ReadTheDocs and Sphinx

[Readthedocs.org](https://readthedocs.org/) is a website that can be used to host documentation for python (and other languate) software packages. It uses a python package called [sphinx](https://pypi.org/project/Sphinx/) to automatically build in examples and documentation from your python package.

## What is included here

In this file, I will walk you through a minimum example of how to set up a documentation website that is formatted based on one theme ([chemper](http://chemper.readthedocs.io/en/latest/)). The read the docs website has extensive documentation and there are A TON of different themes out there. This is intended to help you understand how to get started.

## Disclaimer

Below is a step by step guide, however, there is a STEEP learning curve on Read the Docs and its not particularly intuitive. In this tutorial, we assume you already have a working python package that is installable (see docs on setup files). This is intended to be a guide that shorter and easier to follow than the official Getting Started Guide, but you will likely need to look at outside resources.

## RST files

Read the docs uses rst (reStructuredText) files. These are similar to MarkDown files (such as those used for GitHub README files and jupyter notebooks), in that they have formatting rules that are interpretted to change the format of the text. However, the formatting is significantly different from MarkDown. This file will not include instruction for this format, but might highlight some examples. Below is a list of links our group (Caitlin) has found useful for how to format these files.

* <http://openalea.gforge.inria.fr/doc/openalea/doc/_build/html/source/sphinx/rest_syntax.html>
* <http://docutils.sourceforge.net/docs/user/rst/quickref.html>
* <https://thomas-cokelaer.info/tutorials/sphinx/rest_syntax.html>
* <https://github.com/ralsina/rst-cheatsheet/blob/master/rst-cheatsheet.rst>

As with all languages it is usually fastest to just google what you want to do and follow the relevant discussion board answers

# Building Doc Files Locally

We will start on your computer, get some initial files set up before linking the website to your github repository.

## Download Sphinx and run quickstart

* Install Sphinx

|  |
| --- |
| pip install sphinx sphinx-autobuild sphinx\_rtd\_theme nbsphinx |

* Make a docs directory to store files
* From inside the docs folder call sphinx-quickstart. Note that there are a ton of questions with this call, most of them we use the default, but I will put the full list below **bolding** the ones where I did not chose the default.
  + A side note, the last question is “Create Windows command file? (y/n) [y]: n” I was taught at MolSSI to say “n” to this and I don’t know anyone on a windows computer to have used this program. You may need these files if you’re going to check your build on a windows machine, I don’t know.
  + When done with this call you’ll have a source/ and build/ directory in the docs directory, DO NOT add the build/ directory to your github, it is only there so you can see your output locally.

|  |
| --- |
| (chemper\_both) gitHub/shapes2 (docs)$ sphinx-quickstart  Welcome to the Sphinx 1.7.6 quickstart utility.  Please enter values for the following settings (just press Enter to  accept a default value, if one is given in brackets).  Selected root path: .  You have two options for placing the build directory for Sphinx output.  Either, you use a directory "\_build" within the root path, or you separate  "source" and "build" directories within the root path.  **> Separate source and build directories (y/n) [n]: y**  Inside the root directory, two more directories will be created; "\_templates"  for custom HTML templates and "\_static" for custom stylesheets and other static  files. You can enter another prefix (such as ".") to replace the underscore.  > Name prefix for templates and static dir [\_]:  The project name will occur in several places in the built documentation.  **> Project name: shapes**  **> Author name(s): Caitlin C. Bannan**  **> Project release []: 0.1**  If the documents are to be written in a language other than English,  you can select a language here by its language code. Sphinx will then  translate text that it generates into that language.  For a list of supported codes, see  http://sphinx-doc.org/config.html#confval-language.  > Project language [en]:  The file name suffix for source files. Commonly, this is either ".txt"  or ".rst". Only files with this suffix are considered documents.  > Source file suffix [.rst]:  One document is special in that it is considered the top node of the  "contents tree", that is, it is the root of the hierarchical structure  of the documents. Normally, this is "index", but if your "index"  document is a custom template, you can also set this to another filename.  > Name of your master document (without suffix) [index]:  Sphinx can also add configuration for epub output:  > Do you want to use the epub builder (y/n) [n]:  Indicate which of the following Sphinx extensions should be enabled:  **> autodoc: automatically insert docstrings from modules (y/n) [n]: y**  > doctest: automatically test code snippets in doctest blocks (y/n) [n]:  > intersphinx: link between Sphinx documentation of different projects (y/n) [n]:  > todo: write "todo" entries that can be shown or hidden on build (y/n) [n]:  > coverage: checks for documentation coverage (y/n) [n]:  > imgmath: include math, rendered as PNG or SVG images (y/n) [n]:  **> mathjax: include math, rendered in the browser by MathJax (y/n) [n]: y**  > ifconfig: conditional inclusion of content based on config values (y/n) [n]:  **> viewcode: include links to the source code of documented Python objects (y/n) [n]: y**  > githubpages: create .nojekyll file to publish the document on GitHub pages (y/n) [n]:  A Makefile and a Windows command file can be generated for you so that you  only have to run e.g. make html instead of invoking sphinx-build  directly.  > Create Makefile? (y/n) [y]:  **> Create Windows command file? (y/n) [y]: n**  Creating file ./source/conf.py.  Creating file ./source/index.rst.  Creating file ./Makefile.  Finished: An initial directory structure has been created.  You should now populate your master file ./source/index.rst and create other documentation  source files. Use the Makefile to build the docs, like so:  make builder  where "builder" is one of the supported builders, e.g. html, latex or linkcheck. |

## Edit Source Files

In this section, I will give instructions for how to make a minimum set of rst files so you can see how it works. If you want to look at source code for other websites, I recommend the following projects, which all have links at the top for “Edit on GitHub” that let you see the rst file source.

* Chemper: <http://chemper.readthedocs.io/en/latest/>
* Openforcefield: <http://open-forcefield-toolkit.readthedocs.io/en/latest/installation.html>
  + The link for these source files says “Source” in the menu bar at the top
* 50 Examples for Teaching Python: <http://fiftyexamples.readthedocs.io/en/latest/index.html>

In the rest of this section I will tell you where these files should be and what to put in them to get this example working.

### index.rst

This file by default is in the source file, however, if your build is going to see all your code you should move it up to your top directory file. Initially your file should look something like this:

|  |
| --- |
| .. shapes2 documentation master file, created by  sphinx-quickstart on Fri Jul 20 17:51:08 2018.  You can adapt this file completely to your liking, but it should at least  contain the root `toctree` directive.  Welcome to shapes2's documentation!  ===================================  .. toctree::  :maxdepth: 2  :caption: Contents:  Indices and tables  ==================  \* :ref:`genindex`  \* :ref:`modindex`  \* :ref:`search` |

We are going to make 3 more rst files, you should put their names in the table of contents tree (toctree) by putting their names under the caption, indented like this:

|  |
| --- |
| .. toctree::  :maxdepth: 2  :caption: Contents:  docs/source/intro  docs/source/functions  docs/source/examples |

### intro.rst

In docs/source make a file called **intro.rst**, mine has this:

|  |
| --- |
| Introduction  ============  Write an introduction here  Contributors  ------------  \* `Caitlin C. Bannan (UCI) <https://github.com/bannanc>`\_ |

Here’s some new formatting:

* ==== under a line make a Title
* ----- under makes a subtitle
* The ` ` with <> around a link followed by \_ will put in an extral link so that line will look like:
  + [Caitlin C. Bannan (UCI)](https://github.com/bannanc)

### function.rst

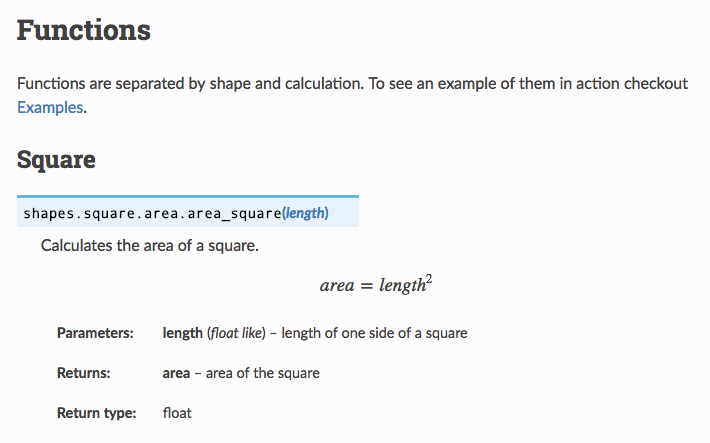
This file will use Sphinx’s auto fill to fill in a custom function or class. Mine looks like this:

|  |
| --- |
| Functions  =========  Functions are separated by shape and calculation. To see an example of them in action checkout :ref:`examples`  Square  ------  .. autofunction:: shapes.square.area.area\_square  .. autofunction:: shapes.square.perimeter.perimeter\_square  Triangle  --------  .. autofunction:: shapes.triangle.area.area\_triangle |

In this implementation it assumes you’re using the Google stype doc strings, such as the one shown below. If you want to use a different doc style you will have to look up how to change your conf.py file.:

|  |
| --- |
| def area\_square(length):  """  Calculates the area of a square.  Parameters  ----------  length (float or int) length of one side of a square  Returns  -------  area (float) - area of the square  """  return length \*\* 2 |

This will automatically grab the functions shapes.square… etc. You can also do this with classes, and it will show all classes and their doc strings. Here is the area\_square output on our website:



### examples.rst

This file will allow you to show example jupyter notebooks you’ve created on your documentation website. This handy as your user can see a complicated use example that they can also download and play with from github if they want to.

|  |
| --- |
| .. \_examples:  Examples  ========  Below are links to examples for inputing jupyter notebooks into your website  .. toctree::  :maxdepth: 1  ../examples/example\_notebook.ipynb |

You can list multiple notebooks, just like the toctree in intro.rst, you can also include these in any toctree in any file you choose.

## Edit Configuration Files

Now we will change the files which are used to build your website pages

### Makefile

This file is located in the docs folder, not in source. We moved the index file up to the top directory so sphinx could find all your code and jupyter notebooks. However, we need to tell the Makefile that. Note, I don’t know anything else about this file, it is possible there are fancier things you can do witht he Makefile, but I’ve always just used the default besides this one change, on line 8 ish, change the SOURCEDIR variable:

|  |
| --- |
| SOURCEDIR = ../ |

### conf.py

These are suggested changes for the conf file, it was originally at sources/conf.py, but it also needs to be moved to the top directory.

Remember, that these are all just suggestions to make the format used in chemper, this file is very customizeable so you may end up needing to change other things. I also don’t understand this well enough to justify all of these decisions.

So All I’m going to show what to change:

* uncomment out import os
  + approx line 15
* update the extensions list to include all of the following:
  + about line 41

|  |
| --- |
| extensions = [  'sphinx.ext.autodoc',  'sphinx.ext.autosummary',  'sphinx.ext.mathjax',  'sphinx.ext.viewcode',  'sphinx.ext.napoleon',  'sphinx.ext.intersphinx'  'nbsphinx',  ] |

* Set the exclude\_patterns as below:
  + about line 73

|  |
| --- |
| exclude\_patterns = ['\_build', '\*\*.ipynb\_checkpoints'] |

* Comment out the line html\_theme = 'alabaster' and add this below it:
  + about line 83

|  |
| --- |
| *# html\_theme = 'alabaster'*  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()]  html\_context = {  'css\_files': [  '\_static/theme\_overrides.css'  ]  }  else:  html\_context = {  'css\_files': [  '//media.readthedocs.org/css/sphinx\_rtd\_theme.css',  '//media.readthedocs.org/css/readthedocs-doc-embed.css',  '\_static/theme\_overrides.css'  ]  } |

* update html\_sidebars to the following
  + about line 126
  + This I can explain, these are things I did not want included in the side bar of my home website, you can play around with these to see if you want them included or not.

|  |
| --- |
| html\_sidebars = {  '\*\*': [  'about.html',  'navigation.html',  'relations.html', # needs 'show\_related': True theme option to display  'searchbox.html',  # 'donate.html', see if I actually need this one  ]  } |

### environment.yml

This is a yml file similar, but simpler than the travis yml file. It tells sphinx/readthedocs which python modules you need to install. Here is what is in my simple example, you’ll need all dependencies for your package. (Side note - technically my example tool doesn’t use openeye, but I wanted an example that included the channels being required. This can also be a list like the dependencies section, just include any conda channel necessary for your dependencies).

|  |
| --- |
| name: chemper  channels:  - openeye  dependencies:  - python==3.5  - openeye-toolkits  - numpydoc  - pip  - pip:  - nbsphinx |

### .readthedocs.yml

This file is in the top directory and tells readthedocs to use conda to install your python environment. It is very short and calls your environment file:

|  |
| --- |
| conda:  file: docs/environment.yml  python:  setup\_py\_install: true  version: 3.5  extra\_requirements:  - tests  - docs |

## Check your build

In the directory with your MakeFile run the following command:

|  |
| --- |
| make html |

To see your build open the file build/html/index.html, this is not a public website yet, but you should be able to use all of your links.

## Push all these files to GitHub

Next you want to add all of these files to a branch in your github repository (this can be master, but below I include directions for how to see the build on other branches as well).

# Set Up ReadTheDocs

If you don’t own your repository, you’ll need to ask the owner (or organization admin) to give you permission in order to do these steps. As in they have to give your username permission, though if you’re on a group project you should all agree on who is creatting the doc website.

### Setup account

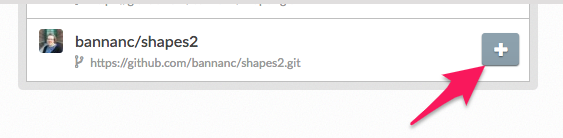
Go to [readthedocs.io](https://readthedocs.org/) and log in through github, its been a while since I did this for the first time, but I think you just say log in with github and then check a box giving it permission to access your repositories.

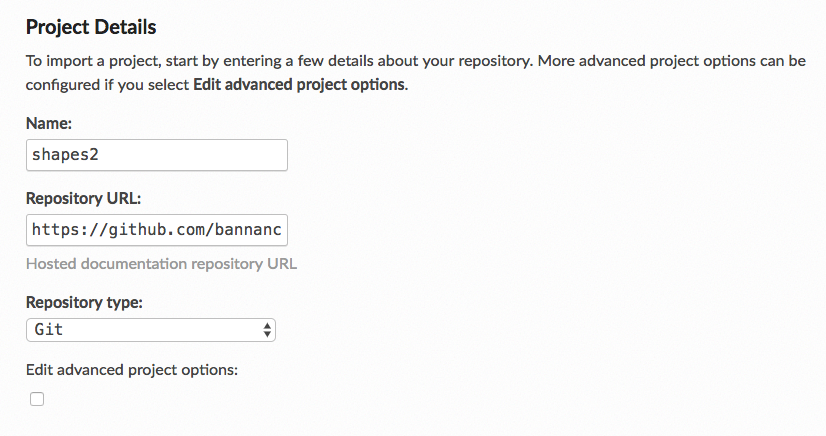
## Create new project on readthedocs:

* Chose import a project



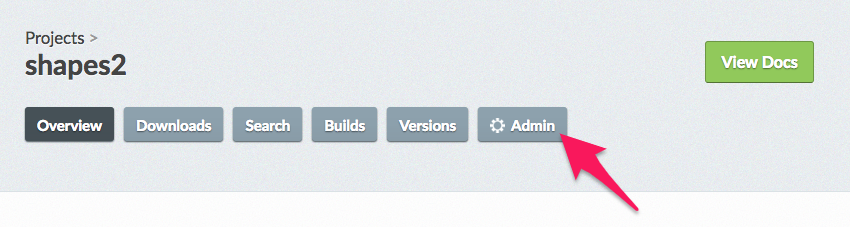
* Find the repository you want to make a documentation page for and click on the “



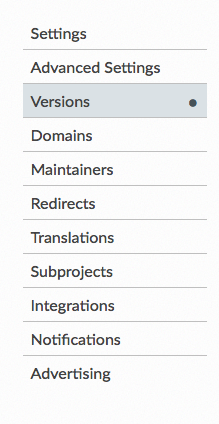
* Assign a name and chose repository type “Git”
  + Note this name has to be unique, that is there can’t be any other readthedocs projects with this name, it is used for your [project].readthedocs.io website. If your project name is available use thate, but if note you may have to come up with a variation
  + Note - For most projects the default advanced options should be fine, but they include options to add a description or a link to the projects home page and tags for your documentation. I believe these can be accessed again after creation, but I’ve never used anything except the defaults.

## Project Settings

Now we will make some small adjustments to make sure your documents are building. You can do this under Admin on the menu bar:



In the sections below, I have descriptions for the things you might want to change. These will not be written in order, but include the important options on the side bar:



### Settings

In this section you can change the “advanced” settings we ignored when initially building the project, however, there shouldn’t be anything you HAVE to change here.

### Versions

On this menu you can chose which branchs from your project you want to have documentation built for. This is handy for testing your documentation before pushing it into the master branch. So you should select active next under that branch name:



Then you can chose if you want it to be public, there are also protected and private builds, but I’m not sure who can(not) see the other options…

## Trigger a build

You can force a new build on your project by going to “Builds” on the main menu and pressing the “Build Version” button.

# Check out your website

## Add Badge to your Repo

Like Travis and CodeCov, ReadTheDocs has a badge you can add to your repository.

On the homepage of the read the docs for your project on the right side you should see this:

If you click on the badge itself, there are options for RST, MarkDown, or HTML that will allow you to add this badge to your repo.

## Website

The website for your docs uses the name you created during setup in the form [name].readthedocs.io