf-dist/fonts/type1/urw/palatino/uplbi8a.pfb  
/usr/share/texmf-dist/fonts/type1/urw/palatino/uplr8a.pfb  
/usr/share/texmf-dist/fonts/type1/urw/palatino/uplri8a.pfb  
/usr/share/texmf-dist/fonts/afm/urw/palatino/uplb8a.afm  
/usr/share/texmf-dist/fonts/afm/urw/palatino/uplbi8a.afm  
/usr/share/texmf-dist/fonts/afm/urw/palatino/uplr8a.afm  
/usr/share/texmf-dist/fonts/afm/urw/palatino/uplri8a.afm
```

So, I can use it if I put `/usr/share/texmf-dist/fonts` in my font path:

```
rst2pdf mydoc.txt -s mystyle.style --font-path /usr/share/texmf-dist/fonts
```

And putting this in my stylesheet, for example:

```
[ "title", { "fontName" : "URWPalladioL-Bold" } ]
```

There are some standard aliases defined so you can use other names:

```
'ITC Bookman'           : 'URW Bookman L',  
'ITC Avant Garde Gothic' : 'URW Gothic L',  
'Palatino'              : 'URW Palladio L',  
'New Century Schoolbook' : 'Century Schoolbook L',  
'ITC Zapf Chancery'      : 'URW Chancery L'
```

So, for example, you can use `Palatino` or `New Century SchoolBook-Oblique` And it will mean `URWPalladioL` or `CenturySchL-Ital`, respectively.

Whenever a font is embedded, you can refer to it in a style by its name, and to its variants by the aliases `Name-Oblique`, `Name-Bold`, `Name-BoldOblique`.

### 8.5.2 The Harder Way (True Type)

The stylesheet has an element is `"embeddedFonts"` that handles embedding True Type fonts in your PDF.

Usually, it's empty, because with the default styles you are not using any font beyond the standard PDF fonts:

```
"embeddedFonts" : [ ],
```

You can put there the name of the font, and `rst2pdf` will try to embed it as described above. Example:

```
"embeddedFonts" : [ "Tuffy" ],
```

Or you can be explicit and tell `rst2pdf` the files that contain each variant of the font.

Suppose you want to use the nice public domain [Tuffy font](#) (included in `rst2pdf`'s source distribution).

You need to give the filenames of all variants:

```
"embeddedFonts" : [ [ "Tuffy.ttf", "Tuffy_Bold.ttf", "Tuffy_Italic.ttf", "Tuffy_Bold_Italic.ttf" ] ],
```

This will provide your styles with fonts called `"Tuffy"` `"Tuffy_Bold"` and so on. They will be available with the names based on the filenames (`Tuffy_Bold`) and also by standard aliases similar to those of the standard PDF fonts (`Tuffy-Bold/Tuffy-Oblique/Tuffy-BoldOblique`).

Now, if you use *italics* in a paragraph whose style uses the Tuffy font, it will use `Tuffy_Italic`. That's why it's better if you use fonts that provide the four variants, and that is the order in which you should put them. If your font lacks a variant, use the `"normal"` variant instead. For example, if you only had `Tuffy.ttf`:

```
"embeddedFonts" : [ [ "Tuffy.ttf", "Tuffy.ttf", "Tuffy.ttf", "Tuffy.ttf" ] ],
```

However, that means that italics and bold in styles using Tuffy will not work correctly (they will display as regular text).

If you want to use this as the base font for your document, you should change the `fontsAlias` section accordingly. For example:

```
"fontsAlias" : {  
  "stdFont": "Tuffy",  
  "stdBold": "Tuffy_Bold",
```

```
"stdItalic": "Tuffy_Italic",  
"stdBoldItalic": "Tuffy_Bold_Italic",  
"stdMono": "Courier"  
},
```

If, on the other hand, you only want a specific style to use the Tuffy font, don't change the `fontAlias`, and set the `fontName` properties for that style. For example:

```
[ "heading1" , {  
  "parent": "normal",  
  "fontName": "Tuffy_Bold",  
  "bulletFontName": "Tuffy_Bold",  
  "fontSize": 18,  
  "bulletFontSize": 18,  
  "leading": 22,  
  "keepWithNext": true,  
  "spaceAfter": 6  
}],
```

By default, `rst2pdf` will search for the fonts in its fonts folder and in the current folder. You can make it search another folder by passing the `--font-folder` option, or you can use absolute paths in your stylesheet.

### 8.5.3 The Harder Way (Type1)

To be written (and implemented and tested)

## 8.6 Page Size and Margins

In your stylesheet, the `pageSetup` element controls your page layout.

Here's the default stylesheet's:

```
"pageSetup" : {  
  "size": "A4",  
  "width": null,  
  "height": null,  
  "margin-top": "2cm",  
  "margin-bottom": "2cm",  
  "margin-left": "2cm",  
  "margin-right": "2cm",  
  "spacing-header": "5mm",  
  "spacing-footer": "5mm",  
  "margin-gutter": "0cm"  
},
```

Size is one of the standard paper sizes, like A4 or LETTER.

Here's a list: A0, A1, A2, A3, A4, A5, A6, B0, B1, B2, B3, B4, B5, B6, LETTER, LEGAL, ELEVENSEVENTEEN.

If you want a non-standard size, set size to null and use width and height.

When specifying width, height or margins, you need to use units, like inch (inches) or cm (centimeters).

When both width/height and size are specified, size will be used, and width/height ignored.

All margins should be self-explanatory, except for margin-gutter. That's the margin in the center of a two-page spread.

This value is added to the left margin of odd pages and the right margin of even pages, adding (or removing, if it's negative) space "in the middle" of opposing pages.

If you intend to bound a printed copy, you may need extra space there. OTOH, if you will display it on-screen on a two-page format (common in many PDF readers, nice for ebooks), a negative value may be pleasant.

## 8.7 *Advanced: the tstyles section*

This is new in 0.12, and usually you don't need to touch it. If you feel adventurous, here's what's in it.

There are two kinds of elements:

### *Lengths*

They are used to layout part of the document that are implemented using tables. for example, "bullet\_lwidth" is the space reserved for bullets in the left side of lists, and "endnote\_lwidth" is the space reserved for the left part of endnotes/footnotes.

### *Table commands*

For a full reference of these, please check the Reportlab User Guide specifically the TableStyle Commands section (section 7.4 in the manual for version 2.3)

Here, however, is a list of the possible commands:

```
BOX (or OUTLINE)
FONT
FONTNAME (or FACE)
FONTSIZE (or SIZE)
GRID
INNERGRID
LEADING
LINEBELOW
LINEABOVE
LINEBEFORE
LINEAFTER
TEXTCOLOR
ALIGNMENT (or ALIGN)
LEFTPADDING
RIGHTPADDING
BOTTOMPADDING
TOPPADDING
BACKGROUND
ROWBACKGROUNDS
COLBACKGROUNDS
VALIGN
```

Each takes as argument a couple of coordinates, where (0,0) is top-left, and (-1,-1) is bottom-right, and 0 or more extra arguments.

For example, INNERGRID takes a linewidth and a color:

```
[ "INNERGRID", [ 0, 0 ], [ -1, -1 ], 0.25, "black" ],
```

That would mean "draw all lines inside the table with .25pt black"

## 8.8 Multiple Stylesheets

When you use a custom stylesheet, you don't need to define *everything* in it. Whatever you don't define will be taken from the default stylesheet. For example, if you only want to change page size, default font and font size, this would be enough:

```
{
  "pageSetup" : {
    "size": "A5",
  },
  "fontsAlias" : {
    "stdFont": "Times-Roman",
  },
  "styles" : [
    [ "normal" , {
      "fontSize": 14
    } ]
  ]
}
```

## 9 Syntax Highlighting

### 9.1 Inline

Rst2pdf adds a non-standard directive, called code-block, which produces syntax highlighted for many languages using [Pygments](#).

For example, if you want to include a python fragment:

```
.. code-block:: python

    def myFun(x,y):
        print x+y
```

```
def myFun(x,y):
    print x+y
```

Notice that you need to declare the language of the fragment. Here's a list of the currently [supported](#).

Rst2pdf includes several stylesheets for highlighting code:

- autumn
- borland
- bw
- colorful
- emacs
- friendly
- fruity
- manni
- murphy
- native
- pastie
- perldoc
- trac
- vs

You can use any of them instead of the default by adding, for example, a `-s murphy` to the command line.

If you already are using a custom stylesheet, use both:

```
rst2pdf mydoc.rst -o mydoc.pdf -s mystyle.json,murphy
```

The default is the same as "emacs".

There is an online demo of pygments showing these styles:

<http://pygments.org/demo/1817/>

#### 9.1.1 Examples

As rst2pdf is in python let's see some examples and variations around python

Python in console

```
>>> my_string="python is great"
>>> my_string.find('great')
```



```
10
>>> my_string.startswith('py')
True
```

Python traceback

```
Traceback (most recent call last):
  File "error.py", line 9, in ?
    main()
  File "error.py", line 6, in main
    print call_error()
  File "error.py", line 2, in call_error
    r = 1/0
ZeroDivisionError: integer division or modulo by zero
Exit 1
```

## 9.2 File inclusion

Also, you can use the code-block directive with an external file, using the `:include:` option:

```
.. code-block:: python
   :include: setup.py
```

This will give a warning if `setup.py` doesn't exist or can't be opened.

### 9.2.1 Include with Boundaries

you can add selectors to limit the inclusion to a portion of the file. the options are:

- `start-at:` string  
will include file beginning at the first occurrence of string, string **included**
- `start-after:` string  
will include file beginning at the first occurrence of string, string **excluded**
- `end-before:` string  
will include file up to the first occurrence of string, string **excluded**
- `end-at:` string  
will include file up to the first occurrence of string, string **included**

Let's display a class from `rst2pdf`:

```
.. code-block:: python
   :include: ../rst2pdf/flowables.py
   :start-at: class Separation(Flowable):
   :end-before: class Reference(Flowable):
```

this command gives

```
class Separation(Flowable):
    """A simple <hr>-like flowable"""

    def wrap(self, w, h):
        self.w = w
```

```

    return w, 1*cm

def draw(self):
    self.canv.line(0, 0.5*cm, self.w, 0.5*cm)

```

## 10 Raw Directive

Rst2pdf has a very limited mechanism to pass commands to reportlab, the PDF generation library. You can use the raw directive to insert pagebreaks and spacers (other reportlab flowables may be added if there's interest), and set page transitions.

The syntax is shell-like, here's an example:

```

One page

.. raw:: pdf

    PageBreak

Another page. Now some space:

.. raw:: pdf

    Spacer 0,200
    Spacer 0 200

And another paragraph.

```

The unit used by the spacer by default is points, and using a space or a comma is the same thing in all cases.

### 10.1 Page Transitions

Page transitions are effects used when you change pages in *Presentation* or *Full Screen* mode (depends on the viewer). You can use it when creating a presentation using PDF files.

The syntax is this:

```

.. raw:: pdf

    Transition effect duration [optional arguments]

```

The optional arguments are:

*direction:*

Can be 0,90,180 or 270 (top,right,bottom,left)

*dimension:*

Can be H or V

*motion:*

Can be I or O (Inside or Outside)

The effects with their arguments are:

- Split duration direction motion
- Blinds duration dimension
- Box duration motion

- Wipe duration direction
- Dissolve duration
- Glitter duration direction

For example:

```
.. raw:: pdf

    Transition Glitter 3 90
```

Uses the Glitter effect, for 3 seconds, at direction 90 degree (from the right?)

Keep in mind that Transition sets the transition *from this page to the next* so the natural thing is to use it before a PageBreak:

```
.. raw:: pdf

    Transition Dissolve 1
    PageBreak
```

## 11 Mathematics

If you have [Matplotlib](#) installed, rst2pdf supports a math role and a math directive. You can use them to insert formulae and mathematical notation in your documents using a subset of LaTeX syntax, but doesn't require you have LaTeX installed.

For example, here's how you use the math directive:

```
.. math::

    \frac{2 \pm \sqrt{7}}{3}
```

And here's the result:

$$\frac{2 \pm \sqrt{7}}{3}$$

If you want to insert mathematical notation in your text like this:  $\pi$  that is the job of the math role:

```
This is :math:`\pi`
```

Produces: This is  $\pi$

Currently, the math role is slightly buggy, and in some cases will produce misaligned and generally broken output. Also, while the math directive embeds fonts and draws your formula as text, the math role embeds an image. That means:

- You can't copy the text of inline math
- Inline math will look worse when printed, or make your file larger.

So, use it only in emergencies ;-)

You can also use an inline substitution of the math directive for things you use often, which is the same as using the math role:

```
This is the square of x: |xsq|

.. |xsq| math:: x^2
```

This is the square of x:  $x^2$

You don't need to worry about fonts, the correct math fonts will be used and embedded in your PDF automatically (they are included with matplotlib).

For an introduction to LaTeX syntax, see the "Typesetting Mathematical Formulae" chapter in "The Not So Short Introduction to LaTeX 2e":

<http://www.tex.ac.uk/tex-archive/info/lshort/english/lshort.pdf>

Basically, the inline form `$a^2$` is similar to the math role, and the display form is similar to the math directive.

Rst2pdf doesn't support numbering equations yet.

## 12 Hyphenation

If you want good looking documents, you want to enable hyphenation.

To do it, you need to install Wordaxe <sup>2</sup>.

If after installing it you get the letter "s" or a black square instead of a hyphen, that means you need to replace the `rl_codecs.py` file from `reportlab` with the one from `wordaxe`.

For more information, see [this issue](#) in `rst2pdf`'s bug tracker.

Also, you may need to set hyphenation to true in one or more styles, and the language for hyphenation via the command line or paragraph styles.

For english, this should be enough:

```
[ "bodytext" , {  
    "alignment": "TA_JUSTIFY",  
    "hyphenation": true  
}],
```

If you are not an english speaker, you need to change the language.

You can use the `-l` or `--language` option. The currently available dictionaries for wordaxe are:

- `de_DE`
- `da`
- `en_GB`
- `en_US`
- `ru`

However, since Wordaxe version 0.2.6, it can use the PyHyphen library if it's available. PyHyphen can use any OpenOffice dictionary, and can even download them automatically. <sup>3</sup>

For example, this will enable german hyphenation globally:

```
rst2pdf -l de_DE mydocument.txt
```

If you are creating a multilingual document, you can declare styles with specific languages. For example, you could inherit `bodytext` for spanish:

```
[ "bodytext_es" , {  
    "parent": "bodytext",  
    "alignment": "TA_JUSTIFY",  
    "hyphenation": true,  
    "language": "es_ES"  
}],
```

And all paragraphs declared of `bodytext_de` style would have spanish hyphenation:

```
.. class:: bodytext_es
```

Debo a la conjunción de un espejo y de una enciclopedia el descubrimiento de Uqbar. El espejo inquietaba el fondo de un corredor en una quinta de la calle Gaona, en Ramos Mejía; la enciclopedia falazmente se llama ***The Anglo-American Cyclopaedia*** (New York, 1917) y es una reimpresión literal, pero también morosa, de la ***Encyclopaedia Britannica*** de 1902.

Here is the result (made thinner to force hyphenation):

Debo a la conjunción de un espejo y de una enciclopedia el descubrimiento de Uqbar. El espejo inquietaba el fondo de un corredor en una quinta de la calle Gaona, en Ramos Mejía; la enciclopedia falazmente se llama *The Anglo-American Cyclopaedia* (New York, 1917) y es una reimpresión literal, pero también morosa, de la *Encyclopaedia Britannica* de 1902.

BTW: That's the beginning of "Tlön, Uqbar, Orbis Tertius", read it, it's cool.

If you explicitly configure a language in a paragraph style and also pass a language in the command line, the style has priority, so remember:

### ***Important***

If you configure the bodytext style to have a language, your document is supposed to be in that language, regardless of what the command line says.

If this is too confusing, let me know, I will try to figure out a simpler way.

## 13 *Page Layout*

By default, your document will have a single column of text covering the space between the margins. You can change that, though, in fact you can do so even in the middle of your document!

To do it, you need to define *Page Templates* in your stylesheet. The default stylesheet already has 3 of them:

```
"pageTemplates" : {
  "coverPage": {
    "frames": [
      [ "0cm", "0cm", "100%", "100%" ]
    ],
    "showHeader" : false,
    "showFooter" : false
  },
  "oneColumn": {
    "frames": [
      [ "0cm", "0cm", "100%", "100%" ]
    ]
  },
}
```

```

"twoColumn": {
  "frames": [
    [ "0cm", "0cm", "49%", "100%" ],
    [ "51%", "0cm", "49%", "100%" ]
  ]
}

```

A page template has a name (oneColumn, twoColumn), some options, and a list of frames. A frame is a list containing this:

```
[ left position, top position, width, height ]
```

For example, this defines a frame "at the very left", "at the very top", "a bit less than half a page wide" and "as tall as possible":

```
[ "0cm", "0cm", "49%", "100%" ]
```

And this means "the bottom third of the page":

```
[ "0cm", "66.66%", "100%", "33.34%" ]
```

You can use all the usual units, cm, mm, inch, and % which means "percentage of the page (excluding margins and headers or footers)". Using % is probably the smartest for columns and gives you a fluid layout, while the other units are better for more "fixed" elements.

Since we can have more than one template, there is a way to specify which one we want to use, and a way to change from one to another.

To specify the first template, do it in your stylesheet, in pageSetup (oneColumn is the default):

```

"pageSetup" : {
  "firstTemplate": "oneColumn"
}

```

Then, to change to another template, in your document use this syntax (will change soon, though):

```

.. raw:: pdf

    PageBreak twoColumn

```

That will trigger a page break, and the new page will use the twoColumn template.

You can see an example of this in the *Montecristo* folder in the source package.

The supported page template options and their defaults are:

- showHeader : True
- defaultHeader : None
  - Has the same effect as the header directive in the document.
- showFooter : True
- defaultFooter : None
  - Has the same effect as the footer directive in the document.

- background: None

The background should be an image, which will be stretched to match your page, so use with caution.

## 14 *Smart Quotes*

Quoted from the [smartypants](#) documentation:

This feature can perform the following transformations:

Straight quotes ( " and ' ) into "curly" quote HTML entities  
Backticks-style quotes (``like this'') into "curly" quote HTML entities  
Dashes (-- and ---) into en- and em-dash entities  
Three consecutive dots (... or . . .) into an ellipsis entity  
This means you can write, edit, and save your posts using plain old ASCII straight quotes, plain dashes, and plain dots, but your published posts (and final PDF output) will appear with smart quotes, em-dashes, and proper ellipses.

You can enable this by passing the `--smart-quotes` option in the command line. By default, it's disabled. Here are the different values you can use (again, from the smartypants docs):

- 0  
Suppress all transformations. (Do nothing.)
- 1  
Performs default SmartyPants transformations: quotes (including ``backticks" -style), em-dashes, and ellipses. "--" (dash dash) is used to signify an em-dash; there is no support for en-dashes.
- 2  
Same as `smarty_pants="1"`, except that it uses the old-school typewriter shorthand for dashes: "--" (dash dash) for en-dashes, "---" (dash dash dash) for em-dashes.
- 3  
Same as `smarty_pants="2"`, but inverts the shorthand for dashes: "--" (dash dash) for em-dashes, and "---" (dash dash dash) for en-dashes.

Currently, even if you enable it, this transformation will only take place in regular paragraphs, titles, headers, footers and block quotes.

In addition, there is currently an incompatibility between this and wordaxe's pyhyphen plugin, so it's not really usable in some cases.

- 
- 1 The `/etc/rst2pdf.conf` location makes sense for Linux and linux-like systems. if you are using rst2pdf in other systems, please contact me and tell me where the system-wide config file should be.
  - 2 You can get Wordaxe from <http://deco-cow.sf.net>. Version 0.3.2 or later is recommended.
  - 3 For more information, please check the PyHyphen website at <http://pyhyphen.googlecode.com>