
nbsphinx

Release 0.3.5

Matthias Geier

2018-09-10

Contents

1	Installation	1
1.1	nbsphinx Packages	2
1.2	nbsphinx Prerequisites	2
1.2.1	Python	2
1.2.2	Sphinx	2
1.2.3	pip	3
1.2.4	pandoc	3
1.2.5	Pygments Lexer for Syntax Highlighting	3
1.2.6	Jupyter Kernel	4
2	Usage	4
2.1	Sphinx Setup	4
2.2	Running Sphinx	5
2.3	Watching for Changes with <code>sphinx-autobuild</code>	5
2.4	Automatic Creation of HTML and PDF output on <code>readthedocs.org</code>	6
2.4.1	Using <code>requirements.txt</code>	6
2.4.2	Using <code>conda</code>	7
2.5	HTML Themes	7
2.5.1	Sphinx's Built-In Themes	8
2.5.2	3rd-Party Themes	8
2.6	Using Notebooks with Git	9
3	Markdown Cells	10
3.1	Equations	10
3.2	Code	10
3.3	Tables	11
3.4	Images	11
3.5	HTML Elements (HTML only)	12
3.6	Info/Warning Boxes	12
3.7	Links to Other Notebooks	13
3.8	Links to *.rst Files (and Other Sphinx Source Files)	14
3.9	Links to Local Files (HTML only)	14
3.10	Links to Domain Objects	14
4	Code Cells	15
4.1	Code, Output, Streams	15
4.2	Cell Magics	15
4.3	Special Display Formats	16
4.3.1	Local Image Files	16

4.3.2	Image URLs	18
4.3.3	Math	19
4.3.4	YouTube Videos	19
4.3.5	Arbitrary JavaScript Output (HTML only)	20
4.3.6	Unsupported Output Types	20
4.4	ANSI Colors	21
5	Raw Cells	23
5.1	Usage	23
5.2	Available Raw Cell Formats	23
5.2.1	None	23
5.2.2	reST	24
5.2.3	Markdown	24
5.2.4	HTML	24
5.2.5	LaTeX	24
5.2.6	Python	24
6	Hidden Cells	24
7	Controlling Notebook Execution	25
7.1	Pre-Executing Notebooks	25
7.1.1	Long-Running Cells	25
7.1.2	Rare Libraries	25
7.1.3	Exceptions	26
7.2	Explicitly Dis-/Enabling Notebook Execution	26
7.3	Ignoring Errors	26
7.4	Cell Execution Timeout	28
8	Prolog and Epilog	28
8.1	Examples	29
9	Notebooks in Sub-Directories	30
9.1	A Sub-Section	31
10	Using toctree In A Notebook	32
10.1	Yet Another Notebook	33
11	Normal reStructuredText Files	34
11.1	Links to Notebooks (and Other Sphinx Source Files)	34
11.2	Links to Notebooks, Ye Olde Way	34
11.3	Sphinx Directives for Info/Warning Boxes	35
11.4	Domain Objects	36
12	External Links	37

nbsphinx is a [Sphinx](http://sphinx-doc.org/)¹ extension that provides a source parser for *.ipynb files. Custom Sphinx directives are used to show [Jupyter Notebook](http://jupyter.org/)² code cells (and of course their results) in both HTML and LaTeX output. Un-evaluated notebooks – i.e. notebooks without stored output cells – will be automatically executed during the Sphinx build process.

Quick Start:

1. Install nbsphinx

¹ <http://sphinx-doc.org/>

² <http://jupyter.org/>

2. Edit your `conf.py` and add `'nbsphinx'` to extensions.
3. Edit your `index.rst` and add the names of your `*.ipynb` files to the `toctree`.
4. Run Sphinx!

Online documentation (and example of use): <http://nbsphinx.readthedocs.io/>

Source code repository (and issue tracker): <https://github.com/spatialaudio/nbsphinx/>

License: MIT – see the file `LICENSE` for details.

All content shown below – except for the section *Normal reStructuredText Files* – was generated from Jupyter notebooks.

The following section was generated from `doc/installation.ipynb`

1 Installation

Note that some packages may be out-of-date. You can always get the newest `nbsphinx` release from PyPI³ (using `pip`). If you want to try the latest development version, have a look at the file `CONTRIBUTING.rst`⁴.

1.1 nbsphinx Packages

5

If you are using the `conda` package manager (e.g. with `Anaconda`⁶ for Linux/macOS/Windows), you can install `nbsphinx` from the `conda-forge`⁷ channel:

```
conda install -c conda-forge nbsphinx
```

If you are using Linux, there are packages available for many distributions.

8

9

On any platform, you can also install `nbsphinx` with `pip`, Python's own package manager:

```
python3 -m pip install nbsphinx --user
```

If you want to install it system-wide for all users (assuming you have the necessary rights), just drop the `--user` flag.

To upgrade an existing `nbsphinx` installation to the newest release, use the `--upgrade` flag:

```
python3 -m pip install nbsphinx --upgrade --user
```

If you suddenly change your mind, you can un-install it with:

```
python3 -m pip uninstall nbsphinx
```

Depending on your Python installation, you may have to use `python` instead of `python3`.

³ <https://pypi.org/project/nbsphinx>

⁴ <https://github.com/spatialaudio/nbsphinx/blob/master/CONTRIBUTING.rst>

⁵ <https://anaconda.org/conda-forge/nbsphinx>

⁶ <https://www.anaconda.com/download/>

⁷ <https://conda-forge.org/>

⁸ <https://repology.org/metapackage/python:nbsphinx/versions>

⁹ <https://pypi.org/project/nbsphinx>

1.2 nbsphinx Prerequisites

Some of the aforementioned packages will install some of these prerequisites automatically, some of the things may be already installed on your computer anyway.

1.2.1 Python

Of course you'll need Python, because both Sphinx and nbsphinx are implemented in Python. There are many ways to get Python. If you don't know which one is best for you, you can try [Anaconda](#)¹⁰.

1.2.2 Sphinx

You'll need [Sphinx](#)¹¹ as well, because nbsphinx is just a Sphinx extension and doesn't do anything on its own.

If you use conda, you can get [Sphinx from the conda-forge channel](#)¹²:

```
conda install -c conda-forge sphinx
```

Alternatively, you can install it with pip (see below):

```
python3 -m pip install Sphinx --user
```

1.2.3 pip

Recent versions of Python already come with pip pre-installed. If you don't have it, you can [install it manually](#)¹³.

1.2.4 pandoc

The stand-alone program [pandoc](#)¹⁴ is used to convert Markdown content to something Sphinx can understand. You have to install this program separately, ideally with your package manager. If you are using conda, you can install [pandoc from the conda-forge channel](#)¹⁵:

```
conda install -c conda-forge pandoc
```

If that doesn't work out for you, have a look at pandoc's [installation instructions](#)¹⁶.

Note:

The use of pandoc in nbsphinx is temporary, but will likely stay that way for a long time, see [issue #36](#)¹⁷.

1.2.5 Pygments Lexer for Syntax Highlighting

To get proper syntax highlighting in code cells, you'll need an appropriate *Pygments lexer*. This of course depends on the programming language of your Jupyter notebooks (more specifically, the `pygments_lexer` metadata of your notebooks).

¹⁰ <https://www.anaconda.com/download/>

¹¹ <http://www.sphinx-doc.org/>

¹² <https://anaconda.org/conda-forge/sphinx>

¹³ <https://pip.pypa.io/en/latest/installing/>

¹⁴ <http://pandoc.org/>

¹⁵ <https://anaconda.org/conda-forge/pandoc>

¹⁶ <http://pandoc.org/installing.html>

¹⁷ <https://github.com/spatialaudio/nbsphinx/issues/36>

For example, if you use Python in your notebooks, you'll have to have the IPython package installed, e.g. with

```
conda install -c conda-forge ipython
```

or

```
python3 -m pip install IPython --user
```

Note:

If you are using Anaconda with the default channel and syntax highlighting in code cells doesn't seem to work, you can try to install IPython from the conda-forge channel or directly with pip, or as a work-around, add 'IPython.sphinxext.ipython_console_highlighting' to extensions in your `conf.py`.

For details, see [Anaconda issue #1430](#)¹⁸ and [nbsphinx issue #24](#)¹⁹.

1.2.6 Jupyter Kernel

If you want to execute your notebooks during the Sphinx build process (see *Controlling Notebook Execution*), you need an appropriate [Jupyter kernel](#)²⁰ installed.

For example, if you use Python, you should install the `ipykernel` package, e.g. with

```
conda install -c conda-forge ipykernel
```

or

```
python3 -m pip install ipykernel --user
```

If you created your notebooks yourself with Jupyter, it's very likely that you have the right kernel installed already.

End of doc/installation.ipynb

The following section was generated from doc/usage.ipynb

2 Usage

2.1 Sphinx Setup

In the directory with your notebook files, run this command (assuming you have [Sphinx](#)²¹ installed already):

```
python3 -m sphinx.quickstart
```

Answer the questions that appear on the screen. In case of doubt, just press the <Return> key repeatedly to take the default values.

After that, there will be a few brand-new files in the current directory. You'll have to make a few changes to the file named `conf.py`. You should at least check if those two variables contain the right things:

¹⁸ <https://github.com/ContinuumIO/anaconda-issues/issues/1430>

¹⁹ <https://github.com/spatialaudio/nbsphinx/issues/24>

²⁰ <http://jupyter.readthedocs.io/en/latest/projects/kernels.html>

²¹ <http://sphinx-doc.org/>

```
extensions = [
    'nbsphinx',
    'sphinx.ext.mathjax',
]
exclude_patterns = ['_build', '**.ipynb_checkpoints']
```

Once your `conf.py` is in place, edit the file named `index.rst` and add the file names of your notebooks (with or without the `.ipynb` extension) to the `toctree`²² directive.

autosummary bug:

If you are using the `sphinx.ext.autosummary` Sphinx extension, there is a [bug in Sphinx \(below version 1.5\)](#)²³ which prevents notebooks from being parsed. As a work-around you can explicitly list all the files for which autosummary should be ran using the `autosummary_generate`²⁴ variable in `conf.py`. For example,

```
autosummary_generate = ['myfile1.rst', 'myfile2.rst']
```

2.2 Running Sphinx

To create the HTML pages, use this command:

```
python3 -m sphinx <source-dir> <build-dir>
```

If you have many notebooks, you can do a parallel build by using the `-j` option:

```
python3 -m sphinx <source-dir> <build-dir> -j<number-of-processes>
```

For example, if your source files are in the current directory and you have 4 CPU cores, you can run this:

```
python3 -m sphinx . _build -j4
```

Afterwards, you can find the main HTML file in `_build/index.html`.

Subsequent builds will be faster, because only those source files which have changed will be re-built. To force re-building all source files, use the `-E` option.

Note:

By default, notebooks will be executed during the Sphinx build process only if they do not have any output cells stored. See [Controlling Notebook Execution](#).

To create LaTeX output, use:

```
python3 -m sphinx <source-dir> <build-dir> -b latex
```

If you don't know how to create a PDF file from the LaTeX output, you should have a look at [Latexmk](#)²⁵ (see also [this tutorial](#)²⁶).

Sphinx can automatically check if the links you are using are still valid. Just invoke it like this:

```
python3 -m sphinx <source-dir> <build-dir> -b linkcheck
```

²² <http://www.sphinx-doc.org/en/master/usage/restructuredtext/directives.html#directive-toctree>

²³ <https://github.com/sphinx-doc/sphinx/issues/2485>

²⁴ http://www.sphinx-doc.org/en/master/ext/autosummary.html#confval-autosummary_generate

²⁵ <http://personal.psu.edu/jcc8//software/latexmk-jcc/>

²⁶ <http://mg.readthedocs.io/latexmk.html>

2.3 Watching for Changes with sphinx-autobuild

If you think it's tedious to run the Sphinx build command again and again while you make changes to your notebooks, you'll be happy to hear that there is a way to avoid that: `sphinx-autobuild`²⁷!

It can be installed with

```
python3 -m pip install sphinx-autobuild --user
```

You can start auto-building your files with

```
python3 -m sphinx_autobuild <source-dir> <build-dir>
```

This will start a local webserver which will serve the generated HTML pages at <http://localhost:8000/>. Whenever you save changes in one of your notebooks, the appropriate HTML page(s) will be re-built and when finished, your browser view will be refreshed automatically. Neat!

You can also abuse this to auto-build the LaTeX output:

```
python3 -m sphinx_autobuild <source-dir> <build-dir> -b latex
```

However, to auto-build the final PDF file as well, you'll need an additional tool. Again, you can use `latexmk` for this (see [above](#)). Change to the build directory and run

```
latexmk -pdf -pvc
```

If your PDF viewer isn't opened because of LaTeX build errors, you can use the command line flag `-f` to *force* creating a PDF file.

2.4 Automatic Creation of HTML and PDF output on readthedocs.org²⁸

There are two different methods, both of which are described below.

In both cases, you'll first have to create an account on <https://readthedocs.org/> and connect your Github/Bitbucket account. Instead of connecting, you can also manually add any publicly available Git/Subversion/Mercurial/Bazaar repository.

After doing the steps described below, you only have to "push" to your repository, and the HTML pages and the PDF file of your stuff are automatically created on readthedocs.org. Awesome!

You can even have different versions of your stuff, just use Git tags and branches and select in the readthedocs.org settings (under "Admin", "Versions") which of those should be created.

If your new versions are not automatically built, go to the "Settings" of your Github repository, continue to "Integrations & services", and make sure that "ReadTheDocs" is listed and activated in the "Services" section. If not, use "Add service". There is probably a similar thing for Bitbucket and others.

Note:

If you want to execute notebooks (see [Controlling Notebook Execution](#)), you'll need to install the appropriate Jupyter kernel. In the examples below, the IPython kernel is installed from the package `ipykernel`.

2.4.1 Using requirements.txt

1. Create a file named `requirements.txt` (or whatever name you wish) in your repository containing the required pip packages:

²⁷ <https://pypi.org/project/sphinx-autobuild>

²⁸ <https://readthedocs.org>

```
sphinx>=1.4
ipykernel
nbsphinx
```

You can also install directly from Github et al., using a specific branch/tag/commit, e.g.

```
git+https://github.com/spatialaudio/nbsphinx.git@master
```

2. In the “Advanced Settings” on readthedocs.org, specify the path to your `requirements.txt` file (or however you called it) in the box labeled “Requirements file”. Kinda obvious, isn’t it?
3. Still in the “Advanced Settings”, make sure the right Python interpreter is chosen. This must be the same version (2.x or 3.x) as you were using in your notebooks!

2.4.2 Using conda

1. Create a file named `readthedocs.yml` in the main directory of your repository, containing the name of yet another file:

```
conda:
  file: readthedocs-environment.yml
```

2. Create the file mentioned above. You can choose whatever name you want (it may also live in a subdirectory, e.g. `doc/environment.yml`), it just has to match whatever is specified in `readthedocs.yml`. The second file describes a [conda environment](#)²⁹ and should contain something like this:

```
channels:
  - conda-forge
dependencies:
  - python==3.5
  - sphinx>=1.4
  - pandoc
  - nbconvert
  - ipykernel
  - pip:
    - nbsphinx
```

It is up to you if you want to install `nbsphinx` with `conda` or with `pip` (but note that the `conda` package might be outdated). And you can of course add further `conda` and `pip` packages. You can also install packages directly from Github et al., using a specific branch/tag/commit, e.g.

```
- pip:
  - git+https://github.com/spatialaudio/nbsphinx.git@master
```

Note:

The specification of the `conda-forge` channel is recommended because it tends to have more recent package versions than the default channel.

Note:

Most of the “Advanced Settings” on readthedocs.org will be ignored if you have a `readthedocs.yml` file.

Warning:

²⁹ <https://conda.io/docs/using/envs.html>

If you have a very long repository name (or branch name), you might run into this quite obscure problem: ‘placeholder too short’³⁰.

2.5 HTML Themes

The nbsphinx extension does *not* provide its own theme, you can use any of the available themes or create a custom one³¹, if you feel like it.

The following (incomplete) list of themes contains up to three links for each theme:

1. The documentation (or the official sample page) of this theme (if available; see also the [documentation of the built-in Sphinx themes](#)³²)
2. How the nbsphinx documentation looks when using this theme
3. How to enable this theme using either `requirements.txt` or `readthedocs.yml` and theme-specific settings (in some cases)

2.5.1 Sphinx’s Built-In Themes

- alabaster³³: [example](#)³⁴, [usage](#)³⁵
- pyramid: [example](#)³⁶, [usage](#)³⁷
- classic: [example](#)³⁸, [usage](#)³⁹
- bizstyle: [example](#)⁴⁰, [usage](#)⁴¹
- haiku: [example](#)⁴², [usage](#)⁴³
- traditional: [example](#)⁴⁴, [usage](#)⁴⁵
- agogo: [example](#)⁴⁶, [usage](#)⁴⁷
- nature: [example](#)⁴⁸, [usage](#)⁴⁹

2.5.2 3rd-Party Themes

- sphinx_rtd_theme⁵⁰: [example](#)⁵¹, [usage](#)⁵²

³⁰ <https://github.com/rtfd/readthedocs.org/issues/1902>

³¹ <http://www.sphinx-doc.org/en/master/theming.html#creating-themes>

³² <http://www.sphinx-doc.org/en/latest/theming.html#builtin-themes>

³³ <https://alabaster.readthedocs.io/>

³⁴ <http://nbsphinx.readthedocs.io/en/alabaster-theme/>

³⁵ <https://github.com/spatialaudio/nbsphinx/compare/alabaster-theme%5E...alabaster-theme>

³⁶ <http://nbsphinx.readthedocs.io/en/pyramid-theme/>

³⁷ <https://github.com/spatialaudio/nbsphinx/compare/pyramid-theme%5E...pyramid-theme>

³⁸ <http://nbsphinx.readthedocs.io/en/classic-theme/>

³⁹ <https://github.com/spatialaudio/nbsphinx/compare/classic-theme%5E...classic-theme>

⁴⁰ <http://nbsphinx.readthedocs.io/en/bizstyle-theme/>

⁴¹ <https://github.com/spatialaudio/nbsphinx/compare/bizstyle-theme%5E...bizstyle-theme>

⁴² <http://nbsphinx.readthedocs.io/en/haiku-theme/>

⁴³ <https://github.com/spatialaudio/nbsphinx/compare/haiku-theme%5E...haiku-theme>

⁴⁴ <http://nbsphinx.readthedocs.io/en/traditional-theme/>

⁴⁵ <https://github.com/spatialaudio/nbsphinx/compare/traditional-theme%5E...traditional-theme>

⁴⁶ <http://nbsphinx.readthedocs.io/en/agogo-theme/>

⁴⁷ <https://github.com/spatialaudio/nbsphinx/compare/agogo-theme%5E...agogo-theme>

⁴⁸ <http://nbsphinx.readthedocs.io/en/nature-theme/>

⁴⁹ <https://github.com/spatialaudio/nbsphinx/compare/nature-theme%5E...nature-theme>

⁵⁰ https://github.com/rtfd/sphinx_rtd_theme

⁵¹ <http://nbsphinx.readthedocs.io/en/rtd-theme/>

⁵² <https://github.com/spatialaudio/nbsphinx/compare/rtd-theme%5E...rtd-theme>

- [bootstrap](#)⁵³: [example](#)⁵⁴, [usage](#)⁵⁵
- [cloud](#)⁵⁶: [example](#)⁵⁷, [usage](#)⁵⁸
- [sphinx_py3doc_enhanced_theme](#)⁵⁹: [example](#)⁶⁰, [usage](#)⁶¹
- [basicstrap](#)⁶²: [example](#)⁶³, [usage](#)⁶⁴
- [dotted](#)⁶⁵: [example](#)⁶⁶, [usage](#)⁶⁷
- [better](#)⁶⁸: [example](#)⁶⁹, [usage](#)⁷⁰
- [guzzle_sphinx_theme](#)⁷¹: [example](#)⁷², [usage](#)⁷³
- [julia](#)⁷⁴: [example](#)⁷⁵, [usage](#)⁷⁶
- [jupyter](#)⁷⁷: [example](#)⁷⁸, [usage](#)⁷⁹

If you know of another Sphinx theme that should be included here, please open an [issue on Github](#)⁸⁰. An overview of many more themes can be found at <https://sphinx-themes.org/>.

2.6 Using Notebooks with Git

[Git](#)⁸¹ is extremely useful for managing source code and it can and should also be used for managing Jupyter notebooks. There is one caveat, however: Notebooks can contain output cells with rich media like images, plots, sounds, HTML, JavaScript and many other types of bulky machine-created content. This can make it hard to work with Git efficiently, because changes in those bulky contents can completely obscure the more interesting human-made changes in text and source code. Working with multiple collaborators on a notebook can become very tedious because of this.

It is therefore highly recommended that you remove all outputs from your notebooks before committing changes to a Git repository (except for the reasons mentioned in *Pre-Executing Notebooks*).

If there are no output cells in a notebook, `nbsphinx` will by default execute the notebook, and the pages generated by Sphinx will therefore contain all the output cells. See *Controlling Notebook Execution* for how this behavior can be customized.

⁵³ <http://sphinx-bootstrap-theme.readthedocs.io/>

⁵⁴ <http://nbsphinx.readthedocs.io/en/bootstrap-theme/>

⁵⁵ <https://github.com/spatialaudio/nbsphinx/compare/bootstrap-theme%5E...bootstrap-theme>

⁵⁶ https://pythonhosted.org/cloud_sptheme/

⁵⁷ <http://nbsphinx.readthedocs.io/en/cloud-theme/>

⁵⁸ <https://github.com/spatialaudio/nbsphinx/compare/cloud-theme%5E...cloud-theme>

⁵⁹ <https://github.com/ionelmc/sphinx-py3doc-enhanced-theme>

⁶⁰ <http://nbsphinx.readthedocs.io/en/py3doc-enhanced-theme/>

⁶¹ <https://github.com/spatialaudio/nbsphinx/compare/py3doc-enhanced-theme%5E...py3doc-enhanced-theme>

⁶² <https://pythonhosted.org/sphinxjp.themes.basicstrap/>

⁶³ <http://nbsphinx.readthedocs.io/en/basicstrap-theme/>

⁶⁴ <https://github.com/spatialaudio/nbsphinx/compare/basicstrap-theme%5E...basicstrap-theme>

⁶⁵ <https://pythonhosted.org/sphinxjp.themes.dotted/>

⁶⁶ <http://nbsphinx.readthedocs.io/en/dotted-theme/>

⁶⁷ <https://github.com/spatialaudio/nbsphinx/compare/dotted-theme%5E...dotted-theme>

⁶⁸ <https://sphinx-better-theme.readthedocs.io/>

⁶⁹ <http://nbsphinx.readthedocs.io/en/better-theme/>

⁷⁰ <https://github.com/spatialaudio/nbsphinx/compare/better-theme%5E...better-theme>

⁷¹ https://github.com/guzzle/guzzle_sphinx_theme

⁷² <http://nbsphinx.readthedocs.io/en/guzzle-theme/>

⁷³ <https://github.com/spatialaudio/nbsphinx/compare/guzzle-theme%5E...guzzle-theme>

⁷⁴ <https://github.com/JuliaLang/JuliaDoc>

⁷⁵ <http://nbsphinx.readthedocs.io/en/julia-theme/>

⁷⁶ <https://github.com/spatialaudio/nbsphinx/compare/julia-theme%5E...julia-theme>

⁷⁷ <https://github.com/jupyter/jupyter-sphinx-theme/>

⁷⁸ <http://nbsphinx.readthedocs.io/en/jupyter-theme/>

⁷⁹ <https://github.com/spatialaudio/nbsphinx/compare/jupyter-theme%5E...jupyter-theme>

⁸⁰ <https://github.com/spatialaudio/nbsphinx/issues>

⁸¹ <https://git-scm.com/>

In the Jupyter Notebook application, you can manually clear all outputs by selecting “Cell” → “All Output” → “Clear” from the menu.

There are several tools available to remove outputs from multiple files at once without having to open them separately. You can even include such a tool as “clean/smudge filters” into your Git workflow, which will strip the output cells automatically whenever a Git command is executed. For details, have a look at those links:

- <https://github.com/kynan/nbstripout>
- https://github.com/toobaz/ipynb_output_filter
- <http://tillahoffmann.github.io/2017/04/17/versioning-jupyter-notebooks-with-git.html>
- <http://timestaley.co.uk/posts/making-git-and-jupyter-notebooks-play-nice/>
- <http://pascalbugnion.net/blog/ipython-notebooks-and-git.html>
- <https://github.com/choldgraf/nbclean>

End of doc/usage.ipynb

The following section was generated from doc/markdown-cells.ipynb

3 Markdown Cells

We can use *emphasis*, **boldface**, preformatted text.

It looks like strike-out text is not supported: [STRIKEOUT:strikethrough].

- Red
- Green
- Blue

-
1. One
 2. Two
 3. Three

Arbitrary Unicode characters should be supported, e.g. $\alpha\beta\delta$. Note, however, that this only works if your HTML browser and your LaTeX processor provide the appropriate fonts.

3.1 Equations

Equations can be formatted really nicely, either inline, like $e^{i\pi} = -1$, or on a separate line, like

$$\int_{-\infty}^{\infty} f(x)\delta(x - x_0)dx = f(x_0) \tag{1}$$

Note: Avoid leading and trailing spaces around math expressions, otherwise errors like the following will occur when Sphinx is running:

ERROR: Unknown interpreted text role "raw-latex".

See also the [pandoc docs](#)⁸²:

⁸² <http://pandoc.org/MANUAL.html#math>

Anything between two \$ characters will be treated as TeX math. The opening \$ must have a non-space character immediately to its right, while the closing \$ must have a non-space character immediately to its left, and must not be followed immediately by a digit.

3.2 Code

We can also write code with nice syntax highlighting:

```
print("Hello, world!")
```

3.3 Tables

A	B	A and B
False	False	False
True	False	False
False	True	False
True	True	True

3.4 Images



PNG file (local):

SVG file (local):



PNG file (remote):

SVG file (remote):

3.5 HTML Elements (HTML only)

It is allowed to use plain HTML elements within Markdown cells. Those elements are passed through to the HTML output and are ignored for the LaTeX output. Below are a few examples.

HTML5 `audio`⁸³ elements can be created like this:

```
<audio src="https://example.org/audio.ogg" controls>alternative text</audio>
```

Example:

The HTML audio element is not supported!

HTML5 `video`⁸⁴ elements can be created like this:

```
<video src="https://example.org/video.ogv" controls>alternative text</video>
```

Example:

The HTML video element is not supported!

The alternative text is shown in browsers that don't support those elements. The same text is also shown in Sphinx's LaTeX output.

Note: You can also use local files for the `<audio>` and `<video>` elements, but you have to create a link to the source file somewhere, because only then are the local files copied to the HTML output directory! You should do that anyway to make the audio/video file accessible to browsers that don't support the `<audio>` and `<video>` elements.

⁸³ <https://developer.mozilla.org/en-US/docs/Web/HTML/Element/audio>

⁸⁴ <https://developer.mozilla.org/en-US/docs/Web/HTML/Element/video>

3.6 Info/Warning Boxes

Warning:

This is an *experimental feature*! Its usage will probably change in the future or it might be removed completely!

Until there is an info/warning extension for Markdown/CommonMark (see [this issue](#)⁸⁵), such boxes can be created by using HTML `<div>` elements like this:

```
<div class="alert alert-info">
**Note:** This is a note!
</div>
```

For this to work reliably, you should obey the following guidelines:

- The `class` attribute has to be either `"alert alert-info"` or `"alert alert-warning"`, other values will not be converted correctly.
- No further attributes are allowed.
- For compatibility with CommonMark, you should add an empty line between the `<div>` start tag and the beginning of the content.

Note:

The text can contain further Markdown formatting. It is even possible to have nested boxes:

... but please don't *overuse* this!

3.7 Links to Other Notebooks

Relative links to local notebooks can be used: *a link to a notebook in a subdirectory*, a link to an orphan notebook (latter won't work in LaTeX output, because orphan pages are not included there).

This is how a link is created in Markdown:

```
[a link to a notebook in a subdirectory](subdir/a-notebook-in-a-subdir.ipynb)
```

Markdown also supports *reference-style* links: *a reference-style link, another version of the same link*.

These can be created with this syntax:

```
[a reference-style link][mylink]
[mylink]: subdir/a-notebook-in-a-subdir.ipynb
```

Links to sub-sections are also possible, e.g. *this subsection*.

This link was created with:

```
[this subsection](subdir/a-notebook-in-a-subdir.ipynb#A-Sub-Section)
```

You just have to remember to replace spaces with hyphens!

BTW, links to sections of the current notebook work, too, e.g. *beginning of this section*.

This can be done, as expected, like this:

⁸⁵ <https://github.com/jupyter/notebook/issues/1292>

```
[beginning of this section] (#Links-to-Other-Notebooks)
```

3.8 Links to *.rst Files (and Other Sphinx Source Files)

Links to files whose extension is in the configuration value `source_suffix`⁸⁶, will be converted to links to the generated HTML/LaTeX pages. Example: *A reStructuredText file*.

This was created with:

```
[A reStructuredText file] (a-normal-rst-file.rst)
```

Links to sub-sections are also possible. Example: *Sphinx Directives*.

This was created with:

```
[Sphinx Directives] (a-normal-rst-file.rst#sphinx-directives-for-info-warning-boxes)
```

Note:

Sphinx section anchors are different from Jupyter section anchors! To create a link to a subsection in an `.rst` file (or another non-notebook source file), you not only have to replace spaces with hyphens, but also slashes and some other characters. In case of doubt, just check the target HTML page generated by Sphinx.

3.9 Links to Local Files (HTML only)

Links to local files (other than Jupyter notebooks and other Sphinx source files) are also possible, e.g. *requirements.txt*.

This was simply created with:

```
[requirements.txt] (requirements.txt)
```

The linked files are automatically copied to the HTML output directory. For LaTeX output, no link is created.

3.10 Links to Domain Objects

Links to Sphinx domain objects⁸⁷ (such as a Python class or JavaScript function) are also possible. For example: *example_python_function*.

This was created with:

```
[example_python_function] (a-normal-rst-file.rst#example_python_function)
```

This is especially useful for use with the Sphinx `autodoc`⁸⁸ extension!

End of doc/markdown-cells.ipynb

The following section was generated from doc/code-cells.ipynb

⁸⁶ http://www.sphinx-doc.org/en/master/config.html#confval-source_suffix

⁸⁷ <http://www.sphinx-doc.org/en/master/usage/restructuredtext/domains.html>

⁸⁸ <http://www.sphinx-doc.org/en/master/ext/autodoc.html>

4 Code Cells

4.1 Code, Output, Streams

An empty code cell:

```
In [1]:
```

Two empty lines:

```
In [1]:
```

Leading/trailing empty lines:

```
In [1]:
```

```
# 2 empty lines before, 1 after
```

A simple output:

```
In [2]: 6 * 7
```

```
Out[2]: 42
```

The standard output stream:

```
In [3]: print('Hello, world!')
```

```
Hello, world!
```

Normal output + standard output

```
In [4]: print('Hello, world!')
        6 * 7
```

```
Hello, world!
```

```
Out[4]: 42
```

The standard error stream is highlighted and displayed just below the code cell. The standard output stream comes afterwards (with no special highlighting). Finally, the “normal” output is displayed.

```
In [5]: import sys
```

```
print("I'll appear on the standard error stream", file=sys.stderr)
print("I'll appear on the standard output stream")
"I'm the 'normal' output"
```

```
I'll appear on the standard output stream
```

```
I'll appear on the standard error stream
```

```
Out[5]: "I'm the 'normal' output"
```

4.2 Cell Magics

IPython can handle code in other languages by means of [cell magics](#)⁸⁹:

```
In [6]: %%bash
        for i in 1 2 3
        do
            echo $i
        done
```

⁸⁹ <http://ipython.readthedocs.io/en/stable/interactive/magics.html#cell-magics>

1
2
3

4.3 Special Display Formats

See IPython example notebook⁹⁰.

TODO: tables? e.g. Pandas DataFrame?

```
In [7]: from IPython.display import display
```

4.3.1 Local Image Files

```
In [8]: from IPython.display import Image  
i = Image(filename='images/notebook_icon.png')  
i
```



```
In [9]: display(i)
```

⁹⁰ <https://nbviewer.jupyter.org/github/ipython/ipython/blob/master/examples/IPython%20Kernel/Rich%20Output.ipynb>



For some reason this doesn't work with `Image(...)`:

```
In [10]: from IPython.display import SVG
         SVG(filename='images/python_logo.svg')
```

4.3.2 Image URLs

```
In [11]: Image(url='https://www.python.org/static/img/python-logo-large.png')
```

```
Out[11]: <IPython.core.display.Image object>
```

```
In [12]: Image(url='https://www.python.org/static/img/python-logo-large.png', embed=True)
```



```
In [13]: Image(url='http://jupyter.org/assets/nav_logo.svg')
```

```
Out[13]: <IPython.core.display.Image object>
```

```
In [14]: Image(url='https://www.python.org/static/favicon.ico')
```

```
Out[14]: <IPython.core.display.Image object>
```

```
In [15]: Image(url='http://python.org/images/python-logo.gif')
```

```
Out[15]: <IPython.core.display.Image object>
```

4.3.3 Math

```
In [16]: from IPython.display import Math
```

```
eq = Math(r"\int_{-\infty}^{\infty} f(x) \delta(x - x_0) dx = f(x_0)")
```

```
eq
```

$$\int_{-\infty}^{\infty} f(x)\delta(x - x_0)dx = f(x_0)$$

```
In [17]: display(eq)
```

$$\int_{-\infty}^{\infty} f(x)\delta(x - x_0)dx = f(x_0)$$

```
In [18]: %%latex
```

```
\begin{equation}
```

```
\int_{-\infty}^{\infty} f(x) \delta(x - x_0) dx = f(x_0)
```

```
\end{equation}
```

$$\int_{-\infty}^{\infty} f(x)\delta(x - x_0)dx = f(x_0)$$

(2)

4.3.4 YouTube Videos

```
In [19]: from IPython.display import YouTubeVideo
```

```
YouTubeVideo('WAikxUGbomY')
```



4.3.5 Arbitrary JavaScript Output (HTML only)

In [20]: `%%javascript`

```
var text = document.createTextNode("Hello, I was generated with JavaScript!");  
// Content appended to "element" will be visible in the output area:  
element.appendChild(text);
```

<IPython.core.display.Javascript object>

Note:

jQuery should be available, but using the readthedocs.org default theme, it's not. See the issue on [Github](#)⁹¹. Other Sphinx themes are not affected by this.

4.3.6 Unsupported Output Types

If a code cell produces data with an unsupported MIME type, the Jupyter Notebook doesn't generate any output. nbsphinx, however, shows a warning message.

```
In [21]: display({  
    'text/x-python': 'print("Hello, world!")',  
    'text/x-haskell': 'main = putStrLn "Hello, world!"',  
  }, raw=True)
```

⁹¹ https://github.com/rtfd/sphinx_rtd_theme/issues/328

Data type cannot be displayed: text/x-haskell, text/x-python

4.4 ANSI Colors

The standard output and standard error streams may contain ANSI escape sequences⁹² to change the text and background colors.

```
In [22]: print('BEWARE: \x1b[1;33;41mugly colors\x1b[m!', file=sys.stderr)
        print('AB\x1b[43mCD\x1b[35mEF\x1b[1mGH\x1b[4mIJ\x1b[7m'
              'KL\x1b[49mMN\x1b[39mOP\x1b[22mQR\x1b[24mST\x1b[27mUV')
```

ABCD **GH**IJKLMN**OP**QRST**UV**

BEWARE: **ugly colors!**

The following code showing the 8 basic ANSI colors is based on <http://tldp.org/HOWTO/Bash-Prompt-HOWTO/x329.html>. Each of the 8 colors has an “intense” variation, which is used for bold text.

```
In [23]: text = ' XYZ '
        formatstring = '\x1b[{}m' + text + '\x1b[m'

        print(' ' * 6 + ' ' * len(text) +
              ''.join(':{~{}}'.format(bg, len(text)) for bg in range(40, 48)))
        for fg in range(30, 38):
            for bold in False, True:
                fg_code = ('1;' if bold else '') + str(fg)
                print(' {:>4} '.format(fg_code) + formatstring.format(fg_code) +
                      ''.join(formatstring.format(fg_code + ';' + str(bg))
                              for bg in range(40, 48)))
```

	40	41	42	43	44	45	46	47
30	XYZ	XYZ	XYZ	XYZ	XYZ	XYZ	XYZ	XYZ
1;30	XYZ	XYZ	XYZ	XYZ	XYZ	XYZ	XYZ	XYZ
31	XYZ	XYZ	XYZ	XYZ	XYZ	XYZ	XYZ	XYZ
1;31	XYZ	XYZ	XYZ	XYZ	XYZ	XYZ	XYZ	XYZ
32	XYZ	XYZ	XYZ	XYZ	XYZ	XYZ	XYZ	XYZ
1;32	XYZ	XYZ	XYZ	XYZ	XYZ	XYZ	XYZ	XYZ
33	XYZ	XYZ	XYZ	XYZ	XYZ	XYZ	XYZ	XYZ
1;33	XYZ	XYZ	XYZ	XYZ	XYZ	XYZ	XYZ	XYZ
34	XYZ	XYZ	XYZ	XYZ	XYZ	XYZ	XYZ	XYZ
1;34	XYZ	XYZ	XYZ	XYZ	XYZ	XYZ	XYZ	XYZ
35	XYZ	XYZ	XYZ	XYZ	XYZ	XYZ	XYZ	XYZ
1;35	XYZ	XYZ	XYZ	XYZ	XYZ	XYZ	XYZ	XYZ
36	XYZ	XYZ	XYZ	XYZ	XYZ	XYZ	XYZ	XYZ
1;36	XYZ	XYZ	XYZ	XYZ	XYZ	XYZ	XYZ	XYZ
37	XYZ	XYZ	XYZ	XYZ	XYZ	XYZ	XYZ	XYZ
1;37	XYZ	XYZ	XYZ	XYZ	XYZ	XYZ	XYZ	XYZ

ANSI also supports a set of 256 indexed colors. The following code showing all of them is based on <http://bitmote.com/index.php?post/2012/11/19/Using-ANSI-Color-Codes-to-Colorize-Your-Bash-Prompt-on-Linux>.

```
In [24]: formatstring = '\x1b[38;5;{0};48;5;{0}mX\x1b[1mX\x1b[m'

        print(' + ' + ''.join(':{2}' .format(i) for i in range(36)))
        print(' 0 ' + ''.join(formatstring.format(i) for i in range(16)))
        for i in range(7):
            i = i * 36 + 16
```

⁹² https://en.wikipedia.org/wiki/ANSI_escape_code

```
print('{:3} '.format(i) + ' '.join(formatstring.format(i + j)
                                   for j in range(36) if i + j < 256))
+ 0 1 2 3 4 5 6 7 8 91011121314151617181920212223242526272829303132333435
0  X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X
16
52
88
124
160
196
232
```

You can even use 24-bit RGB colors:

```
In [25]: start = 255, 0, 0
        end = 0, 0, 255
        length = 79
        out = []

        for i in range(length):
            rgb = [start[c] + int(i * (end[c] - start[c]) / length) for c in range(3)]
            out.append('\x1b[
                '38;2;{rgb[2]};{rgb[1]};{rgb[0]};'
                '48;2;{rgb[0]};{rgb[1]};{rgb[2]}m\x1b[m'.format(rgb=rgb))
        print(' '.join(out))
```



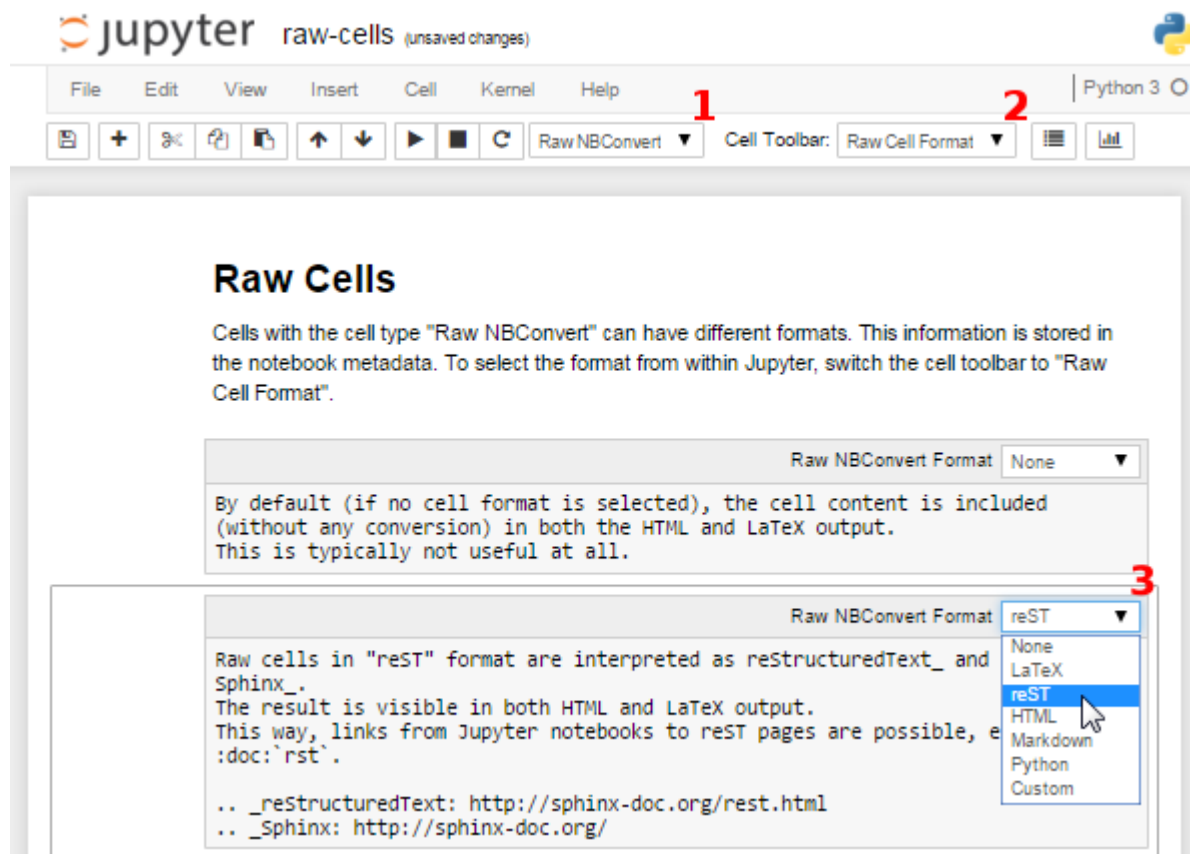
5 Raw Cells

The “Raw NBConvert” cell type can be used to render different code formats into HTML or LaTeX by Sphinx. This information is stored in the notebook metadata and converted appropriately.

5.1 Usage

To select a desired format from within Jupyter, select the cell containing your special code and choose options from the following dropdown menus:

1. Select “Raw NBConvert”
2. Switch the Cell Toolbar to “Raw Cell Format”
3. Chose the appropriate “Raw NBConvert Format” within the cell



5.2 Available Raw Cell Formats

The following examples show how different Jupyter cell formats are rendered by Sphinx.

5.2.1 None

By default (if no cell format is selected), the cell content is included (without any conversion) in both the HTML and LaTeX output. This is typically not useful at all.

"I'm a raw cell with no format."

5.2.2 reST

Raw cells in “reST” format are interpreted as reStructuredText and parsed by Sphinx. The result is visible in both HTML and LaTeX output.

“I’m a *raw cell* in reST⁹³ format.”

5.2.3 Markdown

Raw cells in “Markdown” format are interpreted as Markdown, and the result is included in both HTML and LaTeX output. Since the Jupyter Notebook also supports normal Markdown cells, this might not be useful *at all*.

“I’m a *raw cell* in Markdown⁹⁴ format.”

5.2.4 HTML

Raw cells in “HTML” format are only visible in HTML output. This option might not be very useful, since raw HTML code is also allowed within normal Markdown cells.

5.2.5 LaTeX

Raw cells in “LaTeX” format are only visible in LaTeX output.

I’m a *raw cell* in \LaTeX format.

5.2.6 Python

Raw cells in “Python” format are not visible at all (nor executed in any way).

End of doc/raw-cells.ipynb

The following section was generated from doc/hidden-cells.ipynb

6 Hidden Cells

You can remove cells from the HTML/LaTeX output by adding this to the cell metadata:

```
"nbsphinx": "hidden"
```

Hidden cells are still executed but removed afterwards.

For example, the following hidden cell defines the variable `answer`.

This is the cell after the hidden cell. Although the previous cell is not visible, its result is still available:

```
In [2]: answer
```

```
Out[2]: 42
```

⁹³ <http://sphinx-doc.org/rest.html>

⁹⁴ <https://daringfireball.net/projects/markdown/>

Don't overuse this, because it may make it harder to follow what's going on in your notebook.

Also Markdown cells can be hidden. The following cell is hidden.

This is the cell after the hidden cell.

End of doc/hidden-cells.ipynb

The following section was generated from doc/executing-notebooks.ipynb

7 Controlling Notebook Execution

Notebooks with no outputs are automatically executed during the Sphinx build process. If, however, there is at least one output cell present, the notebook is not evaluated and included as is.

The following notebooks show how this default behavior can be used and customized.

The following section was generated from doc/pre-executed.ipynb

7.1 Pre-Executing Notebooks

Automatically executing notebooks during the Sphinx build process is an important feature of `nbsphinx`. However, there are a few use cases where pre-executing a notebook and storing the outputs might be preferable. Storing any output will, by default, stop `nbsphinx` from executing the notebook.

7.1.1 Long-Running Cells

If you are doing some very time-consuming computations, it might not be feasible to re-execute the notebook every time you build your Sphinx documentation.

So just do it once – when you happen to have the time – and then just keep the output.

```
In [1]: import time
```

```
In [2]: %time time.sleep(60 * 60)
        6 * 7
```

```
CPU times: user 160 ms, sys: 56 ms, total: 216 ms
Wall time: 1h 1s
```

```
Out[2]: 42
```

If you *do* want to execute your notebooks, but some cells run for a long time, you can change the timeout, see [Cell Execution Timeout](#).

7.1.2 Rare Libraries

You might have created results with a library that's hard to install and therefore you have only managed to install it on one very old computer in the basement, so you probably cannot run this whenever you build your Sphinx docs.

```
In [3]: from a_very_rare_library import calculate_the_answer
```

```
In [4]: calculate_the_answer()
```

```
Out[4]: 42
```

7.1.3 Exceptions

If an exception is raised during the Sphinx build process, it is stopped (the build process, not the exception!). If you want to show to your audience how an exception looks like, you have two choices:

1. Allow errors – either generally or on a per-notebook basis – see *Ignoring Errors*.
2. Execute the notebook beforehand and save the results, like it's done in this example notebook:

```
In [5]: 1 / 0
```

```
-----  
ZeroDivisionError                                Traceback (most recent call last)  
<ipython-input-5-b710d87c980c> in <module>()  
----> 1 1 / 0
```

```
ZeroDivisionError: division by zero
```

End of doc/pre-executed.ipynb

The following section was generated from doc/never-execute.ipynb

7.2 Explicitly Dis-/Enabling Notebook Execution

If you want to include a notebook without outputs and yet don't want nbsphinx to execute it for you, you can explicitly disable this feature.

You can do this globally by setting the following option in `conf.py`:

```
nbsphinx_execute = 'never'
```

Or on a per-notebook basis by adding this to the notebook's JSON metadata:

```
"nbsphinx": {  
  "execute": "never"  
},
```

There are three possible settings, "always", "auto" and "never". By default (= "auto"), notebooks with no outputs are executed and notebooks with at least one output are not. As always, per-notebook settings take precedence over the settings in `conf.py`.

This very notebook has its metadata set to "never", therefore the following cell is not executed:

```
In [ ]: 6 * 7
```

End of doc/never-execute.ipynb

The following section was generated from doc/allow-errors.ipynb

7.3 Ignoring Errors

Normally, if an exception is raised while executing a notebook, the Sphinx build process is stopped immediately.

If a notebook contains errors on purpose (or if you are too lazy to fix them right now), you have three options:

1. Manually execute the notebook in question and save the results, see *the pre-executed example notebook*.

2. Allow errors in all notebooks by setting this option in `conf.py`:

```
nbsphinx_allow_errors = True
```

3. Allow errors on a per-notebook basis by adding this to the notebook's JSON metadata:

```
"nbsphinx": {  
  "allow_errors": true  
},
```

This very notebook is an example for the last option. The results of the following code cells are not stored within the notebook, therefore it is executed during the Sphinx build process. Since the above-mentioned `allow_errors` flag is set in this notebook's metadata, all cells are executed although most of them cause an exception.

```
In [1]: nonsense
```

```
-----  
NameError                                Traceback (most recent call last)  
<ipython-input-1-7dd4c0df649c> in <module>()  
----> 1 nonsense
```

```
NameError: name 'nonsense' is not defined
```

```
In [2]: 42 / 0
```

```
-----  
ZeroDivisionError                        Traceback (most recent call last)  
<ipython-input-2-52cebea8b64f> in <module>()  
----> 1 42 / 0
```

```
ZeroDivisionError: division by zero
```

```
In [3]: print 'Hello, world!'
```

```
File "<ipython-input-3-653b30cd70a8>", line 1  
  print 'Hello, world!'  
      ^
```

```
SyntaxError: Missing parentheses in call to 'print'
```

```
In [4]: 6 ~ 7
```

```
File "<ipython-input-4-8300b2622db3>", line 1  
  6 ~ 7  
    ^
```

```
SyntaxError: invalid syntax
```

```
In [5]: 6 * 7
```

```
Out[5]: 42
```

End of doc/allow-errors.ipynb

The following section was generated from `doc/timeout.ipynb`

7.4 Cell Execution Timeout

By default, `nbconvert` (which is used to execute the notebooks during the Sphinx build process) will give a cell 30 seconds to execute before it times out.

If you would like to change the amount of time given for a cell, you can change the timeout length for all notebooks by setting the following option in `conf.py`:

```
nbsphinx_timeout = 60
```

Or change the timeout length on a per-notebook basis by adding this to the notebook's JSON metadata:

```
"nbsphinx": {
  "timeout": 60
},
```

The timeout is given in seconds, use `-1` to disable the timeout.

Alternatively, you can manually execute the notebook in question and save the results, see *the pre-executed example notebook*.

End of `doc/timeout.ipynb`

End of `doc/executing-notebooks.ipynb`

The following section was generated from `doc/prolog-and-epilog.ipynb`

8 Prolog and Epilog

When including notebooks in your Sphinx documentation, you can choose to add some generic content before and after each notebook. This can be done with the configuration values `nbsphinx_prolog` and `nbsphinx_epilog` in the file `conf.py`.

The prolog and epilog strings can hold arbitrary `reST`⁹⁵ markup. Particularly, the `only`⁹⁶ and `raw`⁹⁷ directives can be used to have different content for HTML and LaTeX output.

Those strings are also processed by the `Jinja2`⁹⁸ templating engine. This means you can run Python-like code within those strings. You have access to the current `Sphinx build environment`⁹⁹ via the variable `env`. Most notably, you can get the file name of the current notebook with

```
{% env.doc2path(env.docname, base=None) %}
```

Have a look at the [Jinja2 template documentation](#)¹⁰⁰ for more information.

Warning:

If you use invalid syntax, you might get an error like this:

```
jinja2.exceptions.TemplateSyntaxError: expected token ':', got '}'
```

⁹⁵ <http://sphinx-doc.org/rest.html>

⁹⁶ <http://www.sphinx-doc.org/en/master/usage/restructuredtext/directives.html#directive-only>

⁹⁷ <https://docutils.readthedocs.io/en/sphinx-docs/ref/rst/directives.html#raw-data-pass-through>

⁹⁸ <http://jinja.pocoo.org/>

⁹⁹ <http://www.sphinx-doc.org/en/master/extdev/envapi.html>

¹⁰⁰ <http://jinja.pocoo.org/docs/latest/templates/>

This is especially prone to happen when using raw LaTeX, with its abundance of braces. To avoid clashing braces you can try to insert additional spaces or LaTeX macros that don't have a visible effect, like e.g. `\strut{}`. For example, you can avoid three consecutive opening braces with something like that:

```
\texttt{\strut}{\{ env.doc2path(env.docname, base=None) \}}
```

NB: The three consecutive closing braces in this example are not problematic.

8.1 Examples

You can include a simple static string, using `reST101` markup if you like:

```
nbsphinx_epilog = """
----

Generated by nbsphinx_ from a Jupyter_ notebook.

.. _nbsphinx: http://nbsphinx.readthedocs.io/
.. _Jupyter: https://jupyter.org/
"""
```

Using some additional Jinja2 markup and the information from the `env` variable, you can create URLs that point to the current notebook file, but located on some other server:

```
nbsphinx_prolog = """
Go there: https://example.org/notebooks/{ env.doc2path(env.docname, base=None) }

----
"""
```

You can also use separate content for HTML and LaTeX output, e.g.:

```
nbsphinx_prolog = """
{% set docname = env.doc2path(env.docname, base=None) %}

.. only:: html

    Go there: https://example.org/notebooks/{ docname }

.. only:: latex

    The following section was created from :file:`{ docname }`.
"""
```

For a more involved example for different HTML and LaTeX versions, see the file `conf.py` of the `nbsphinx` documentation.

End of doc/prolog-and-epilog.ipynb

¹⁰¹ <http://sphinx-doc.org/rest.html>

9 Notebooks in Sub-Directories

You can organize your notebooks in subdirectories and `nbsphinx` will take care that relative links to other notebooks, images and other files still work.



Let's see if links to local images work:

```
In [1]: from IPython.display import Image
        Image(filename='../images/notebook_icon.png')
```



Warning:

There may be problems with images in output cells if your source directory contains symbolic links, see [issue #49¹⁰²](#).

A link to a notebook in the same sub-directory: [link](#).

A link to a notebook in the parent directory: [link](#).

A link to a local file: [link](#).

9.1 A Sub-Section

This is just for testing inter-notebook links, see [this section](#).

End of doc/subdir/a-notebook-in-a-subdir.ipynb

¹⁰² <https://github.com/spatialaudio/nbsphinx/issues/49>

10 Using toctree In A Notebook

In Sphinx-based documentation, there is typically a file called `index.rst` which contains one or more `toctree`¹⁰³ directives. Those can be used to pull in further source files (which themselves can contain `toctree` directives).

With `nbsphinx` it is possible to get a similar effect within a Jupyter notebook using the `"nbsphinx-toctree"` cell metadata. Markdown cells with `"nbsphinx-toctree"` metadata are not converted like “normal” Markdown cells. Instead, they are only scanned for links to other notebooks (or `*.rst` files and other Sphinx source files) and those links are added to a `toctree` directive. External links can also be used, but they will not be visible in the LaTeX output.

If there is a section title in the cell, it is used as `toctree` caption (but it also works without a title).

Note:

All other content of such a cell is *ignored!*

Use ...

```
"nbsphinx-toctree": {}
```

... for the default settings, ...

```
"nbsphinx-toctree": {
  "maxdepth": 2
}
```

... for setting the `:maxdepth:` option, or...

```
"nbsphinx-toctree": {
  "hidden": true
}
```

... for setting the `:hidden:` option.

Of course, multiple options can be used at the same time, e.g.

```
"nbsphinx-toctree": {
  "maxdepth": 3,
  "numbered": true
}
```

For more options, have a look at the [Sphinx documentation](#)¹⁰⁴. All options can be used – except `:glob:`, which can only be used in *rst files* and in *raw reST cells*.

Note that in the HTML output, a `toctree` cell generates an in-line table of contents (containing links) at its position in the notebook, whereas in the LaTeX output, a new (sub-)section with the actual content is inserted at its position. All content below the `toctree` cell will appear after the table of contents/inserted section, respectively. If you want to use the LaTeX output, it is recommended that you don't add further cells below a `toctree` cell, otherwise their content may appear at unexpected places. Multiple `toctree` cells in a row should be fine, though.

The following cell is tagged with `"nbsphinx-toctree"` metadata and contains a link to the notebook [yet-another.ipynb](#) and an external link (which will only be visible in the HTML output). It also contains a section title which will be used as `toctree` caption.

¹⁰³ <http://www.sphinx-doc.org/en/master/usage/restructuredtext/directives.html#directive-toctree>

¹⁰⁴ <http://www.sphinx-doc.org/en/master/usage/restructuredtext/directives.html#directive-toctree>

The following section was generated from `doc/yet-another.ipynb`

10.1 Yet Another Notebook

This notebook is only here to show how (sub-)toctrees can be created with Markdown cell metadata. See *there*.

End of `doc/yet-another.ipynb`

End of `doc/subdir/toctree.ipynb`

11 Normal reStructuredText Files

This is a normal RST file.

Note: Those still work!

11.1 Links to Notebooks (and Other Sphinx Source Files)

Links to Sphinx source files can be created like normal [Sphinx hyperlinks](#)¹⁰⁵, just using a relative path to the local file: *link*.

```
using a relative path to the local file: link_.  
.. \_link: subdir/a-notebook-in-a-subdir.ipynb
```

If the link text has a space (or some other strange character) in it, you have to surround it with backticks: *a notebook link*.

```
surround it with backticks: `a notebook link`_.  
.. \_a notebook link: subdir/a-notebook-in-a-subdir.ipynb
```

You can also use an [anonymous hyperlink target](#)¹⁰⁶, like this: *link*. If you have multiple of those, their order matters!

```
like this: link___.  
__ subdir/a-notebook-in-a-subdir.ipynb
```

Finally, you can use [Embedded URIs](#)¹⁰⁷, like this *link*.

```
like this `link <subdir/a-notebook-in-a-subdir.ipynb>`_.
```

Note: These links should also work on Github and in other rendered reStructuredText pages.

Links to subsections are also possible by adding a hash sign (#) and the section title to any of the above-mentioned link variants. You have to replace spaces in the section titles by hyphens. For example, see this *subsection*.

```
For example, see this subsection_.  
.. \_subsection: subdir/a-notebook-in-a-subdir.ipynb#A-Sub-Section
```

11.2 Links to Notebooks, Ye Olde Way

In addition to the way shown above, you can also create links to notebooks (and other Sphinx source files) with `:ref:`¹⁰⁸. This has some disadvantages:

- It is arguably a bit more clunky.

¹⁰⁵ <http://www.sphinx-doc.org/en/stable/rest.html#external-links>

¹⁰⁶ <http://docutils.sourceforge.net/docs/ref/rst/restructuredtext.html#anonymous-hyperlinks>

¹⁰⁷ <http://docutils.sourceforge.net/docs/ref/rst/restructuredtext.html#embedded-uris-and-aliases>

¹⁰⁸ <http://www.sphinx-doc.org/en/stable/markup/inline.html#role-ref>

- Because `:ref:` is a Sphinx feature, the links don't work on Github and other rendered reStructuredText pages that use plain old docutils.

It also has one important advantage:

- The link text can automatically be taken from the actual section title.

A link with automatic title looks like this: *Notebooks in Sub-Directories*.

```
:ref:`/subdir/a-notebook-in-a-subdir.ipynb`
```

But you can also provide *your own link title*.

```
:ref:`your own link title </subdir/a-notebook-in-a-subdir.ipynb>`
```

However, if you want to use your own title, you are probably better off using the method described above in *Links to Notebooks (and Other Sphinx Source Files)*.

Links to subsections are also possible, e.g. *A Sub-Section* (the subsection title is used as link text) and *alternative text*.

These links were created with:

```
:ref:`/subdir/a-notebook-in-a-subdir.ipynb#A-Sub-Section`  
:ref:`alternative text </subdir/a-notebook-in-a-subdir.ipynb#A-Sub-Section>`
```

Note:

- The paths have to be relative to the top source directory and they have to start with a slash (/).
 - Spaces in the section title have to be replaced by hyphens!
-

11.3 Sphinx Directives for Info/Warning Boxes

Warning:

This is an experimental feature! Its usage may change in the future or it might disappear completely, so don't use it for now.

With a bit of luck, it will be possible (some time in the future) to create info/warning boxes in Markdown cells, see <https://github.com/jupyter/notebook/issues/1292>. If this ever happens, `nbsphinx` will provide directives for creating such boxes. For now, there are two directives available: `nbinfo` and `nbwarning`. This is how an info box looks like:

Note:

This is an info box.

It may include nested formatting, even another info/warning box:

Warning: You should probably not use nested boxes!

11.4 Domain Objects

`example_python_function(foo)`

This is just for testing domain object links. See [this section](#).

Parameters `foo (str)` – Example string parameter

12 External Links

notebook_sphinxext.py

Notebooks can be included in `*.rst` files with a custom notebook directive. Uses `runipy` to execute notebooks and `nbconvert` to convert the result to HTML.

No LaTeX support.

<https://github.com/ngoldbaum/RunNotebook>

https://bitbucket.org/yt_analysis/yt-doc/src/default/extensions/notebook_sphinxext.py

https://github.com/matthew-brett/perrin-academy/blob/master/sphinxext/notebook_sphinxext.py

nb2plots

Notebook to reStructuredText converter which uses a modified version of the matplotlib `plot` directive.

<https://github.com/matthew-brett/nb2plots>

brole

A Sphinx role for IPython notebooks

<https://github.com/matthew-brett/brole>

Sphinx-Gallery

<http://sphinx-gallery.readthedocs.io/>

sphinx-nbexamples

<http://sphinx-nbexamples.readthedocs.io/>

<https://github.com/Chilipp/sphinx-nbexamples>

nbsphinx-link

<https://github.com/vidartf/nbsphinx-link>

Uses `nbsphinx`, but supports notebooks outside the Sphinx source directory.

See <https://github.com/spatialaudio/nbsphinx/pull/33> for some limitations.

DocOnce

<http://hplgit.github.io/doconce/doc/web/index.html>

Converting Notebooks to reStructuredText

https://github.com/perrette/dimarray/blob/master/docs/scripts/nbconvert_to_rst.py

<https://gist.github.com/hadim/16e29b5848672e2e497c>

<http://sphinx-ipyynb.readthedocs.io/>

Converting Notebooks to HTML for Blog Posts

http://dongweiming.github.io/divingintoipyynb_nikola/posts/nbconvert.html

https://github.com/getpelican/pelican-plugins/blob/master/liquid_tags/notebook.py

Further Posts and Issues

<https://github.com/ipython/ipython/issues/4936>

<https://mail.scipy.org/pipermail/ipython-user/2013-December/013490.html> (not available anymore)

End of doc/links.ipynb