## Summary of steps for creating documentation with Sphinx

Step 1 Documenting your code with docstrings (Google vs. Numpy)

Step 2 Building the directory tree for your package

Step 3 Creating a virtual environment for Sphinx

Step 4 Activating your virtual environment

Step 5 (Optional) Converting .doc, .docx, .html to .rst

Step 6 Starting with sphinx-quickstart (making your documentation)

Step 7 Ensuring sphinx-quickstart ran correctly

Step 8 Modifying conf.py

Step 9 Creating HTML documentation with make html

Step 10 Ensuring make html ran correctly (viewing index.html)

Step 11 Adding additional HTML pages to your documentation

**Creating and Documenting Python Packages**

Note: Guide here assumes you have created a virtual environment.

## Create requirements.txt documenting installed packages in environment

(**Step 1**) Open Cygwin (or Korn) and activate your virtual environment

* cmd
* Path\to\environment\Scripts\activate.bat

(e.g., C:\Users\username\Python\envs\ENVNAME\Scripts\activate.bat

* pip freeze > requirements.txt

## Create documentation with Sphinx

(**Step 1**) When writing your python code, document each function using the Google or NumPy style of docstrings ([https://sphinxcontrib-napoleon.readthedocs.io/en/latest/#google-vs-numpy](https://sphinxcontrib-napoleon.readthedocs.io/en/latest/" \l "google-vs-numpy))

(**Step 2**) Once you are done developing your code, put your package together. A proposed directory tree follows (not all folders may be relevant for every project)

* Main folder named after the project (e.g., projectname\_master)
* Within in the main folder:
  + docs/ (documentation saved here)
  + projectname/ (project’s .py code here)
  + tests/ (files used for testing)
  + data/ (files code needs to ingest or produces—helpful for testing)
  + bin/ (holds executables if your project has any)
* Ensure all the files associated with your project are saved to the right subfolders



Figure 1. Sample directory tree for a project named helloworld

(**Step 3**) Once you are at this point, you need to have an environment with Sphinx installed. You can create a new environment and install Sphinx there (using [conda create](https://docs.conda.io/projects/conda/en/latest/user-guide/tasks/manage-environments.html" \l "creating-an-environment-with-commands)).

Either way, for the following steps, make sure you are working in an activated virtual environment with Sphinx installed.

In my case, I created one central environment with all the necessary packages to create documentation with Sphinx.

For reference, here’s the requirement.txt for my documentation virtual environment:

|  |  |  |
| --- | --- | --- |
| alabaster==0.7.12  asn1crypto==0.24.0  Babel==2.7.0  certifi==2019.3.9  cffi==1.12.3  chardet==3.0.4  Click==7.0  commonmark==0.9.0  cryptography==2.7  docutils==0.14  future==0.17.1  idna==2.8  imagesize==1.1.0  Jinja2==2.10.1  livereload==2.6.1 | m2r==0.2.1  Markdown==2.6.11  MarkupSafe==1.1.1  mistune==0.8.4  mkdocs==1.0.4  packaging==19.0  pycparser==2.19  Pygments==2.4.2  pyOpenSSL==19.0.0  pyparsing==2.4.0  PySocks==1.7.0  pytz==2019.1  PyYAML==5.1.1  recommonmark==0.5.0  requests==2.22.0 | six==1.12.0  snowballstemmer==1.2.1  Sphinx==2.1.0  sphinx-rtd-theme==0.4.3  sphinxcontrib-applehelp==1.0.1  sphinxcontrib-devhelp==1.0.1  sphinxcontrib-htmlhelp==1.0.2  sphinxcontrib-jsmath==1.0.1  sphinxcontrib-qthelp==1.0.2  sphinxcontrib-serializinghtml==1.1.1  tornado==6.0.2  urllib3==1.24.3  win-inet-pton==1.1.0  wincertstore==0.2 |

(**Step 4**) If not already active, activate your environment with Sphinx

(**Step 5a - Optional**) If you have Microsoft Word files (doc, docx) that you want to include in your documentation see: <https://peintinger.com/?p=365>

(**Step 5b – Optional**) If you have an HTML file that you want to include in your documentation you will need to convert it to .rst. pandocs can be used to do this. In your command shell, navigate to where the .html is located. Then execute the following command

**pandoc -s -f html -t rst myfile.html -o myfile.rst**

-s tells pandoc to produce a standalone document

-t tells pandoc to produce reStructuredText output

-f tells pandoc the input format

-o tells pandoc where to save the output (will create the file during execution)

(**Step 6**) In the subsequent steps, for the purpose of this tutorial, I will be assuming

projectname = *formatter*

the package name will be *formatter\_master*

and will be using the directory structure:

**|-doc**

**|**

**|-restart\_formatter**

**|**

**|-test**

**|**

**|-README.md**

**|**

**|-requirements.txt**

**|**

**|-tutorial.html**

Figure 2. Directory tree used in this tutorial to produce documentation with Sphinx

In the command shell, cd to the folder you want to save your documentation in (e.g., doc/)

**cd C:\Users\username\Documents\python\_codes\docs**

then enter the following command to run Sphinx

**sphinx-quickstart**

When asked “ *Separate source and build directories (y/n) [n]*: ” Enter **y .**

Then input the project name, your name, the project version, and project language when prompted

(FYI: usual convention for project version method X.Y.Z, which generally corresponds to major.minor.patch)

(**Step 7**) Look in your docs/ folder (or wherever you executed sphinx-quickstart from) and verify build/, soruce/, *make.bat*, and *Makefile* were created.

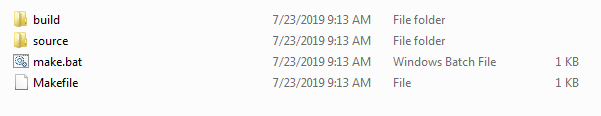


Figure 3. Output from running sphinx-start located in /docs folder

(**Step 8**) Open source/ and find the file *conf.py*. Open conf.py in a text editor (e.g., notepad++) and make the following modifications to the file:

* Uncomment (that is, delete the hashtag, *#*) from the following lines

# import os

# import sys

# sys.path.insert(0, os.path.abspath('.'))

* Underneath sys.path.insert(0, os.path.abspath('.')) add the following line

autodoc\_mock\_imports = ['pandas', 'numpy', 'pathlib']

Replace 'pandas', 'numpy', 'pathlib' with whatever packages your code actually uses (check out your requirements.txt if you need)

* We also need to specify the path to the module that we want to generate documentation for (the following change assumes you are using the directory tree structure discussed in the previous steps, see *Figure 2*).

Change sys.path.insert(0, os.path.abspath('.')) to sys.path.insert(0, os.path.abspath('../../'))

* Find the line html\_theme = 'alabaster' and change to html\_theme = 'sphinx\_rtd\_theme'
* Underneath html\_theme = 'sphinx\_rtd\_theme' copy the following text block

# Add on\_rtd for readthedocs.org

on\_rtd = os.environ.get('READTHEDOCS', None) == 'True'

if not on\_rtd: # only import and set the theme if we're building docs locally

import sphinx\_rtd\_theme

html\_theme = 'sphinx\_rtd\_theme'

html\_theme\_path = [sphinx\_rtd\_theme.get\_html\_theme\_path()]

* Find the line

extensions = [

]

and replace with

extensions = ['sphinx.ext.autodoc', 'sphinx.ext.coverage', 'sphinx.ext.napoleon', 'm2r']

* Below the extensions add source\_suffix = ['.rst', '.md']
* Save conf.py and close it.

(**Step 9**) Return to your command shell and run to build your documentation HTML file

**make html**

(**Step 10**) Verify *index.html* was created (located in docs/build/html/index.html)

(**Step 11**) Start adding files to your documentation (this step will vary depending on what you want to include). For the example provided here, it is assumed you have 3 python files: *db2pd.py*, *dbsummary.py*, and *gdx2db.py*. These files are located in formatter\_master/formatter/ and your documentation is located in ormatter\_master/docs/source and formatter\_master/docs/build.

* Open a text editor and create two files: *genindex.rst* and *py-modindex.rst*
* Copy into genindex:

Index

=====

* Copy into py-modindex:

Module Index

=====

* Save both files in docs/source
* Also in /source create a file called *api.rst* (this will house the API documentation for your python scripts)
* Add the following text block to api.rst

API documentation

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

.. automodule:: formatter.db2pd

:members:

.. automodule:: formatter.dbsummary

:members:

.. automodule:: formatter.gdx2db

:members:

* Include the three .rst files in the table of contents tree by opening and editing docs/source/index.rst to match the below (Note! the indentation is important – 3 spaces under..toctree:: header):

Welcome to NAME OF YOUR PROJECT's documentation!

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

.. toctree::

:maxdepth: 2

:caption: Contents:

api.rst

.. toctree::

:caption: Indices:

:hidden:

genindex

py-modindex

* Run the **make html** command again in the shell.
* Go back to docs/build/html – you should see *api.html* there now. If you open *index.html* you can see what you have created.
* Create a README file to introduce users to your package. In a text editor create a file called readme.rst and save it in docs/source. Copy the following text block into readme.rst:

-----------

README

-----------

.. mdinclude:: ../../README.md

* Create README.md in formatter\_master/. An idea of how to create a Markdown document is below (you can also visit <https://www.markdownguide.org/basic-syntax>). Modify the text in gray below to fit your project:

!!! Sample README Template Only – You’ll need to modify for your project !!!

<img src="./\_static/dbicon.png" alt="drawing" width="90"/> Package Name

## Overview

Description goes here.

## Getting Started

.. tip::

The inforamtion here is a general reference. A tutorial for using the code can be found in the test folder (link below)

accompanying this package.

You can link to the Notebook tutorial here: [tutorial.ipynb](../../../test/ "tutorial notebook"). The tutorial notebook can be opened in a Jupyter notebook. Or you can follow the HTML version of the code

provided in this documentation [(link)](../html/tutorial.html "tutorial.html").

## Prerequisites

To make sure you have all the requirements to run the scripts, it is recommended you set up a Python virtual environment. The full list of

packages installed in the virtual environment used to develop this package is in [requirements.txt](../../../requirements.txt "requirements").

Dependencies can be installed using

```

pip install -r requirements.txt

```

## Testing setup

You can test that everything is set up correctly by following the instructions in the [User's Tutorial](../html/tutorial.html "tutorial.html")

If you choose to run the tutorial in Jupyter Notebook you can find the \*.ipynb\* in ``/doc/test``. This directory also has the

necessary input files to run the notebook from this folder.

Running the tutorial in Jupyter is straightforward: <br>

Launch Jupyter, navigate to the ``/test`` folder, and open ``tutorial.ipynb`` in Jupyter. You should then be able to walk through the steps in the tutorial

by running the code in each code block.

<img src="./\_static/datacowgirl.png" alt="drawing" width="200"/> Happy data wrangling!

## POC for additional questions or to report problems

\* \*\*Your Name\*\*

* Add readme.rst to docs/source/index.rst

Welcome to NAME OF YOUR PROJECT's documentation!

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

.. toctree::

:maxdepth: 3

:caption: Contents:

readme.rst

api.rst

.. toctree::

:caption: Indices:

:hidden:

genindex

py-modindex

* To add more pages to your documentation, continue to add more \*.rst files to index.rst
* When you have finished adding all your \*.rst files to index.rst, go back to your command shell and run the *make html* command a final time to compile everything.

**make html**

* That’s it. Open up build/html/index.html to see what you have made!