
exam Documentation

Release

mi

Oct 10, 2018

CONTENTS:

1	Introduction	3
1.1	Paragraphs	3
1.2	Inline markup	3
1.3	Lists and Quote-like blocks	4
1.4	Source Code	5
1.5	Tables	5
1.6	Hyperlinks	5
1.6.1	External links	5
1.6.2	Internal links	6
1.7	Sections	6
1.8	Citations	6
1.9	Comments	6
1.10	Source encoding	6
2	Section to cross-reference	7
3	Directives	9
4	Installation!	13
5	Quickstart	15
6	This is a tables	17
6.1	Tables	17
6.1.1	Tables 1	17
6.2	Doctest blocks	18
6.3	Definition lists	18
6.4	Broken lines	18
7	Images	19
7.1	Figures	19
8	Index and tables	21
	Bibliography	23

Table of Contents

- *Welcome to prueba's documentation!*
- *Index and tables*

INTRODUCTION

This section is a brief introduction to reStructuredText (reST) concepts and syntax, intended to provide authors with enough information to author documents productively. Since reST was designed to be a simple, unobtrusive markup language, this will not take too long.

1.1 Paragraphs

The paragraph is the most basic block in a reST document. Paragraphs are simply chunks of text separated by one or more blank lines. As in Python, indentation is significant in reST, so all lines of the same paragraph must be left-aligned to the same level of indentation.

1.2 Inline markup

The standard reST inline markup is quite simple: use

- one asterisk: *text* for emphasis (italics),
- two asterisks: **text** for strong emphasis (boldface), and
- backquotes: `text` for code samples.

If asterisks or backquotes appear in running text and could be confused with inline markup delimiters, they have to be escaped with a backslash.

Be aware of some restrictions of this markup:

- it may not be nested,
- content may not start or end with whitespace: `* text*` is wrong,
- it must be separated from surrounding text by non-word characters. Use a backslash escaped space to work around that: `this is one word.`

These restrictions may be lifted in future versions of the docutils.

reST also allows for custom “interpreted text roles”, which signify that the enclosed text should be interpreted in a specific way. Sphinx uses this to provide semantic markup and cross-referencing of identifiers, as described in the appropriate section. The general syntax is `:rolename:`content``.

Standard reST provides the following roles:

- emphasis – alternate spelling for *emphasis*
- strong – alternate spelling for **strong**
- literal – alternate spelling for `literal`

- subscript – subscript text
- superscript – superscript text
- title-reference – for titles of books, periodicals, and other materials

1.3 Lists and Quote-like blocks

List markup is natural: just place an asterisk at the start of a paragraph and indent properly. The same goes for numbered lists; they can also be autonumbered using a # sign:

- This is a bulleted list.
 - It has two items, the second item uses two lines.
1. This is a numbered list.
 2. It has two items too.
 3. This is a numbered list.
 4. It has two items too.

Nested lists are possible, but be aware that they must be separated from the parent list items by blank lines:

- this is
- a list
 - with a nested list
 - and some subitems
- and here the parent list continues

Definition lists are created as follows:

term (up to a line of text) Definition of the term, which must be indented
and can even consist of multiple paragraphs

next term Description.

Note that the term cannot have more than one line of text.

Quoted paragraphs are created by just indenting them more than the surrounding paragraphs.

Line blocks are a way of preserving line breaks:

These lines are
broken exactly like in
the source file.

There are also several more special blocks available:

- field lists
- option lists
- quoted literal blocks
- doctest blocks

1.4 Source Code

Literal code blocks (ref) are introduced by ending a paragraph with the special marker `::`. The literal block must be indented (and, like all paragraphs, separated from the surrounding ones by blank lines):

This is a normal text paragraph. The next paragraph is a code sample:

```
It is not processed in any way, except
that the indentation is removed.

It can span multiple lines.
```

This is a normal text paragraph again.

The handling of the `::` marker is smart:

- If it occurs as a paragraph of its own, that paragraph is completely left out of the document.
- If it is preceded by whitespace, the marker is removed.
- If it is preceded by non-whitespace, the marker is replaced by a single colon.

That way, the second sentence in the above example’s first paragraph would be rendered as “The next paragraph is a code sample:”.

1.5 Tables

Two forms of tables are supported. For grid tables , you have to “paint” the cell grid yourself. They look like this:

Header row, column 1 (header rows optional)	Header 2	Header 3	Header 4
body row 1, column 1	column 2	column 3	column 4
body row 2	

Simple tables (ref) are easier to write, but limited: they must contain more than one row, and the first column cannot contain multiple lines. They look like this:

A	B	A and B
False	False	False
True	False	False
False	True	False
True	True	True

1.6 Hyperlinks

1.6.1 External links

Use `Link text` for inline web links. If the link text should be the web address, you don’t need special markup at all, the parser finds links and mail addresses in ordinary text.

You can also separate the link and the target definition, like this:

This is a paragraph that contains a `link`.

1.6.2 Internal links

Internal linking is done via a special reST role provided by Sphinx, see the section on specific markup, Cross-referencing arbitrary locations.

1.7 Sections

Section headers are created by underlining (and optionally overlining) the section title with a punctuation character, at least as long as the text:

Normally, there are no heading levels assigned to certain characters as the structure is determined from the succession of headings. However, this convention is used in Python’s Style Guide for documenting which you may follow:

- # with overline, for parts
- – with overline, for chapters
- =, for sections
- -, for subsections
- ^, for subsubsections
- ”, for paragraphs

Of course, you are free to use your own marker characters (see the reST documentation), and use a deeper nesting level, but keep in mind that most target formats (HTML, LaTeX) have a limited supported nesting depth.

1.8 Citations

Standard reST citations are supported, with the additional feature that they are “global”, i.e. all citations can be referenced from all files. Use them like so:

Lorem ipsum *[Ref]* dolor sit amet.

Citation usage is similar to footnote usage, but with a label that is not numeric or begins with #

1.9 Comments

Every explicit markup block which isn’t a valid markup construct (like the footnotes above) is regarded as a comment. For example:

You can indent text after a comment start to form multiline comments:

1.10 Source encoding

Since the easiest way to include special characters like em dashes or copyright signs in reST is to directly write them as Unicode characters, one has to specify an encoding. Sphinx assumes source files to be encoded in UTF-8 by default; you can change this with the `source_encoding` config value.

SECTION TO CROSS-REFERENCE

This is the text of the section.

It refers to the section itself, see For more details, see *Introduction*.

Link to another *Inline markup*.

Link to another *The same* at the same Inline markup.

Link to *Images*.

Link to *Figures*.

DIRECTIVES

Note: This function is not suitable for sending spam e-mails.

Warning: An important bit of information about an API that a user should be very aware of when using whatever bit of API the warning pertains to. The content of the directive should be written in complete sentences and include all appropriate punctuation. This differs from note in that it is recommended over note for information regarding security.

New in version 2.5: The *spam* parameter.

Deprecated since version 3.1: Use `spam()` instead.

See also:

Module `zipfile` Documentation of the `zipfile` standard module.

GNU tar manual, Basic Tar Format Documentation for tar archive files, including GNU tar extensions.

title

- A list of
- short items
- that should be
- displayed
- horizontally

environment A structure where information about all documents under the root is saved, and used for cross-referencing. The environment is pickled after the parsing stage, so that successive runs only need to read and parse new and changed documents.

source directory The directory which, including its subdirectories, contains all source files for one Sphinx project.

Sphinx Sphinx is a tool that makes it easy to create intelligent and beautiful documentation. It was originally created for the Python documentation, and it has excellent facilities for the documentation of software projects in a range of languages.

RST RST is an easy-to-read, what-you-see-is-what-you-get plain text markup syntax and parser system. It is useful for in-line program documentation (such as Python docstrings), for quickly creating simple web pages, and for standalone documents. RST is designed for extensibility for specific application domains. The RST parser is a component of Docutils.

Sublime Text Sublime Text is a sophisticated text editor for code, markup and prose. You'll love the slick user interface, extraordinary features and amazing performance.

Note: This function is not suitable for sending spam e-mails.

How to include images in project:



```
try_stmt ::= try1_stmt | try2_stmt
try1_stmt ::= "try" ":" suite
              ("except" [expression ["," target]] ":" suite)+
              ["else" ":" suite]
              ["finally" ":" suite]
try2_stmt ::= "try" ":" suite
              "finally" ":" suite
```

Danger: Beware killer rabbits!

Topic Title

Subsequent indented lines comprise the body of the topic, and are interpreted as body elements.

Sidebar Title

Optional Sidebar Subtitle

Subsequent indented lines comprise the body of the sidebar, and are interpreted as body elements.

The ‘rm’ command is very dangerous. If you are logged in as root and enter

```
cd /
rm -rf *
```

you will erase the entire contents of your file system.

This paragraph might be rendered in a custom way.

Parsing the above results in the following pseudo-XML:

<container classes="custom">

<paragraph> This paragraph might be rendered in a custom way.

The “container” directive is the equivalent of HTML’s <div> element. It may be used to group a sequence of elements for user- or application-specific purposes.

INSTALLATION!

- This is a bulleted list.
 - It has two items, the second item uses two lines.
1. This is a numbered list.
 2. It has two items too.
 3. This is a numbered list.
 4. It has two items too.

term (up to a line of text) Definition of the term, which must be indented
and can even consist of multiple paragraphs

next term Description.

These lines are
broken exactly like in
the source file.

QUICKSTART

The standard reST inline markup is quite simple: use one asterisk: *text* for emphasis (italics), two asterisks: **text** for strong emphasis (boldface), and backquotes: `text` for code samples.

If asterisks or backquotes appear in running text and could be confused with inline markup delimiters, they have to be escaped with a backslash.

This is a normal text paragraph. The next paragraph is a code sample:

```
It is not processed in any way, except
that the indentation is removed.

It can span multiple lines.
```

This is a normal text paragraph again.

Header row, column 1 (header rows optional)	Header 2	Header 3	Header 4
body row 1, column 1	column 2	column 3	column 4
body row 2	

THIS IS A TABLES

6.1 Tables

6.1.1 Tables 1

H4 – Subsubsection

Table 6.1: Truth
table for “not”

A	not A
False	True
True	False

Header row, column 1 (header rows optional)	Header 2	Header 3	Header 4
body row 1, column 1	column 2	column 3	column 4
body row 2	

The tables are as seen

A	B	A and B
False	False	False
True	False	False
False	True	False
True	True	True

Use [Link text](#) for inline web links. If the link text should be the web address, you don’t need special markup at all, the parser finds links and mail addresses in ordinary text.

This is a paragraph that contains [a link](#).

The handling of the :: marker is smart:

If it occurs as a paragraph of its own, that paragraph is completely left out of the document. If it is preceded by whitespace, the marker is removed. If it is preceded by non-whitespace, the marker is replaced by a single colon. That way, the second sentence in the above example’s first paragraph would be rendered as “The next paragraph is a code sample:”.

6.2 Doctest blocks

```
>>> 1 + 1
2
```

6.3 Definition lists

term (up to a line of text) Definition of the term, which must be indented
and can even consist of multiple paragraphs

next term Description.

6.4 Broken lines

These lines are
broken exactly like in
the source file.

IMAGES




How to include images in project:



7.1 Figures



Fig. 7.1: This is the caption of the figure (a simple paragraph).
The legend consists of all elements after the caption. In this case, the legend consists of this paragraph and the following table:

Symbol	Meaning
	Campground
	Lake
	Mountain

INDEX AND TABLES

- genindex
- modindex
- search

BIBLIOGRAPHY

[Ref] Book or article reference, URL or whatever.