User_Manual

Release 1.0

Penny

Sep 16, 2025

CONTENTS:

Add your content using `reStructuredText` syntax. See the reStructuredText documentation for details.

CHAPTER

ONE

GETTING STARTED

1.1 Installing your doc directory

You may already have sphinx sphinx installed – you can check by doing:

```
python -c 'import sphinx'
```

If that fails grab the latest version of and install it with:

```
> sudo easy_install -U Sphinx
```

Now you are ready to build a template for your docs, using sphinx-quickstart:

```
> sphinx-quickstart
```

accepting most of the defaults. I choose "sampledoc" as the name of my project. cd into your new directory and check the contents:

```
home:~/tmp/sampledoc> ls
Makefile      _static         conf.py
build         _templates      index.rst
```

The index.rst is the master ReST for your project, but before adding anything, let's see if we can build some html:

```
make html
```

If you now point your browser to `build/html/index.html`, you should see a basic sphinx site.

1.1.1 Fetching the data

Now we will start to customize out docs. Grab a couple of files from the web site or git. You will need `getting_started.rst` and `images/basic_screenshot.png`. All of the files live in the "completed" version of this tutorial, but since this is a tutorial, we'll just grab them one at a time, so you can learn what needs to be changed where. Since we have more files to come, I'm going to grab the whole git directory and just copy the files I need over for now. First, I'll cd up back into the directory containing my project, check out the "finished" product from git, and then copy in just the files I need into my `sampledoc` directory:

```
home:~/tmp/sampledoc> pwd
/Users/jdhunter/tmp/sampledoc
home:~/tmp/sampledoc> cd ..
home:~/tmp> git clone https://github.com/matplotlib/sampledoc.git tutorial
Cloning into 'tutorial'...
remote: Counting objects: 87, done.
remote: Compressing objects: 100% (43/43), done.
remote: Total 87 (delta 45), reused 83 (delta 41)
Unpacking objects: 100% (87/87), done.
Checking connectivity... done
home:~/tmp> cp tutorial/getting_started.rst sampledoc/
home:~/tmp> cp tutorial/_static/images/basic_screenshot.png sampledoc/_static/
```

The last step is to modify `index.rst` to include the `getting_started.rst` file (be careful with the indentation, the "g" in "getting_started" should line up with the ':' in `:maxdepth`:

```
Contents:

.. toctree::
   :maxdepth: 2

   getting_started.rst
```

and then rebuild the docs:

```
cd sampledoc
make html
```

When you reload the page by refreshing your browser pointing to `build/html/index.html`, you should see a link to the "Getting Started" docs, and in there this page with the screenshot. *Voila!*

We can also use the image directive in `index.rst` to include to the screenshot above with:

```
.. image::
   images/basic_screenshot.png
```

Next we'll customize the look and feel of our site to give it a logo, some custom css, and update the navigation panels to look more like the sphinx <http://sphinx.pocoo.org/> site itself – see custom_look.

CHAPTER

TWO

SPHINX EXTENSIONS FOR EMBEDDED PLOTS, MATH AND MORE

Sphinx is written in python, and supports the ability to write custom extensions. We've written a few for the matplotlib documentation, some of which are part of matplotlib itself in the matplotlib.sphinxext module, some of which are included only in the sphinx doc directory, and there are other extensions written by other groups, eg numpy and ipython. We're collecting these in this tutorial and showing you how to install and use them for your own project. First let's grab the python extension files from the `sphinxext` directory from git (see *Fetching the data*), and install them in our `sampledoc` project `sphinxext` directory:

```
home:~/tmp/sampledoc> mkdir sphinxext
home:~/tmp/sampledoc> cp ../sampledoc_tut/sphinxext/*.py sphinxext/
home:~/tmp/sampledoc> ls sphinxext/
apigen.py  docscrape.py  docscrape_sphinx.py  numpydoc.py
```

In addition to the builtin matplotlib extensions for embedding pyplot plots and rendering math with matplotlib's native math engine, we also have extensions for syntax highlighting ipython sessions, making inhertiance diagrams, and more.

We need to inform sphinx of our new extensions in the `conf.py` file by adding the following. First we tell it where to find the extensions:

```
# If your extensions are in another directory, add it here. If the
# directory is relative to the documentation root, use
# os.path.abspath to make it absolute, like shown here.
sys.path.append(os.path.abspath('sphinxext'))
```

And then we tell it what extensions to load:

```
# Add any Sphinx extension module names here, as strings. They can be extensions
# coming with Sphinx (named 'sphinx.ext.*') or your custom ones.
extensions = ['matplotlib.sphinxext.only_directives',
              'matplotlib.sphinxext.plot_directive',
              'IPython.sphinxext.ipython_directive',
              'IPython.sphinxext.ipython_console_highlighting',
              'sphinx.ext.mathjax',
              'sphinx.ext.autodoc',
              'sphinx.ext.doctest',
              'sphinx.ext.inheritance_diagram',
              'numpydoc']
```

Now let's look at some of these in action. You can see the literal source for this file at *This file*.

2.1 ipython sessions

Michael Droettboom contributed a sphinx extension which does pygments syntax highlighting on ipython sessions. Just use ipython as the language in the `sourcecode` directive:

```
.. sourcecode:: ipython

   In [69]: lines = plot([1,2,3])

   In [70]: setp(lines)
     alpha: float
     animated: [True | False]
     antialiased or aa: [True | False]
     ...snip
```

and you will get the syntax highlighted output below.

```
In [69]: lines = plot([1,2,3])

In [70]: setp(lines)
  alpha: float
  animated: [True | False]
  antialiased or aa: [True | False]
  ...snip
```

This support is included in this template, but will also be included in a future version of Pygments by default.

2.2 Using math

In sphinx you can include inline math $x \leftarrow y\ x \forall y\ x - y$ or display math

$$W^{3\beta}_{\delta_1 \rho_1 \sigma_2} = U^{3\beta}_{\delta_1 \rho_1} + \frac{1}{8\pi 2} \int_{\alpha_2}^{\alpha_2} d\alpha_2' \left[\frac{U^{2\beta}_{\delta_1 \rho_1} - \alpha_2' U^{1\beta}_{\rho_1 \sigma_2}}{U^{0\beta}_{\rho_1 \sigma_2}} \right]$$

To include math in your document, just use the math directive; here is a simpler equation:

```
.. math::

  W^{3\beta}_{\delta_1 \rho_1 \sigma_2} \approx U^{3\beta}_{\delta_1 \rho_1}
```

which is rendered as

$$W^{3\beta}_{\delta_1 \rho_1 \sigma_2} \approx U^{3\beta}_{\delta_1 \rho_1}$$

Recent versions of Sphinx include built-in support for math. There are three flavors:

- sphinx.ext.imgmath: uses dvipng to render the equation
- sphinx.ext.mathjax: renders the math in the browser using Javascript
- sphinx.ext.jsmath: it's an older code, but it checks out

Additionally, matplotlib has its own math support:

- matplotlib.sphinxext.mathmpl

See the matplotlib mathtext guide for lots more information on writing mathematical expressions in matplotlib.

2.3 Inserting matplotlib plots

Inserting automatically-generated plots is easy. Simply put the script to generate the plot in the `pyplots` directory, and refer to it using the `plot` directive. First make a `pyplots` directory at the top level of your project (next to :`conf.py`) and copy the `ellipses.py`` file into it:

```
home:~/tmp/sampledoc> mkdir pyplots
home:~/tmp/sampledoc> cp ../sampledoc_tut/pyplots/ellipses.py pyplots/
```

You can refer to this file in your sphinx documentation; by default it will just inline the plot with links to the source and PF and high resolution PNGS. To also include the source code for the plot in the document, pass the `include-source` parameter:

```
.. plot:: pyplots/ellipses.py
   :include-source:
```

In the HTML version of the document, the plot includes links to the original source code, a high-resolution PNG and a PDF. In the PDF version of the document, the plot is included as a scalable PDF.

You can also inline code for plots directly, and the code will be executed at documentation build time and the figure inserted into your docs; the following code:

```
.. plot::

   import matplotlib.pyplot as plt
   import numpy as np
   x = np.random.randn(1000)
   plt.hist( x, 20)
   plt.grid()
   plt.title(r'Normal: $\mu=%.2f, \sigma=%.2f$'%(x.mean(), x.std()))
   plt.show()
```

produces this output:

See the matplotlib pyplot tutorial and the gallery for lots of examples of matplotlib plots.

2.4 Inheritance diagrams

Inheritance diagrams can be inserted directly into the document by providing a list of class or module names to the `inheritance-diagram` directive.

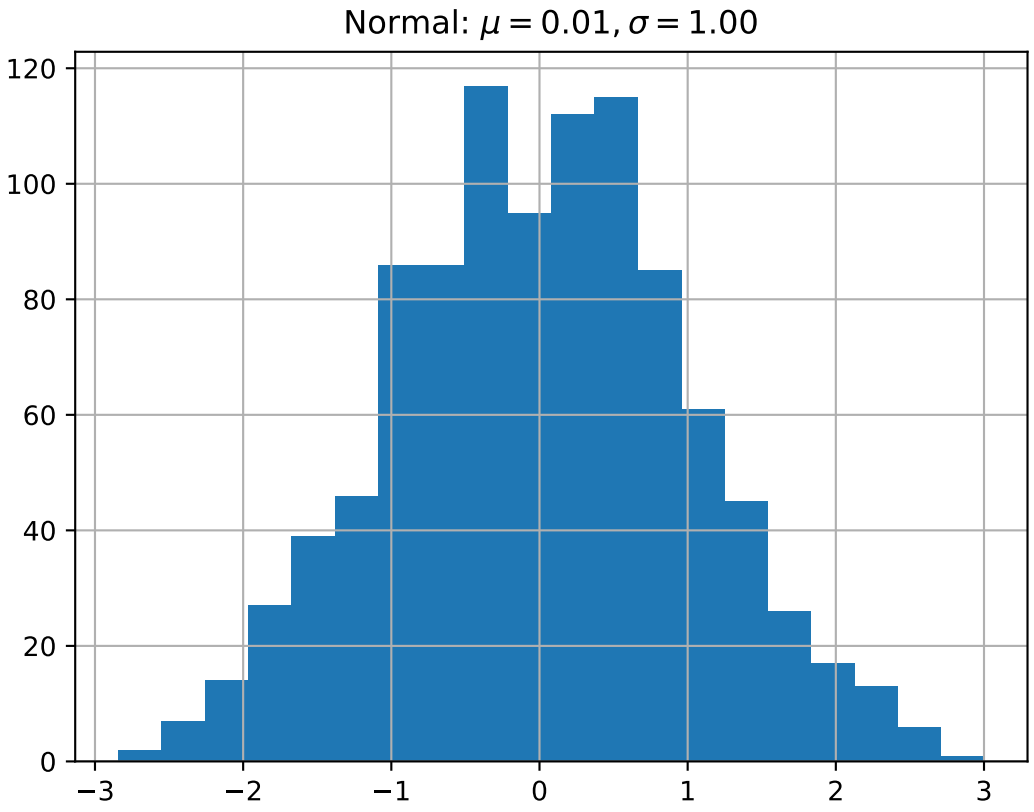
For example:

```
.. inheritance-diagram:: codecs
```

See the *ipython_directive* for a tutorial on embedding stateful, matplotlib aware ipython sessions into your rest docs with multiline and doctest support.

2.5 This file

```
```


Normal: $\mu = 0.01, \sigma = 1.00$
120
100
80
60
40
20
0
−3
−2
−1
0
1
2
3

CHAPTER

THREE

MARKDOWN HEADING 1

https://github.com/tchapi/markdown-cheatsheet

Markup : # Heading 1 #

-OR-

Markup : ========== (below H1 text)

3.1 Heading 2

Markup : ## Heading 2 ##

-OR-

Markup: ———— (below H2 text)

3.1.1 Heading 3

Markup : ### Heading 3 ###

Heading 4

Markup : #### Heading 4 ####

Common text

```
Markup :  Common text
```

Emphasized text

```
Markup :  _Emphasized text_ or *Emphasized text*
```

~~Strikethrough text~~

```
Markup :  ~~Strikethrough text~~
```

Strong text

```
Markup :  __Strong text__ or **Strong text**
```

Strong emphasized text

```
Markup :  ___Strong emphasized text___ or ***Strong emphasized text***
```

Named Link and http://www.google.fr/ or http://example.com/

```
Markup :  [Named Link](http://www.google.fr/ "Named link title") and http://www.google.
↪fr/ or <http://example.com/>
```

Table, like this one :

First Header	Second Header
Content Cell	Content Cell
Content Cell	Content Cell

```
First Header  | Second Header
------------- | -------------
Content Cell  | Content Cell
Content Cell  | Content Cell
```

`code()`

```
Markup :  `code()`
```

```
    var specificLanguage_code =
    {
        "data": {
            "lookedUpPlatform": 1,
            "query": "Kasabian+Test+Transmission",
            "lookedUpItem": {
                "name": "Test Transmission",
                "artist": "Kasabian",
                "album": "Kasabian",
                "picture": null,
                "link": "http://open.spotify.com/track/5jhJur5n4fasblLSCOcrTp"
            }
        }
    }
```

```
Markup : ```javascript
         ```
```

- Bullet list
  - Nested bullet
    - Sub-nested bullet etc
- Bullet list item 2

```
Markup : * Bullet list
           * Nested bullet
               * Sub-nested bullet etc
         * Bullet list item 2
```
```

1. A numbered list
    1. A nested numbered list
    2. Which is numbered
2. Which is numbered

```
Markup : 1. A numbered list
             1. A nested numbered list
             2. Which is numbered
         2. Which is numbered
```

- [ ] An uncompleted task
- [x] A completed task

```
Markup : - [ ] An uncompleted task
         - [x] A completed task
```

> Blockquote
>
> > Nested blockquote

```
Markup :  > Blockquote
          >> Nested Blockquote
```

Horizontal line :

```
Markup :  - - - -
```

Image with alt :


```
Markup : ![picture alt](http://via.placeholder.com/200x150 "Title is optional")
```

Foldable text:

```
Markup : <details>
           <summary>Title 1</summary>
           <p>Content 1 Content 1 Content 1 Content 1 Content 1</p>
         </details>
```

```
<h3>HTML</h3>
<p> Some HTML code here </p>
```

Hotkey:

F

F

```
Markup : <kbd>F</kbd>
```

Hotkey list:

Key	Symbol
Option	
Control	
Command	
Shift	
Caps Lock	
Tab	
Esc	
Power	
Return	
Delete	
Up	↑
Down	↓
Left	←
Right	→

Emoji:

:exclamation: Use emoji icons to enhance text. :+1: Look up emoji codes at emoji-cheat-sheet.com

```
Markup : Code appears between colons :EMOJICODE:
```

CHAPTER

FOUR

INDICES AND TABLES

- genindex
- modindex
- search