

# Sphinx API Documentation: Step 0 towards Reusable Code

Yubo “Paul” Yang, THW-IL, 2017/10/18



**CODING HORROR**

programming and human factors

Google Custom Search



Understand and deliver High C/C++ Code Quality. Quality Gates, Technical Debt, Code Query Language. Easy to set up and to use.

23 Jan 2007

## If It Isn't Documented, It Doesn't Exist

Nicholas Zakas enumerates [the number one reason why good JavaScript libraries fail](#):

**Lack of documentation.** No matter how wonderful your library is and how intelligent its design, if you're the only one who understands it, it doesn't do any good. Documentation means not just autogenerated API references, but also *annotated* examples and in-depth tutorials. You need all three to make sure your library can be easily adopted.

**Act I: It is all so simple**

# Sphinx API Documentation Requires docstring only

“A docstring is a string literal that occurs as the first statement in a module, function, class, or method definition.” – PEP 257, Python.org

```
6 def isosurf(ax,vol,level_frac=0.25):
7     """draw iso surface of volumetric data on matplotlib axis at given level
8
9     Example usage:
10     from mpl_toolkits.mplot3d import Axes3D # enable 3D projection
11     vol = np.random.randn(10,10,10)
12     fig = plt.figure()
13     ax = fig.add_subplot(1,1,1,projection='3d')
14     isosurf(ax,vol)
15     plt.show()
16
17     Args:
18     ax (plt.Axes3D): ax = fig.add_subplot(1,1,1,projection="3d")
19     vol (np.array): 3D volumetric data having shape (nx,ny,nz)
20     level_frac (float): 0.0->1.0, isosurface value as a fraction between min and max
21
22     Returns:
23     Poly3DCollection: mesh
24
25     Effect:
26     draw on ax """
27     from skimage import measure
28     from mpl_toolkits.mplot3d.art3d import Poly3DCollection
29     nx,ny,nz = vol.shape
30     lmin,lmax = vol.min(),vol.max()
31     --
```

Sphinx

## qharv.inspect.volumetric module

qharv.inspect.volumetric.axes\_func\_on\_grid3d(axes, func, grid\_shape) [\[source\]](#)  
put a function define in axes units on a 3D grid :param axes: dtype=float, shape=(3,3); 3D lattice vectors in row major (i.e. a1 = axes[0]) :type axes: np.array :param func: 3D function defined on the unit cube :type func: RegularGridInterpolator :param grid\_shape: dtype=int, shape=(3,); shape of real space grid :type grid\_shape: np.array

**Returns:** dtype=float, shape=grid\_shape; volumetric data  
**Return type:** grid (np.array)

qharv.inspect.volumetric.isosurf(ax, vol, level\_frac=0.25) [\[source\]](#)  
draw iso surface of volumetric data on matplotlib axis at given level

Example usage:

```
from mpl_toolkits.mplot3d import Axes3D # enable 3D projection vol =
np.random.randn(10,10,10) fig = plt.figure() ax =
fig.add_subplot(1,1,1,projection='3d') isosurf(ax,vol) plt.show()
```

**Parameters:**

- **ax** (*plt.Axes3D*) – ax = fig.add\_subplot(1,1,1,projection="3d")
- **vol** (*np.array*) – 3D volumetric data having shape (nx,ny,nz)
- **level\_frac** (*float*) – 0.0->1.0, isosurface value as a fraction between min and max

**Returns:** mesh  
**Return type:** Poly3DCollection

**Effect:**

draw on ax

## Sphinx API Documentation Setup Takes **ONE** Command

```
sphinx-apidoc -A "Paul" -F -o docs src/
```

- sphinx-apidoc is a command available in the sphinx package, which can be installed without admin privileges using `pip install --user sphinx`.
- -A specifies the author name
- -F triggers a full setup
- -o specifies the documentation directory
- src/ is the folder that contains your Python package

### **Gotcha:**

sphinx-apidoc cannot build documentation if it cannot **import your module!**

**Ref:** jlk's blog post on [raxcloud](#)

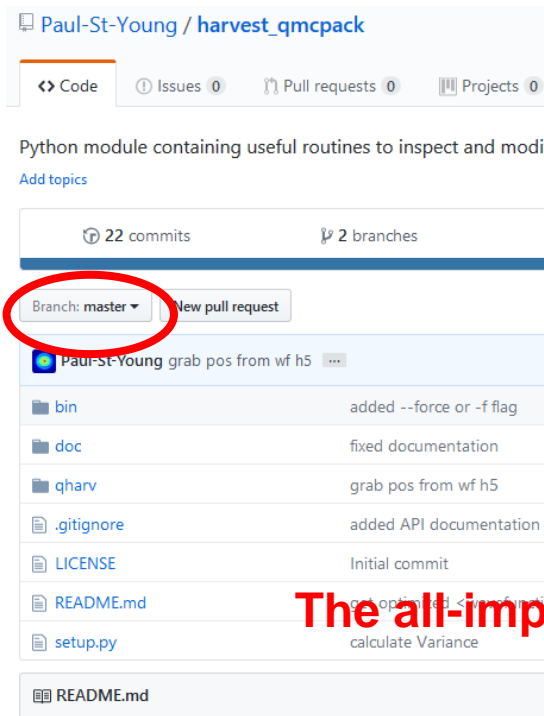
Sphinx API Documentation Generation also takes **ONE** Command

```
make html
```

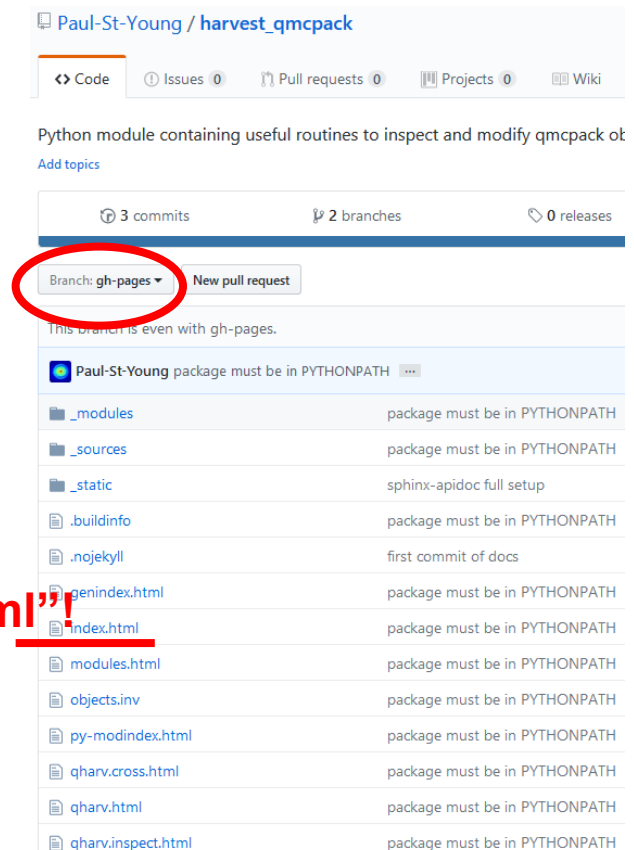
- pdf version can be generated using ``make latexpdf``. You will need the latexmk package, which should be easily obtainable via ``apt-get/dnf install latexmk``.
- Read the Makefile to see what other options are available.

# Publish Documentation on GitHub Pages

**Goal** Make gh-pages branch to hold the output of `make html`.



**The all-important “index.html”!**



## Gotcha:

1. sphinx-apidoc cannot build documentation if it cannot **import your module!**
2. There are more than one way to **create an empty gh-pages branch** in your repository (see Ref [1] vs. Ref [3]). No need to be stubborn.

Ref [1] Luca Sbardella blog [post](#) on 2010/02/09

Ref [2] Ryan Dale's sphinxdoc-test GitHub [repository](#)

Ref [3] jlk's blog post on [raxcloud](#)

Conclusions: Sphinx is as easy as 1,2,3!



1. Write docstring for your functions
2. One command to setup: `sphinx-apidoc -A "Paul" -o docs src/`
3. One command to run: `make html`

**Ready to publish!**

**Act II:** *What* just happened?



`sphinx-apidoc` converts .py to .rst

```
sphinx-apidoc -A "Paul" -F -o docs src/
```

- docs folder contains one .rst file for each .py file in your src/ (with one extra index.rst)
- -F generates conf.py and Makefile, you may wish to add a few extensions in conf.py

default

```
extensions = ['sphinx.ext.autodoc',  
             'sphinx.ext.todo',  
             'sphinx.ext.viewcode']
```

customized

```
extensions = ['sphinx.ext.autodoc',  
             'sphinx.ext.napoleon',  
             'sphinx.ext.coverage',  
             'sphinx.ext.mathjax',  
             'sphinx.ext.viewcode',  
             'sphinx.ext.githubpages']
```

Google style docstring



pages

- For more custom setup, run `sphinx-quickstart` for *the quiz*.

# For Our Beloved Python 2

- `conf.py` uses Python 3 style by default!




- To change style, edit `conf.py`:
  1. change ``html_theme = 'alabaster'`` to ``html_theme = 'classic'``
  2. remove unnecessary cram:

```
html_sidebars = {  
    '**': [  
        #'about.html',  
        'searchbox.html',  
        'globaltoc.html',  
        #'navigation.html',  
        'relations.html',  
        #'donate.html',  
    ]  
}
```

`make html` converts .rst to .html

## make html

- `make html` fills docs/\_build/html folder with html files
- Try `firefox docs/\_build/html/index.html`  
A terminal window showing the command `ls` and its output. The output lists several files: genindex.html, \_modules, objects.inv, search.html, \_sources, index.html, nexus, obj.html, py-modindex.html, searchindex.js, and \_static. The files are displayed in two columns.

```
[yyang173@localhost html]$ ls
genindex.html  _modules      objects.inv    search.html    _sources
index.html     nexus obj.html   py-modindex.html  searchindex.js  _static
```
- The 'githubpages' extension will add an empty '**.nojekyll**' file to docs/\_build/html/
- `touch docs/\_build/html/.nojekyll` should also work, but is not automatic

# **Act III: Beginning of the End**

# API Documentation is Step 0 towards Reusable Code



## CODING HORROR

programming and human factors

Google Custom Search



Understand and deliver High C/C++ Code Quality. Quality Gates, Technical Debt, Code Query Language. Easy to set up and to use.

23 Jan 2007

## If It Isn't Documented, It Doesn't Exist

Nicholas Zakas enumerates [the number one reason why good JavaScript libraries fail](#):

**Lack of documentation.** No matter how wonderful your library is and how intelligent its design, if you're the only one who understands it, it doesn't do any good. Documentation means not just autogenerated API references, but also *annotated* examples and in-depth tutorials. You need all three to make sure your library can be easily adopted.

# What can Withstand the Revenges of Time?

- API Documentation
- Tests
- Examples

build passing build passing codecov 96% circleci passing python 2.7 python 3.5 pypi package 0.19.0 DOI 10.5281/zenodo.49911

## scikit-learn

scikit-learn is a Python module for machine learning

The project was started in 2007 by David Cournapeau and others who have contributed. See the [AUTHORS.rst](#) file for more details.

It is currently maintained by a team of volunteers.

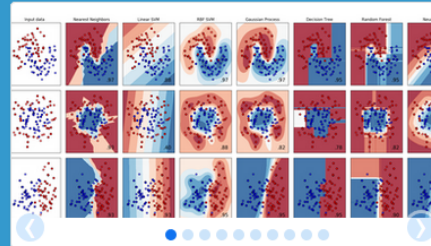
Website: <http://scikit-learn.org>



Home Installation Documentation Examples

Google Custom Search

Search



## scikit-learn

Machine Learning in Python

- Simple and efficient tools for data mining and data analysis
- Accessible to everybody, and reusable in various contexts
- Built on NumPy, SciPy, and matplotlib
- Open source, commercially usable - BSD license

### Classification

Identifying to which category an object belongs to.

**Applications:** Spam detection, Image recognition.

**Algorithms:** SVM, nearest neighbors, random forest, ...

— Examples

### Regression

Predicting a continuous-valued attribute associated with an object.

**Applications:** Drug response, Stock prices.

**Algorithms:** SVR, ridge regression, Lasso, ...

— Examples

### Clustering

Automatic grouping of similar objects into sets.

**Applications:** Customer segmentation, Grouping experiment outcomes

**Algorithms:** k-Means, spectral clustering, mean-shift, ...

— Examples

### Dimensionality reduction

Reducing the number of random variables to consider.

**Applications:** Visualization, Increased efficiency

**Algorithms:** PCA, feature selection, non-negative matrix factorization, ...

— Examples

### Model selection

Comparing, validating and choosing parameters and models.

**Goal:** Improved accuracy via parameter tuning

**Modules:** grid search, cross-validation, metrics, ...

— Examples

### Preprocessing

Feature extraction and normalization.

**Application:** Transforming input data such as text for use with machine learning algorithms.

**Modules:** preprocessing, feature extraction, ...

— Examples

**To be continued ...**