How to use rst2pdf

Contents

Getting started

Learn what the rst2pdf tool is, how to install it, and how to get started with using it.

Introduction

Learn what is rst2pdf and what are the initial conditions.

The rst2pdf tool creates PDF documents from your ReStructuredText markup. This Open Source, free-to-use tool will help you create documents, slide decks and other PDF output very quickly and easily.

Important: In this page you can learn how to install and use this tool only on Ubuntu.

Important: To install rst2pdf, you need pip. See the What is pip section to find out what it is in general. In the Installing pip section, you will find the installation procedure. If you already have pip, you can go straight to the Installing rst2pdf section.

What is pip

Learn what the pip is and what it is used for.

Pip is a recursive acronym that can stand for either Pip Installs Packages or Pip Installs Python. Alternatively, pip stands for Preferred Installer Program. It is a powerful tool that helps you properly manage distribution packages in Python.

If you don't have pip, below you will find instructions on how to install it. If you already have pip, go straight to the rst2pdf tool installation.

Installing pip

Learn how to install pip.

1. Press **ctrl+alt+t** to run the command line.
2. At the command line, type `sudo apt update` and press **enter**.
3. When you get a message that asks for your computer password, type it in and press **enter**.
4. Type `apt install python3-pip` and press **enter**.
5. Continue following the instructions that appear on the command line.
6. If the instructions stop appearing, it means that the program has been successfully installed.

After you have installed pip, you can move on to the rst2pdf tool installation.

Installing rst2pdf

Learn how to install the rst2pdf tool.

1. If you don't have the command line opened, press **ctrl+alt+t** to do so.
2. At the command line, type `pip3 install rst2pdf` and press **enter**.

After that you should see information about a successful installation.

If you want to always run the program easily and quickly by just typing rst2pdf into the command line, follow the instruction Adding pip programs to $PATH.

Adding pip programs to $PATH

Learn how to add pip programs to $PATH so you can always run the rst2pdf tool easily and quickly.

1. Open **Files**.
2. Go to **Home**.
3. Click the icon in the upper right corner and select **Show Hidden Files**.
   After that, folders and files with a comma at the beginning should appear in the **Home section**.
4. Open the *.bashrc* file.
5. At the end of the file type: `export PATH=$PATH:$HOME/.local/bin` and click **Save**.

How to start using rst2pdf

Learn how to start using rst2pdf. You can use a sample file.

1. Below is an example text you can use to get started with rst2pdf. Copy it and paste into the **Text Editor**. Then save it as a *sample.rst* file.

```
PDF Documents From Your Text Editor
###################################

This tool uses a plain text format for content, then applies styles to
 make the nice document you see here.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aliquam maximus
 tortor sem, vel pellentesque leo fringilla et. Aliquam
imperdiet nisi eget dui finibus sagittis. Nunc malesuada libero vel
 dignissim pharetra. Cras egestas vehicula quam, et accumsan
arcu lacinia auctor. Integer imperdiet sagittis justo, vel varius nulla
 dapibus finibus. Cras rhoncus mattis pellentesque. Quisque
vel sapien sed tellus convallis accumsan. Praesent volutpat sapien at
 lacinia scelerisque. Phasellus neque libero, consectetur in
neque id, egestas elementum nisl.

Mauris eu dolor non massa auctor suscipit. Donec sit amet aliquet eros, id
 sodales leo. Duis erat ipsum, laoreet eget nulla at,
euismod ullamcorper mi. Curabitur vel orci a libero ullamcorper finibus.
 Sed vel lectus sapien. Praesent mollis et dui at laoreet.
Donec eleifend, nunc nec bibendum luctus, massa lorem vestibulum justo, a
 convallis nunc turpis ut urna. Proin venenatis erat et
ante convallis efficitur. Lorem ipsum dolor sit amet, consectetur
 adipiscing elit. In neque turpis, sollicitudin maximus egestas
sed, finibus a odio. Nam eu eros id enim vehicula hendrerit at vel orci.
 Curabitur volutpat tempor purus ut auctor. Mauris
vulputate sollicitudin porttitor. Nunc consectetur lectus nibh, et commodo
 purus porttitor ut. Nulla facilisi.

Further Sections, Headings, etc
===============================

Sed id interdum quam. Donec suscipit, justo ac blandit vestibulum, massa
 est semper dolor, vel sodales nulla metus eu purus. In
risus tortor, interdum pellentesque dolor a, rhoncus volutpat orci. Ut
 commodo purus orci, non aliquet massa maximus in. Morbi
semper dui ipsum, a dictum nisi pretium ut. Vestibulum bibendum lacinia
 viverra. In hac habitasse platea dictumst. Nunc vestibulum
maximus mollis. Cras suscipit dictum condimentum. Nam vulputate, enim at
 gravida placerat, tellus nibh accumsan ligula, auctor
```

```
ornare purus leo ac orci. Quisque ultricies dictum purus. In facilisis
 feugiat suscipit.
```

2. Press **ctrl+alt+t** to run the command line.
3. Type `rst2pdf sample.rst`.
4. Check for a new file called *sample.pdf*. It should be in the same folder as the *sample.rst* file.

And you are already on your rst2pdf journey!

Command line options

Learn what you can do using the command line, such as how to add a configuration file, how to add a stylesheet, or change the font.

If you need help with the commands, type `rst2pdf -h` or `rst2pdf --help` and press **enter**. In the command line, you should see an instruction with command line options, similar to the one below.

Here are the command line options to type after the command `rst2pdf sample.rst` (where *sample.rst* is the name of the rst file from which you want to generate the PDF).

Note: Only after you have typed all the commands you need, press **enter**.

Table 1: Command line options

Command	Description
`--config=FILE`	Config file to use. Default = *~/.rst2pdf/config*
`-o FILE, --output=FILE`	Write the PDF to FILE
`-s STYLESHEETS, --stylesheets=STYLESHEETS`	A comma-separated list of custom stylesheets. Default = ""
`-c, --compressed`	Create a compressed PDF. Default = False
`--print-stylesheet`	Print the default stylesheet and exit
`--baseurl=URL`	The base URL for relative URLs
`--header=HEADER`	Page header if not specified in the document
`--footer=FOOTER`	Page footer if not specified in the document
`--section-header-depth=N`	Sections up to this dept will be used in the header and footer's replacement of ###Section###. Default = 2

Command	Description
`--smart-quotes=VALUE`	Try to convert ASCII quotes, ellipsis and dashes to the typographically correct equivalent. Default = 0 The possible values are: 1. Suppress all transformations. (Do nothing.) 2. 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. 3. Same as --smart-quotes=1, except that it uses the old-school typewriter shorthand for dashes: "--" (dash dash) for en-dashes, "---" (dash dash dash) for em-dashes. 4. Same as --smart-quotes=2, but inverts the shorthand for dashes: "--" (dash dash) for em-dashes, and "---" (dash dash dash) for en-dashes.
`--fit-literal-mode=MODE`	What to do when a literal is too wide. One of error, overflow, shrink, truncate. Default = "shrink"
`--fit-background-mode=MODE`	How to fit the background image to the page. One of scale, scale_width or center. Default = "center"
`--inline-links`	Shows target between parenthesis instead of active link
`--repeat-table-rows`	Repeats header row for each splitted table
`--raw-html`	Support embedding raw HTML. Default = False
`-q, --quiet`	Print less information
`-v, --verbose`	Print debug information
`--very-verbose`	Print even more debug information
`--version`	Print version number and exit
`--default-dpi=NUMBER`	DPI for objects sized in pixels. Default = 300
`--show-frame-boundary`	Show frame borders (only useful for debugging). Default = False
`--disable-splittables`	Don't use splittable flowables in some elements. Only try this if you can't process a document any other way
`-b LEVEL, --break-level=LEVEL`	Maximum section level that starts in a new page. Default = 0
`--first-page-on-right`	When using double sided pages, the first page will start on the right hand side. (Book Style)
`--blank-first-page`	Add a blank page at the beginning of the document
`--break-side=VALUE`	How section breaks work. Can be "even", and sections start in an even page,"odd", and sections start in odd pages, or "any" and sections start in the next page,be it even or odd. See also the -b option
`--date-invariant`	Don't store the current date in the PDF. Useful mainly for the test suite, where we don't want the PDFs to change

Command	Description
`--custom-cover=FILE`	Template file used for the cover page. Default = *cover.tmpl*
`--use-numbered-links`	When using numbered sections, adds the numbers to all links referring to the section headers. Default = False
`--strip-elements-with-class=CLASS`	Remove elements with this CLASS from the output. Can be used multiple times

Configuration file

Learn how to create a configuration file that will help you style your documents easier and faster.

Since version 0.8, rst2pdf can 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*.

What is a configuration file

Learn why you should create a configuration file.

The configuration file has similar functions to the commands listed in the Command line options section. But to make configuration faster and more convenient, you can create a special file where you put all the configuration options you are interested in. The main advantage of such a file is that you can reuse it many times. You can think about it as a template where you have your chosen style, font, etc.

Example file

Here you can find 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
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=""

# A colon-separated list of folders to search for stylesheets. Example:
# stylesheet_path="~/styles:/usr/share/styles"
stylesheet_path=""

# Language to be used for hyphenation support

language="en_US"
```

```
# Default page header and footer
header=null
footer=null

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

fit_mode="shrink"

# How to adjust the background image to the page.
# Can be: "scale" and "center"

fit_background_mode="center"

# 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

# How section breaks work. Can be "even", and sections start in an
# even page, "odd", and sections start in odd pages, or "any" and
# sections start in the next page, be it even or odd.

break_side="any"

# Add a blank page at the beginning of the document

blank_first_page=false

# Treat the first page as even (default false, treat it as odd)

first_page_even=false

# Smart quotes.
# 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 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 2, but inverts the shorthand for dashes: "--" (dash dash) for
# em-dashes, and "---" (dash dash dash) for en-dashes.

smartquotes=0

# Footnote backlinks enabled or not (default: enabled)

footnote_backlinks=true

# Show footnotes inline instead of at the end of the document

inline_footnotes=false

# Cover page template.
# It will be searched in the document's folder, in ~/.rst2pdf/templates and
# in the templates subfolder of the package folder

# custom_cover = cover.tmpl

# Use floating images.
```

```
# Makes the behaviour of images with the :align: attribute more like
 rst2html's

floating_images = false

# Support the ..raw:: html directive
raw_html = false
```

Headers and footers

Learn how to create and configure headers and footers.

rST 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 will be replaced:

`###Page###`	Replaced by the current page number
`###Title###`	Replaced by the document title
`###Section###`	Replaced by the current section title

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.

Images

Learn useful things about inserting images.

You can find basic information about images in rST at thomas-cokelaer.info

Here you learn additional things important when generating a PDF.

Inline

You can insert images in the middle of your text.

```
This |biohazard| means you have to run.
.. |biohazard| image:: assets/biohazard.png
```

Image size

Learn how to adjust the size of the image.

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 centimeters or inches, or as a percentage of the available width and you can ignore DPI.

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`, `px`, `in`, `cm`, `mm`, `pt`, `pc`, `%`, "".

- `em`: this is the same as your base style's font size. By default: 10 points
- `px`: pixels. If you specify the size using this unit, rst2pdf converts it to inches using the default DPI explained above
- `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
- No unit: if you just use a number, it is considered as pixels

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

Since there is no specified size, rst2pdf tries 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.

So, if your image is 6000 pixels wide, and is set to 1200DPI, it is 5 inches wide. If your image doesn't have a DPI property set, and doesn't have it's desired size specified, rst2pdf arbitrarily decides it should use 300DPI (or whatever you choose with the `--default-dpi` option).

Styles

Here you can find information about the appearance of your document. You will learn how to use the default stylesheet and how to create and use your own.

The only special thing about using rt2pdf here is the syntax of the stylesheet.

You can make rst2pdf print the default stylesheet:

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

Here you have a table with the most basic attributes and their values that the default stylesheet has assigned to them:

Table 2: Default attributes values

Attribute	Default value	Description
`fontName`	Helvetica	Font type

Attribute	Default value	Description
fontSize	10	Font size
leading	12	Space between adjacent lines of type
leftIndent	0	Indent on the left
rightIndent	0	Indent on the right
firstLineIndent	0	Indent of the first line of an item
alignment	left	Placing an item on the page
spaceBefore	0	Space before item
spaceAfter	0	Space after item
bulletFontName	Helvetica	Bullet font type
bulletFontSize	10	Bullet font size
bulletText	\u2022	Bullet point type
bulletIndent	0	Indent of bullet points
textColor	black	Color of the text
backColor	None	Color of the background
wordWrap	None	Breaking a section of text into lines
borderWidth	0	Setting the width of an element's four borders
borderPadding	0	Generate space around an item's content, inside of any defined borders
borderColor	None	Setting the color of an item 's four borders
borderRadius	None	Defining the radius of the item's corners

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

Using class directive

Learn how to style individual paragraphs.

To do this, you must first define the class in the stylesheet. Follow the instructions.

1. Open any text editor.

2. Define the styles according to the JSON syntax. Take a look at the following example:

```
"styles": {
    "large-font": {
        "fontSize": 25
    },
    "very-large-font": {
        "fontSize": 40
    }
}
```

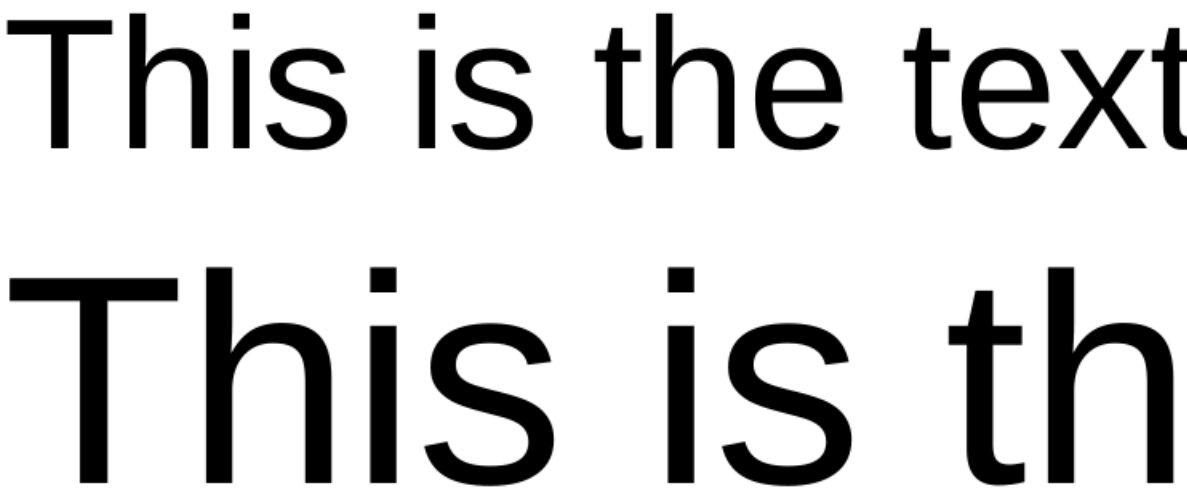
3. Save as a *.json* file, for example *style.json*.
4. Then create another document in **Text Editor**.
5. Use a class directive, as in the following example:

```
..class:: large-font

This is the text in font 25.

..class:: very-large-font

This is the text in font 40.
```

6. Save as an *.rst* file.
7. Use rst2pdf to generate a PDF.

Note: Remember to attach the stylesheet you created.

You should get a result like the one in the image



below:

Interpreted text roles

Learn what interpreted text roles are and how to use them

Interpreted text roles are an extension mechanism for inline markup in reStructuredText.

Standard roles are described in reStructuredText Interpreted Text Roles. See the Interpreted Text section in the reStructuredText Markup Specification for syntax details. Interpreted text roles are defined in stylesheet in the same way as classes. They are generally similar to classes. The main difference between them is that classes style an entire single paragraph after you call a directive, while roles style only the portion of text selected in the appropriate way.

Using interpreted text roles

Learn what to do to use interpreted text roles in your document.

1. Open **Text Editor**.
2. Define the styles according to the JSON syntax. Take a look at the following example:

```
"styles": {
    "redtext": {
        "textColor": red
    },
}
```

3. Save as a *.json* file, for example *style.json*.
4. Then create another document in a text editor.
5. Use an interpreted text role, as in the following example:

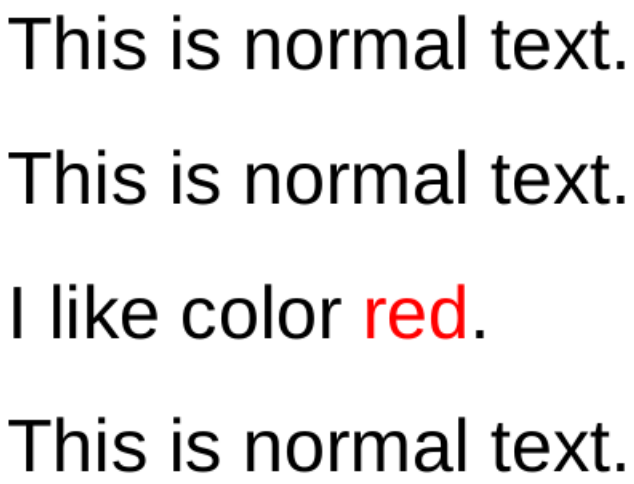
```
This is normal text.

.. role:: redtext

This is normal text.

I like color :redtext:`red`.

This is normal text.
```

6. Save as an *.rst* file.

7. Use rst2pdf to generate a PDF.

   Note: Remember to attach the stylesheet you created.

   You should get a result like the one in the image



   below:

Available attributes

Here you have listed the only attributes that work on styles when used for interpreted roles (inline styles).

- `fontName`
- `fontSize`
- `textColor`
- `backColor`

For more information about this, please check the rST docs

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` use 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

- `kerning` switches to document to DejaVU Sans font and enables kerning

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

These stylesheets 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

These stylesheets change the appearance of the code highlight.

- `abap`
- `algol_nu`
- `algol`
- `arduino`
- `autumn`
- `borland`
- `bw`
- `colorful`
- `default`
- `emacs`
- `friendly`
- `fruity`
- `igor`
- `lovelace`
- `manni`
- `monokai`
- `murphy`
- `native`
- `paraiso-dark`
- `paraiso-light`
- `pastie`
- `perldoc`
- `rainbow_dash`
- `rrt`
- `sas`
- `solarized-dark`
- `solarized-light`
- `sphinx`
- `stata-dark`
- `stata-light`
- `stata`
- `styles`
- `tango`

- `trac`
- `vim`
- `vs`
- `xcode`

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

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

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

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
```

Stylesheet syntax

Learn how to build your own stylesheet.

It's a JSON file with several elements in it. If you want to learn more about JSON, go to json.org.

Style definition

Learn more about configuring the style.

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

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

This means that the style called `normal` inherits style `base`. So, each property not defined in the normal style is taken from the base style.

It is recommended not to 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.

Page size and margins

Learn how to change alignment on the page.

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

Here's the default stylesheet's element:

```
"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 in (inches) or cm (centimeters).

When both width/height and size are specified, size is 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. On the other hand, if you display it on-screen on a two-page format (common in many PDF readers, nice for ebooks), a negative value may be pleasant.

Page layout

Learn how to configure the number and appearance of columns.

By default, your document has 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 three 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, bottom position, width, height, left padding, bottom
 padding, right padding, top padding]
```

All the padding values are optional and default to 6 points.

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

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

And this means "the top 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 you can have more than one template, there is a way to specify which one you 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:

```
.. raw:: pdf
PageBreak twoColumn
```

That triggers a page break, and the new page uses the twoColumn template.

The supported page template options and their defaults are:

- `showHeader` : True
- `defaultHeader` : None

Note: The above header options have the same effect as the header directive in the document.

- `showFooter` : True
- `defaultFooter` :None

Note: The above footer options have the same effect as the footer directive in the document

- `background` :None

The background should be an image, which is centered in your page or stretched to match your page size, depending on the `--fit-background-mode` option, so use with caution.

Styling your document

Which styles you need to modify to achieve your desired result is not obvious. In this section, you learn some hints and pointers to that effect.

The base styles

There are three styles which have great effect, they are `base`, `normal` and `bodytext`.

Here’s an example, the `twelvepoint` stylesheet:

```
{"styles": [["base", {"fontSize": 12}]]}
```

Since all other styles inherit `base`, changing the `fontSize` changes the `fontSize` for everything in your document.

The `normal` style is meant for most elements, so usually it's the same as changing `base`.

The `bodytext` style is for elements that form paragraphs. So, for example, you can set your document to be left-aligned like this:

```
{"styles": [["bodytext", {"alignment": "left"}]]}
```

There are elements, however, that don't inherit from `bodytext`, for example headings and the styles used in the table of contents. Those are elements that are not real paragraphs, so they should not follow the indentation and spacing you use for your document's main content.

The `heading` style is inherited by all sorts of titles: section titles, topic titles, admonition titles, etc.

Syntax highlighting

Learn about the options for highlighting codeblocks.

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. See pygments.org for a list of those currently supported.

You can use the `linenos` option to display line numbers:

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

You can use the `hl_lines` option to emphasize certain lines by dimming the other lines. This parameter takes a space separated list of line numbers. The other lines are then styled with the class `pygments_diml` that defaults to gray. For example, to highlight `print "line a"` and `print "line b"`:

```
def myFun(x,y):
    print "line a"
    print "line b"
    print "line c"
```

The overall look of a code block is controlled by the "code" style or by a class you apply to it using the `.. class::` directive. Additionally, if you want to change some properties when using different languages, you can define styles with the name of the language. For example, a `python` style will be applied to code blocks created with `.. code-block:: python`.

The look of the line numbers is controlled by the `linenumbers` style.

File inclusion

Learn how to add an external code file.

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

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

This gives a warning when `setup.py` doesn't exist or can't be opened.

Include with boundaries

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

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

Let's display a class from rst2pdf:

```
.. code-block:: python
:include: assets/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)
```

Licenses

Here you can find the license for rst2pdf.

This is the license for rst2pdf:

```
The above copyright notice and this permission notice shall be included in
 all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED “AS IS”, WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES
OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN
 NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE
FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT,
 TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION
WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
```

Some fragments of rst2pdf are copied from ReportLab under the following license:

```
Copyright © 2000-2008, ReportLab Inc.
All rights reserved.

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

* Redistributions of source code must retain this above copyright notice,
 this list of conditions and the following disclaimer.
* Redistributions in binary form must reproduce the above copyright notice,
 this list of conditions and/or other materials provided
with the distribution.
* Neither the name of the company nor the names of its contributors may be
 used to endorse or promote products derived from this
software without specific prior written permission.

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