**ReStructuredText Command Language 380-F01**

**(adapted from:** <https://www.sphinx-doc.org/en/master/usage/restructuredtext/index.html> **)**

reStructuredText (reST) is the default plaintext markup language used by both Docutils and Sphinx. Docutils provides the basic reStructuredText syntax, while Sphinx extends this to support additional functionality.

The below guides go through the most important aspects of reST. For the authoritative reStructuredText reference, refer to the [docutils documentation](http://docutils.sourceforge.net/rst.html).

* [reStructuredText Primer](http://www.sphinx-doc.org/en/master/usage/restructuredtext/basics.html)
  + [Paragraphs](http://www.sphinx-doc.org/en/master/usage/restructuredtext/basics.html#paragraphs)
  + [Inline markup](http://www.sphinx-doc.org/en/master/usage/restructuredtext/basics.html#inline-markup)
  + [Lists and Quote-like blocks](http://www.sphinx-doc.org/en/master/usage/restructuredtext/basics.html#lists-and-quote-like-blocks)
  + [Literal blocks](http://www.sphinx-doc.org/en/master/usage/restructuredtext/basics.html#literal-blocks)
  + [Doctest blocks](http://www.sphinx-doc.org/en/master/usage/restructuredtext/basics.html#doctest-blocks)
  + [Tables](http://www.sphinx-doc.org/en/master/usage/restructuredtext/basics.html#tables)
  + [Hyperlinks](http://www.sphinx-doc.org/en/master/usage/restructuredtext/basics.html#hyperlinks)
  + [Sections](http://www.sphinx-doc.org/en/master/usage/restructuredtext/basics.html#sections)
  + [Field Lists](http://www.sphinx-doc.org/en/master/usage/restructuredtext/basics.html#field-lists)
  + [Roles](http://www.sphinx-doc.org/en/master/usage/restructuredtext/basics.html#roles)
  + [Explicit Markup](http://www.sphinx-doc.org/en/master/usage/restructuredtext/basics.html#explicit-markup)
  + [Directives](http://www.sphinx-doc.org/en/master/usage/restructuredtext/basics.html#directives)
  + [Images](http://www.sphinx-doc.org/en/master/usage/restructuredtext/basics.html#images)
  + [Footnotes](http://www.sphinx-doc.org/en/master/usage/restructuredtext/basics.html#footnotes)
  + [Citations](http://www.sphinx-doc.org/en/master/usage/restructuredtext/basics.html#citations)
  + [Substitutions](http://www.sphinx-doc.org/en/master/usage/restructuredtext/basics.html#substitutions)
  + [Comments](http://www.sphinx-doc.org/en/master/usage/restructuredtext/basics.html#comments)
  + [HTML Metadata](http://www.sphinx-doc.org/en/master/usage/restructuredtext/basics.html#html-metadata)
  + [Source encoding](http://www.sphinx-doc.org/en/master/usage/restructuredtext/basics.html#source-encoding)
  + [Gotchas](http://www.sphinx-doc.org/en/master/usage/restructuredtext/basics.html#gotchas)
* [Roles](http://www.sphinx-doc.org/en/master/usage/restructuredtext/roles.html)
  + [Cross-referencing syntax](http://www.sphinx-doc.org/en/master/usage/restructuredtext/roles.html#cross-referencing-syntax)
  + [Math](http://www.sphinx-doc.org/en/master/usage/restructuredtext/roles.html#math)
  + [Other semantic markup](http://www.sphinx-doc.org/en/master/usage/restructuredtext/roles.html#other-semantic-markup)
  + [Substitutions](http://www.sphinx-doc.org/en/master/usage/restructuredtext/roles.html#substitutions)
* [Directives](http://www.sphinx-doc.org/en/master/usage/restructuredtext/directives.html)
  + [Table of contents](http://www.sphinx-doc.org/en/master/usage/restructuredtext/directives.html#table-of-contents)
  + [Paragraph-level markup](http://www.sphinx-doc.org/en/master/usage/restructuredtext/directives.html#paragraph-level-markup)
  + [Showing code examples](http://www.sphinx-doc.org/en/master/usage/restructuredtext/directives.html#showing-code-examples)
  + [Glossary](http://www.sphinx-doc.org/en/master/usage/restructuredtext/directives.html#glossary)
  + [Meta-information markup](http://www.sphinx-doc.org/en/master/usage/restructuredtext/directives.html#meta-information-markup)
  + [Index-generating markup](http://www.sphinx-doc.org/en/master/usage/restructuredtext/directives.html#index-generating-markup)
  + [Including content based on tags](http://www.sphinx-doc.org/en/master/usage/restructuredtext/directives.html#including-content-based-on-tags)
  + [Tables](http://www.sphinx-doc.org/en/master/usage/restructuredtext/directives.html#tables)
  + [Math](http://www.sphinx-doc.org/en/master/usage/restructuredtext/directives.html#math)
  + [Grammar production displays](http://www.sphinx-doc.org/en/master/usage/restructuredtext/directives.html#grammar-production-displays)
* [Field Lists](http://www.sphinx-doc.org/en/master/usage/restructuredtext/field-lists.html)
  + [File-wide metadata](http://www.sphinx-doc.org/en/master/usage/restructuredtext/field-lists.html#file-wide-metadata)
* [Domains](http://www.sphinx-doc.org/en/master/usage/restructuredtext/domains.html)
  + [Basic Markup](http://www.sphinx-doc.org/en/master/usage/restructuredtext/domains.html#basic-markup)
  + [The Python Domain](http://www.sphinx-doc.org/en/master/usage/restructuredtext/domains.html#the-python-domain)
  + [The C Domain](http://www.sphinx-doc.org/en/master/usage/restructuredtext/domains.html#the-c-domain)
  + [The C++ Domain](http://www.sphinx-doc.org/en/master/usage/restructuredtext/domains.html#cpp-domain)
  + [The Standard Domain](http://www.sphinx-doc.org/en/master/usage/restructuredtext/domains.html#the-standard-domain)
  + [The JavaScript Domain](http://www.sphinx-doc.org/en/master/usage/restructuredtext/domains.html#the-javascript-domain)
  + [The reStructuredText domain](http://www.sphinx-doc.org/en/master/usage/restructuredtext/domains.html#the-restructuredtext-domain)
  + [The Math Domain](http://www.sphinx-doc.org/en/master/usage/restructuredtext/domains.html#the-math-domain)
  + [More domains](http://www.sphinx-doc.org/en/master/usage/restructuredtext/domains.html#more-domains)

**reStructuredText Primer**

reStructuredText is the default plaintext markup language used by Sphinx. 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.

See also

The authoritative [reStructuredText User Documentation](http://docutils.sourceforge.net/rst.html). The “ref” links in this document link to the description of the individual constructs in the reST reference.

**Paragraphs**

The paragraph ([ref](http://docutils.sourceforge.net/docs/ref/rst/restructuredtext.html#paragraphs)) 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.

**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: thisis\ \*one\*\ word.

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

It is also possible to replace or expand upon some of this inline markup with roles. Refer to [Roles](http://www.sphinx-doc.org/en/master/usage/restructuredtext/basics.html#rst-roles-alt) for more information.

**Lists and Quote-like blocks**

List markup ([ref](http://docutils.sourceforge.net/docs/ref/rst/restructuredtext.html#bullet-lists)) 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.

#. This is a numbered list.

#. 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 ([ref](http://docutils.sourceforge.net/docs/ref/rst/restructuredtext.html#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 ([ref](http://docutils.sourceforge.net/docs/ref/rst/restructuredtext.html#block-quotes)) are created by just indenting them more than the surrounding paragraphs.

Line blocks ([ref](http://docutils.sourceforge.net/docs/ref/rst/restructuredtext.html#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 ([ref](http://docutils.sourceforge.net/docs/ref/rst/restructuredtext.html#field-lists), with caveats noted in [Field Lists](http://www.sphinx-doc.org/en/master/usage/restructuredtext/basics.html#rst-field-lists))
* option lists ([ref](http://docutils.sourceforge.net/docs/ref/rst/restructuredtext.html#option-lists))
* quoted literal blocks ([ref](http://docutils.sourceforge.net/docs/ref/rst/restructuredtext.html#quoted-literal-blocks))
* doctest blocks ([ref](http://docutils.sourceforge.net/docs/ref/rst/restructuredtext.html#doctest-blocks))

**Literal blocks**

Literal code blocks ([ref](http://docutils.sourceforge.net/docs/ref/rst/restructuredtext.html#literal-blocks)) 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:”.

Code highlighting can be enabled for these literal blocks on a document-wide basis using the [highlight](http://www.sphinx-doc.org/en/master/usage/restructuredtext/directives.html#_blank) directive and on a project-wide basis using the [highlight\_language](http://www.sphinx-doc.org/en/master/usage/configuration.html#confval-highlight_language) configuration option. The [code-block](http://www.sphinx-doc.org/en/master/usage/restructuredtext/directives.html#_blank) directive can be used to set highlighting on a block-by-block basis. These directives are discussed later.

**Doctest blocks**

Doctest blocks ([ref](http://docutils.sourceforge.net/docs/ref/rst/restructuredtext.html#doctest-blocks)) are interactive Python sessions cut-and-pasted into docstrings. They do not require the [literal blocks](http://www.sphinx-doc.org/en/master/usage/restructuredtext/basics.html#rst-literal-blocks) syntax. The doctest block must end with a blank line and should *not* end with with an unused prompt:

>>> 1 + 1

2

**Tables**

For *grid tables* ([ref](http://docutils.sourceforge.net/docs/ref/rst/restructuredtext.html#grid-tables)), you have to “paint” the cell grid yourself. They look like this:

+------------------------+------------+----------+----------+

| Header row, column 1 | Header 2 | Header 3 | Header 4 |

| (header rows optional) | | | |

+========================+============+==========+==========+

| body row 1, column 1 | column 2 | column 3 | column 4 |

+------------------------+------------+----------+----------+

| body row 2 | ... | ... | |

+------------------------+------------+----------+----------+

*Simple tables* ([ref](http://docutils.sourceforge.net/docs/ref/rst/restructuredtext.html#simple-tables)) are easier to write, but limited: they must contain more than one row, and the first column cells 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

===== ===== =======

Two more syntaxes are supported: *CSV tables* and *List tables*. They use an *explicit markup block*. Refer to [Tables](http://www.sphinx-doc.org/en/master/usage/restructuredtext/directives.html#table-directives) for more information.

**Hyperlinks**

**External links**

Use `Link text <https://domain.invalid/>`\_ 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.

Important

There must be a space between the link text and the opening < for the URL.

You can also separate the link and the target definition ([ref](http://docutils.sourceforge.net/docs/ref/rst/restructuredtext.html#hyperlink-targets)), like this:

This is a paragraph that contains `a link`\_.

.. \_a link: https://domain.invalid/

**Internal links**

Internal linking is done via a special reST role provided by Sphinx, see the section on specific markup, [Cross-referencing arbitrary locations](http://www.sphinx-doc.org/en/master/usage/restructuredtext/roles.html#ref-role).

**Sections**

Section headers ([ref](http://docutils.sourceforge.net/docs/ref/rst/restructuredtext.html#sections)) are created by underlining (and optionally overlining) the section title with a punctuation character, at least as long as the text:

=================

This is a heading

=================

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](https://docs.python.org/devguide/documenting.html#style-guide) 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.

**Field Lists**

Field lists ([ref](http://docutils.sourceforge.net/docs/ref/rst/restructuredtext.html#field-lists)) are sequences of fields marked up like this:

:fieldname: Field content

They are commonly used in Python documentation:

def my\_function(my\_arg, my\_other\_arg):

"""A function just for me.

:param my\_arg: The first of my arguments.

:param my\_other\_arg: The second of my arguments.

:returns: A message (just for me, of course).

"""

Sphinx extends standard docutils behavior and intercepts field lists specified at the beginning of documents. Refer to [Field Lists](http://www.sphinx-doc.org/en/master/usage/restructuredtext/field-lists.html) for more information.

**Roles**

A role or “custom interpreted text role” ([ref](http://docutils.sourceforge.net/docs/ref/rst/restructuredtext.html#roles)) is an inline piece of explicit markup. It signifies that 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`.

Docutils supports the following roles:

* [emphasis](http://docutils.sourceforge.net/docs/ref/rst/roles.html#emphasis) – equivalent of \*emphasis\*
* [strong](http://docutils.sourceforge.net/docs/ref/rst/roles.html#strong) – equivalent of \*\*strong\*\*
* [literal](http://docutils.sourceforge.net/docs/ref/rst/roles.html#literal) – equivalent of ``literal``
* [subscript](http://docutils.sourceforge.net/docs/ref/rst/roles.html#subscript) – subscript text
* [superscript](http://docutils.sourceforge.net/docs/ref/rst/roles.html#superscript) – superscript text
* [title-reference](http://docutils.sourceforge.net/docs/ref/rst/roles.html#title-reference) – for titles of books, periodicals, and other materials

Refer to [Roles](http://www.sphinx-doc.org/en/master/usage/restructuredtext/roles.html) for roles added by Sphinx.

**Explicit Markup**

“Explicit markup” ([ref](http://docutils.sourceforge.net/docs/ref/rst/restructuredtext.html#explicit-markup-blocks)) is used in reST for most constructs that need special handling, such as footnotes, specially-highlighted paragraphs, comments, and generic directives.

An explicit markup block begins with a line starting with .. followed by whitespace and is terminated by the next paragraph at the same level of indentation. (There needs to be a blank line between explicit markup and normal paragraphs. This may all sound a bit complicated, but it is intuitive enough when you write it.)

**Directives**

A directive ([ref](http://docutils.sourceforge.net/docs/ref/rst/restructuredtext.html#directives)) is a generic block of explicit markup. Along with roles, it is one of the extension mechanisms of reST, and Sphinx makes heavy use of it.

Docutils supports the following directives:

* Admonitions: [attention](http://docutils.sourceforge.net/docs/ref/rst/directives.html#attention), [caution](http://docutils.sourceforge.net/docs/ref/rst/directives.html#caution), [danger](http://docutils.sourceforge.net/docs/ref/rst/directives.html#danger), [error](http://docutils.sourceforge.net/docs/ref/rst/directives.html#error), [hint](http://docutils.sourceforge.net/docs/ref/rst/directives.html#hint), [important](http://docutils.sourceforge.net/docs/ref/rst/directives.html#important), [note](http://docutils.sourceforge.net/docs/ref/rst/directives.html#note), [tip](http://docutils.sourceforge.net/docs/ref/rst/directives.html#tip), [warning](http://docutils.sourceforge.net/docs/ref/rst/directives.html#warning) and the generic [admonition](http://docutils.sourceforge.net/docs/ref/rst/directives.html#admonitions). (Most themes style only “note” and “warning” specially.)
* Images:
  + [image](http://docutils.sourceforge.net/docs/ref/rst/directives.html#image) (see also [Images](http://www.sphinx-doc.org/en/master/usage/restructuredtext/basics.html#images) below)
  + [figure](http://docutils.sourceforge.net/docs/ref/rst/directives.html#figure) (an image with caption and optional legend)
* Additional body elements:
  + [contents](http://docutils.sourceforge.net/docs/ref/rst/directives.html#table-of-contents) (a local, i.e. for the current file only, table of contents)
  + [container](http://docutils.sourceforge.net/docs/ref/rst/directives.html#container) (a container with a custom class, useful to generate an outer <div> in HTML)
  + [rubric](http://docutils.sourceforge.net/docs/ref/rst/directives.html#rubric) (a heading without relation to the document sectioning)
  + [topic](http://docutils.sourceforge.net/docs/ref/rst/directives.html#topic), [sidebar](http://docutils.sourceforge.net/docs/ref/rst/directives.html#sidebar) (special highlighted body elements)
  + [parsed-literal](http://docutils.sourceforge.net/docs/ref/rst/directives.html#parsed-literal) (literal block that supports inline markup)
  + [epigraph](http://docutils.sourceforge.net/docs/ref/rst/directives.html#epigraph) (a block quote with optional attribution line)
  + [highlights](http://docutils.sourceforge.net/docs/ref/rst/directives.html#highlights), [pull-quote](http://docutils.sourceforge.net/docs/ref/rst/directives.html#pull-quote) (block quotes with their own class attribute)
  + [compound](http://docutils.sourceforge.net/docs/ref/rst/directives.html#compound-paragraph) (a compound paragraph)
* Special tables:
  + [table](http://docutils.sourceforge.net/docs/ref/rst/directives.html#table) (a table with title)
  + [csv-table](http://docutils.sourceforge.net/docs/ref/rst/directives.html#csv-table) (a table generated from comma-separated values)
  + [list-table](http://docutils.sourceforge.net/docs/ref/rst/directives.html#list-table) (a table generated from a list of lists)
* Special directives:
  + [raw](http://docutils.sourceforge.net/docs/ref/rst/directives.html#raw-data-pass-through) (include raw target-format markup)
  + [include](http://docutils.sourceforge.net/docs/ref/rst/directives.html#include) (include reStructuredText from another file) – in Sphinx, when given an absolute include file path, this directive takes it as relative to the source directory
  + [class](http://docutils.sourceforge.net/docs/ref/rst/directives.html#class) (assign a class attribute to the next element) [1](http://www.sphinx-doc.org/en/master/usage/restructuredtext/basics.html#id2)
* HTML specifics:
  + [meta](http://docutils.sourceforge.net/docs/ref/rst/directives.html#meta) (generation of HTML <meta> tags, see also [HTML Metadata](http://www.sphinx-doc.org/en/master/usage/restructuredtext/basics.html#html-meta) below)
  + [title](http://docutils.sourceforge.net/docs/ref/rst/directives.html#metadata-document-title) (override document title)
* Influencing markup:
  + [default-role](http://docutils.sourceforge.net/docs/ref/rst/directives.html#default-role) (set a new default role)
  + [role](http://docutils.sourceforge.net/docs/ref/rst/directives.html#role) (create a new role)

Since these are only per-file, better use Sphinx’s facilities for setting the [default\_role](http://www.sphinx-doc.org/en/master/usage/configuration.html#confval-default_role).

Warning

Do *not* use the directives [sectnum](http://docutils.sourceforge.net/docs/ref/rst/directives.html#sectnum), [header](http://docutils.sourceforge.net/docs/ref/rst/directives.html#header) and [footer](http://docutils.sourceforge.net/docs/ref/rst/directives.html#footer).

Directives added by Sphinx are described in [Directives](http://www.sphinx-doc.org/en/master/usage/restructuredtext/directives.html).

Basically, a directive consists of a name, arguments, options and content. (Keep this terminology in mind, it is used in the next chapter describing custom directives.) Looking at this example,

.. function:: foo(x)

foo(y, z)

:module: some.module.name

Return a line of text input from the user.

function is the directive name. It is given two arguments here, the remainder of the first line and the second line, as well as one option module (as you can see, options are given in the lines immediately following the arguments and indicated by the colons). Options must be indented to the same level as the directive content.

The directive content follows after a blank line and is indented relative to the directive start.

**Images**

reST supports an image directive ([ref](http://docutils.sourceforge.net/docs/ref/rst/directives.html#image)), used like so:

.. image:: gnu.png

(options)

When used within Sphinx, the file name given (here gnu.png) must either be relative to the source file, or absolute which means that they are relative to the top source directory. For example, the file sketch/spam.rst could refer to the image images/spam.png as ../images/spam.png or /images/spam.png.

Sphinx will automatically copy image files over to a subdirectory of the output directory on building (e.g. the \_static directory for HTML output.)

Interpretation of image size options (width and height) is as follows: if the size has no unit or the unit is pixels, the given size will only be respected for output channels that support pixels. Other units (like pt for points) will be used for HTML and LaTeX output (the latter replaces pt by bp as this is the TeX unit such that 72bp=1in).

Sphinx extends the standard docutils behavior by allowing an asterisk for the extension:

.. image:: gnu.\*

Sphinx then searches for all images matching the provided pattern and determines their type. Each builder then chooses the best image out of these candidates. For instance, if the file name gnu.\* was given and two files gnu.pdf and gnu.png existed in the source tree, the LaTeX builder would choose the former, while the HTML builder would prefer the latter. Supported image types and choosing priority are defined at [Builders](http://www.sphinx-doc.org/en/master/usage/builders/index.html).

Note that image file names should not contain spaces.

Changed in version 0.4: Added the support for file names ending in an asterisk.

Changed in version 0.6: Image paths can now be absolute.

Changed in version 1.5: latex target supports pixels (default is 96px=1in).

**Footnotes**

For footnotes ([ref](http://docutils.sourceforge.net/docs/ref/rst/restructuredtext.html#footnotes)), use [#name]\_ to mark the footnote location, and add the footnote body at the bottom of the document after a “Footnotes” rubric heading, like so:

Lorem ipsum [#f1]\_ dolor sit amet ... [#f2]\_

.. rubric:: Footnotes

.. [#f1] Text of the first footnote.

.. [#f2] Text of the second footnote.

You can also explicitly number the footnotes ([1]\_) or use auto-numbered footnotes without names ([#]\_).

**Citations**

Standard reST citations ([ref](http://docutils.sourceforge.net/docs/ref/rst/restructuredtext.html#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.

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

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

**Substitutions**

reST supports “substitutions” ([ref](http://docutils.sourceforge.net/docs/ref/rst/restructuredtext.html#substitution-definitions)), which are pieces of text and/or markup referred to in the text by |name|. They are defined like footnotes with explicit markup blocks, like this:

.. |name| replace:: replacement \*text\*

or this:

.. |caution| image:: warning.png

:alt: Warning!

See the [reST reference for substitutions](http://docutils.sourceforge.net/docs/ref/rst/restructuredtext.html#substitution-definitions) for details.

If you want to use some substitutions for all documents, put them into [rst\_prolog](http://www.sphinx-doc.org/en/master/usage/configuration.html#confval-rst_prolog) or [rst\_epilog](http://www.sphinx-doc.org/en/master/usage/configuration.html#confval-rst_epilog) or put them into a separate file and include it into all documents you want to use them in, using the include directive. (Be sure to give the include file a file name extension differing from that of other source files, to avoid Sphinx finding it as a standalone document.)

Sphinx defines some default substitutions, see [Substitutions](http://www.sphinx-doc.org/en/master/usage/restructuredtext/roles.html#default-substitutions).

**Comments**

Every explicit markup block which isn’t a valid markup construct (like the footnotes above) is regarded as a comment ([ref](http://docutils.sourceforge.net/docs/ref/rst/restructuredtext.html#comments)). For example:

.. This is a comment.

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

..

This whole indented block

is a comment.

Still in the comment.

**HTML Metadata**

The meta directive ([ref](http://docutils.sourceforge.net/docs/ref/rst/directives.html#meta)) allows specifying the HTML [metadata element](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/meta) of a Sphinx documentation page. For example, the directive:

.. meta::

:description: The Sphinx documentation builder

:keywords: Sphinx, documentation, builder

will generate the following HTML output:

<meta name="description" content="The Sphinx documentation builder">

<meta name="keywords" content="Sphinx, documentation, builder">

Also, Sphinx will add the keywords as specified in the meta directive to the search index. Thereby, the lang attribute of the meta element is considered. For example, the directive:

.. meta::

:keywords: backup

:keywords lang=en: pleasefindthiskey pleasefindthiskeytoo

:keywords lang=de: bittediesenkeyfinden

adds the following words to the search indices of builds with different language configurations:

* pleasefindthiskey, pleasefindthiskeytoo to *English* builds;
* bittediesenkeyfinden to *German* builds;
* backup to builds in all languages.

**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](http://www.sphinx-doc.org/en/master/usage/configuration.html#confval-source_encoding) config value.

**Gotchas**

There are some problems one commonly runs into while authoring reST documents:

* **Separation of inline markup:** As said above, inline markup spans must be separated from the surrounding text by non-word characters, you have to use a backslash-escaped space to get around that. See [the reference](http://docutils.sourceforge.net/docs/ref/rst/restructuredtext.html#substitution-definitions) for the details.
* **No nested inline markup:** Something like \*see :func:`foo`\* is not possible.

Footnotes

[1](http://www.sphinx-doc.org/en/master/usage/restructuredtext/basics.html#id1)

When the default domain contains a class directive, this directive will be shadowed. Therefore, Sphinx re-exports it as rst-class.

**Roles**

Sphinx uses interpreted text roles to insert semantic markup into documents. They are written as :rolename:`content`.

Note

The default role (`content`) has no special meaning by default. You are free to use it for anything you like, e.g. variable names; use the [default\_role](http://www.sphinx-doc.org/en/master/usage/configuration.html#confval-default_role) config value to set it to a known role – the [any](http://www.sphinx-doc.org/en/master/usage/restructuredtext/roles.html#_blank) role to find anything or the [py:obj](http://www.sphinx-doc.org/en/master/usage/restructuredtext/domains.html#_blank) role to find Python objects are very useful for this.

See [Domains](http://www.sphinx-doc.org/en/master/usage/restructuredtext/domains.html) for roles added by domains.

**Cross-referencing syntax**

Cross-references are generated by many semantic interpreted text roles. Basically, you only need to write :role:`target`, and a link will be created to the item named *target* of the type indicated by *role*. The link’s text will be the same as *target*.

There are some additional facilities, however, that make cross-referencing roles more versatile:

* You may supply an explicit title and reference target, like in reST direct hyperlinks: :role:`title <target>` will refer to *target*, but the link text will be *title*.
* If you prefix the content with !, no reference/hyperlink will be created.
* If you prefix the content with ~, the link text will only be the last component of the target. For example, :py:meth:`~Queue.Queue.get` will refer to Queue.Queue.get but only display get as the link text. This does not work with all cross-reference roles, but is domain specific.

In HTML output, the link’s title attribute (that is e.g. shown as a tool-tip on mouse-hover) will always be the full target name.

**Cross-referencing anything**

:any:

New in version 1.3.

This convenience role tries to do its best to find a valid target for its reference text.

* First, it tries standard cross-reference targets that would be referenced by [doc](http://www.sphinx-doc.org/en/master/usage/restructuredtext/roles.html#_blank), [ref](http://www.sphinx-doc.org/en/master/usage/restructuredtext/roles.html#_blank) or [option](http://www.sphinx-doc.org/en/master/usage/restructuredtext/roles.html#_blank).

Custom objects added to the standard domain by extensions (see [Sphinx.add\_object\_type()](http://www.sphinx-doc.org/en/master/extdev/appapi.html#_blank)) are also searched.

* Then, it looks for objects (targets) in all loaded domains. It is up to the domains how specific a match must be. For example, in the Python domain a reference of :any:`Builder` would match the sphinx.builders.Builder class.

If none or multiple targets are found, a warning will be emitted. In the case of multiple targets, you can change “any” to a specific role.

This role is a good candidate for setting [default\_role](http://www.sphinx-doc.org/en/master/usage/configuration.html#confval-default_role). If you do, you can write cross-references without a lot of markup overhead. For example, in this Python function documentation

.. function:: install()

This function installs a `handler` for every signal known by the

`signal` module. See the section `about-signals` for more information.

there could be references to a glossary term (usually :term:`handler`), a Python module (usually :py:mod:`signal` or :mod:`signal`) and a section (usually :ref:`about-signals`).

The [any](http://www.sphinx-doc.org/en/master/usage/restructuredtext/roles.html#_blank) role also works together with the [intersphinx](http://www.sphinx-doc.org/en/master/usage/extensions/intersphinx.html#_blank) extension: when no local cross-reference is found, all object types of intersphinx inventories are also searched.

**Cross-referencing objects**

These roles are described with their respective domains:

* [Python](http://www.sphinx-doc.org/en/master/usage/restructuredtext/domains.html#python-roles)
* [C](http://www.sphinx-doc.org/en/master/usage/restructuredtext/domains.html#c-roles)
* [C++](http://www.sphinx-doc.org/en/master/usage/restructuredtext/domains.html#cpp-roles)
* [JavaScript](http://www.sphinx-doc.org/en/master/usage/restructuredtext/domains.html#js-roles)
* [ReST](http://www.sphinx-doc.org/en/master/usage/restructuredtext/domains.html#rst-roles)

**Cross-referencing arbitrary locations**

:ref:

To support cross-referencing to arbitrary locations in any document, the standard reST labels are used. For this to work label names must be unique throughout the entire documentation. There are two ways in which you can refer to labels:

* If you place a label directly before a section title, you can reference to it with :ref:`label-name`. For example:
* .. \_my-reference-label:
* Section to cross-reference
* --------------------------
* This is the text of the section.
* It refers to the section itself, see :ref:`my-reference-label`.

The :ref: role would then generate a link to the section, with the link title being “Section to cross-reference”. This works just as well when section and reference are in different source files.

Automatic labels also work with figures. For example:

.. \_my-figure:

.. figure:: whatever

Figure caption

In this case, a reference :ref:`my-figure` would insert a reference to the figure with link text “Figure caption”.

The same works for tables that are given an explicit caption using the [table](http://docutils.sourceforge.net/docs/ref/rst/directives.html#table) directive.

* Labels that aren’t placed before a section title can still be referenced, but you must give the link an explicit title, using this syntax: :ref:`Link title <label-name>`.

Note

Reference labels must start with an underscore. When referencing a label, the underscore must be omitted (see examples above).

Using [ref](http://www.sphinx-doc.org/en/master/usage/restructuredtext/roles.html#_blank) is advised over standard reStructuredText links to sections (like `Section title`\_) because it works across files, when section headings are changed, will raise warnings if incorrect, and works for all builders that support cross-references.

**Cross-referencing documents**

New in version 0.6.

There is also a way to directly link to documents:

:doc:

Link to the specified document; the document name can be specified in absolute or relative fashion. For example, if the reference :doc:`parrot` occurs in the document sketches/index, then the link refers to sketches/parrot. If the reference is :doc:`/people` or :doc:`../people`, the link refers to people.

If no explicit link text is given (like usual: :doc:`Monty Python members </people>`), the link caption will be the title of the given document.

**Referencing downloadable files**

New in version 0.6.

:download:

This role lets you link to files within your source tree that are not reST documents that can be viewed, but files that can be downloaded.

When you use this role, the referenced file is automatically marked for inclusion in the output when building (obviously, for HTML output only). All downloadable files are put into the \_downloads subdirectory of the output directory; duplicate filenames are handled.

An example:

See :download:`this example script <../example.py>`.

The given filename is usually relative to the directory the current source file is contained in, but if it absolute (starting with /), it is taken as relative to the top source directory.

The example.py file will be copied to the output directory, and a suitable link generated to it.

Not to show unavailable download links, you should wrap whole paragraphs that have this role:

.. only:: builder\_html

See :download:`this example script <../example.py>`.

**Cross-referencing figures by figure number**

New in version 1.3.

Changed in version 1.5: *numref* role can also refer sections. And *numref* allows *{name}* for the link text.

:numref:

Link to the specified figures, tables, code-blocks and sections; the standard reST labels are used. When you use this role, it will insert a reference to the figure with link text by its figure number like “Fig. 1.1”.

If an explicit link text is given (as usual: :numref:`Image of Sphinx (Fig. %s) <my-figure>`), the link caption will serve as title of the reference. As placeholders, *%s* and *{number}* get replaced by the figure number and *{name}* by the figure caption. If no explicit link text is given, the [numfig\_format](http://www.sphinx-doc.org/en/master/usage/configuration.html#confval-numfig_format) setting is used as fall-back default.

If [numfig](http://www.sphinx-doc.org/en/master/usage/configuration.html#confval-numfig) is False, figures are not numbered, so this role inserts not a reference but the label or the link text.

**Cross-referencing other items of interest**

The following roles do possibly create a cross-reference, but do not refer to objects:

:envvar:

An environment variable. Index entries are generated. Also generates a link to the matching [envvar](http://www.sphinx-doc.org/en/master/usage/restructuredtext/domains.html#_blank) directive, if it exists.

:token:

The name of a grammar token (used to create links between [productionlist](http://www.sphinx-doc.org/en/master/usage/restructuredtext/directives.html#_blank) directives).

:keyword:

The name of a keyword in Python. This creates a link to a reference label with that name, if it exists.

:option:

A command-line option to an executable program. This generates a link to a [option](http://www.sphinx-doc.org/en/master/usage/restructuredtext/domains.html#_blank) directive, if it exists.

The following role creates a cross-reference to a term in a [glossary](http://www.sphinx-doc.org/en/master/usage/restructuredtext/directives.html#glossary-directive):

:term:

Reference to a term in a glossary. A glossary is created using the glossary directive containing a definition list with terms and definitions. It does not have to be in the same file as the term markup, for example the Python docs have one global glossary in the glossary.rst file.

If you use a term that’s not explained in a glossary, you’ll get a warning during build.

**Math**

:math:

Role for inline math. Use like this:

Since Pythagoras, we know that :math:`a^2 + b^2 = c^2`.

:eq:

Same as math:numref.

**Other semantic markup**

The following roles don’t do anything special except formatting the text in a different style:

:abbr:

An abbreviation. If the role content contains a parenthesized explanation, it will be treated specially: it will be shown in a tool-tip in HTML, and output only once in LaTeX.

Example: :abbr:`LIFO (last-in, first-out)`.

New in version 0.6.

:command:

The name of an OS-level command, such as rm.

:dfn:

Mark the defining instance of a term in the text. (No index entries are generated.)

:file:

The name of a file or directory. Within the contents, you can use curly braces to indicate a “variable” part, for example:

... is installed in :file:`/usr/lib/python2.{x}/site-packages` ...

In the built documentation, the x will be displayed differently to indicate that it is to be replaced by the Python minor version.

:guilabel:

Labels presented as part of an interactive user interface should be marked using guilabel. This includes labels from text-based interfaces such as those created using curses or other text-based libraries. Any label used in the interface should be marked with this role, including button labels, window titles, field names, menu and menu selection names, and even values in selection lists.

Changed in version 1.0: An accelerator key for the GUI label can be included using an ampersand; this will be stripped and displayed underlined in the output (example: :guilabel:`&Cancel`). To include a literal ampersand, double it.

:kbd:

Mark a sequence of keystrokes. What form the key sequence takes may depend on platform- or application-specific conventions. When there are no relevant conventions, the names of modifier keys should be spelled out, to improve accessibility for new users and non-native speakers. For example, an *xemacs* key sequence may be marked like :kbd:`C-x C-f`, but without reference to a specific application or platform, the same sequence should be marked as :kbd:`Control-x Control-f`.

:mailheader:

The name of an RFC 822-style mail header. This markup does not imply that the header is being used in an email message, but can be used to refer to any header of the same “style.” This is also used for headers defined by the various MIME specifications. The header name should be entered in the same way it would normally be found in practice, with the camel-casing conventions being preferred where there is more than one common usage. For example: :mailheader:`Content-Type`.

:makevar:

The name of a **make** variable.

:manpage:

A reference to a Unix manual page including the section, e.g. :manpage:`ls(1)`. Creates a hyperlink to an external site rendering the manpage if [manpages\_url](http://www.sphinx-doc.org/en/master/usage/configuration.html#confval-manpages_url) is defined.

:menuselection:

Menu selections should be marked using the menuselection role. This is used to mark a complete sequence of menu selections, including selecting submenus and choosing a specific operation, or any subsequence of such a sequence. The names of individual selections should be separated by -->.

For example, to mark the selection “Start > Programs”, use this markup:

:menuselection:`Start --> Programs`

When including a selection that includes some trailing indicator, such as the ellipsis some operating systems use to indicate that the command opens a dialog, the indicator should be omitted from the selection name.

menuselection also supports ampersand accelerators just like [guilabel](http://www.sphinx-doc.org/en/master/usage/restructuredtext/roles.html#_blank).

:mimetype:

The name of a MIME type, or a component of a MIME type (the major or minor portion, taken alone).

:newsgroup:

The name of a Usenet newsgroup.

Todo

Is this not part of the standard domain?

:program:

The name of an executable program. This may differ from the file name for the executable for some platforms. In particular, the .exe (or other) extension should be omitted for Windows programs.

:regexp:

A regular expression. Quotes should not be included.

:samp:

A piece of literal text, such as code. Within the contents, you can use curly braces to indicate a “variable” part, as in [file](http://www.sphinx-doc.org/en/master/usage/restructuredtext/roles.html#_blank). For example, in :samp:`print 1+{variable}`, the part variable would be emphasized.

If you don’t need the “variable part” indication, use the standard ``code`` instead.

Changed in version 1.8: Allowed to escape curly braces with backslash

There is also an [index](http://www.sphinx-doc.org/en/master/usage/restructuredtext/directives.html#_blank) role to generate index entries.

The following roles generate external links:

:pep:

A reference to a Python Enhancement Proposal. This generates appropriate index entries. The text “PEP *number*” is generated; in the HTML output, this text is a hyperlink to an online copy of the specified PEP. You can link to a specific section by saying :pep:`number#anchor`.

:rfc:

A reference to an Internet Request for Comments. This generates appropriate index entries. The text “RFC *number*” is generated; in the HTML output, this text is a hyperlink to an online copy of the specified RFC. You can link to a specific section by saying :rfc:`number#anchor`.

Note that there are no special roles for including hyperlinks as you can use the standard reST markup for that purpose.

**Substitutions**

The documentation system provides three substitutions that are defined by default. They are set in the build configuration file.

|release|

Replaced by the project release the documentation refers to. This is meant to be the full version string including alpha/beta/release candidate tags, e.g. 2.5.2b3. Set by [release](http://www.sphinx-doc.org/en/master/usage/configuration.html#confval-release).

|version|

Replaced by the project version the documentation refers to. This is meant to consist only of the major and minor version parts, e.g. 2.5, even for version 2.5.1. Set by [version](http://www.sphinx-doc.org/en/master/usage/configuration.html#confval-version).

|today|

Replaced by either today’s date (the date on which the document is read), or the date set in the build configuration file. Normally has the format April 14, 2007. Set by [today\_fmt](http://www.sphinx-doc.org/en/master/usage/configuration.html#confval-today_fmt) and [today](http://www.sphinx-doc.org/en/master/usage/configuration.html#confval-today).