nbsphinx

Release 0.9.3-3-g7031e97

Matthias Geier

2016-05-22

Contents

1 Installation **5**
- 1.1 `nbsphinx` Packages . . . 5
- 1.2 `nbsphinx` Prerequisites . . . 5
  - 1.2.1 Python . . . 5
  - 1.2.2 Sphinx . . . 6
  - 1.2.3 pip . . . 6
  - 1.2.4 pandoc . . . 6
  - 1.2.5 Pygments Lexer for Syntax Highlighting . . . 6
  - 1.2.6 Jupyter Kernel . . . 7

2 Usage **7**
- 2.1 Project Setup . . . 7
- 2.2 Running Sphinx . . . 8
- 2.3 Watching for Changes with `sphinx-autobuild` . . . 9
- 2.4 Automatic Creation of HTML and PDF output on readthedocs.org . . . 9
  - 2.4.1 Using `requirements.txt` . . . 10
  - 2.4.2 Using `conda` . . . 10
- 2.5 Automatic Creation of HTML and PDF output on GitLab Pages . . . 11
- 2.6 HTML Themes . . . 11
  - 2.6.1 Sphinx's Built-In Themes . . . 12
  - 2.6.2 3rd-Party Themes . . . 12
- 2.7 Using Notebooks with Git . . . 14

3 Configuration **15**
- 3.1 Sphinx Configuration Values . . . 15
  - 3.1.1 `exclude_patterns` . . . 15
  - 3.1.2 `extensions` . . . 15
  - 3.1.3 `html_css_files` . . . 15
  - 3.1.4 `html_sourcelink_suffix` . . . 16
  - 3.1.5 `latex_additional_files` . . . 16
  - 3.1.6 `mathjax3_config` . . . 16
  - 3.1.7 `pygments_style` . . . 16
  - 3.1.8 `suppress_warnings` . . . 16
- 3.2 `nbsphinx` Configuration Values . . . 17
  - 3.2.1 `nbsphinx_allow_errors` . . . 17
  - 3.2.2 `nbsphinx_assume_equations` . . . 17
  - 3.2.3 `nbsphinx_codecell_lexer` . . . 17
  - 3.2.4 `nbsphinx_custom_formats` . . . 17

3.2.5 nbsphinx_epilog . . . 17
3.2.6 nbsphinx_execute . . . 17
3.2.7 nbsphinx_execute_arguments . . . 17
3.2.8 nbsphinx_input_prompt . . . 18
3.2.9 nbsphinx_kernel_name . . . 18
3.2.10 nbsphinx_output_prompt . . . 18
3.2.11 nbsphinx_prolog . . . 18
3.2.12 nbsphinx_prompt_width . . . 18
3.2.13 nbsphinx_requirejs_options . . . 18
3.2.14 nbsphinx_requirejs_path . . . 18
3.2.15 nbsphinx_responsive_width . . . 19
3.2.16 nbsphinx_thumbnails . . . 19
3.2.17 nbsphinx_timeout . . . 19
3.2.18 nbsphinx_widgets_options . . . 19
3.2.19 nbsphinx_widgets_path . . . 19

4 Markdown Cells **19**
4.1 Equations . . . 20
4.1.1 Automatic Equation Numbering . . . 21
4.1.2 Manual Equation Numbering . . . 21
4.2 Citations . . . 22
4.2.1 Footnote citations . . . 22
4.3 Code . . . 23
4.4 Tables . . . 23
4.5 Images . . . 23
4.5.1 Using the HTML <img> tag . . . 24
4.5.2 SVG support for LaTeX . . . 24
4.6 Cell Attachments . . . 25
4.7 HTML Elements (HTML only) . . . 25
4.8 Info/Warning Boxes . . . 26
4.9 Links to Other Notebooks . . . 27
4.10 Links to *.rst Files (and Other Sphinx Source Files) . . . 28
4.11 Links to Local Files . . . 28
4.12 Links to Domain Objects . . . 28

5 Code Cells **29**
5.1 Code, Output, Streams . . . 29
5.2 Special Display Formats . . . 30
5.2.1 Local Image Files . . . 30
5.2.2 Image URLs . . . 31
5.2.3 Math . . . 31
5.2.4 Plots . . . 32
5.2.5 Pandas Dataframes . . . 34
5.2.6 Markdown Content . . . 34
5.2.6.1 Markdown . . . 35
5.2.7 YouTube Videos . . . 35
5.2.8 Interactive Widgets (HTML only) . . . 36
5.2.8.1 Troubleshooting . . . 37
5.2.9 Arbitrary JavaScript Output (HTML only) . . . 38
5.2.10 Unsupported Output Types . . . 38
5.3 ANSI Colors . . . 38

6 Raw Cells **40**
6.1 Usage . . . 40
6.1.1 Jupyter Notebook . . . 40
6.1.2 JupyterLab . . . 41

6.2 Available Raw Cell Formats . . . 42
6.2.1 None . . . 42
6.2.2 reST . . . 42
6.2.3 Markdown . . . 43
6.2.4 HTML . . . 43
6.2.5 LaTeX . . . 43
6.2.6 Python . . . 43

7 Hidden Cells **43**

8 Controlling Notebook Execution **44**
8.1 Pre-Executing Notebooks . . . 44
8.1.1 Long-Running Cells . . . 44
8.1.2 Rare Libraries . . . 44
8.1.3 Exceptions . . . 45
8.1.4 Client-specific Outputs . . . 45
8.1.5 Interactive Input . . . 45
8.2 Explicitly Dis-/Enabling Notebook Execution . . . 46
8.3 Ignoring Errors . . . 46
8.4 Ignoring Errors on a Per-Cell Basis . . . 47
8.5 Configuring the Kernels . . . 48
8.5.1 Kernel Name . . . 48
8.5.2 Kernel Arguments . . . 48
8.5.3 Environment Variables . . . 48
8.6 Cell Execution Timeout . . . 49

9 Prolog and Epilog **49**
9.1 Examples . . . 50

10 Custom Notebook Formats **51**
10.1 Example: Jupytext . . . 51

11 Notebooks in Sub-Directories **52**
11.1 A Sub-Section . . . 53
11.2 That's a "Strange" Section . . . 53

12 Creating Thumbnail Galleries **53**
12.1 Gallery With Nested Documents . . . 53
12.1.1 Default Thumbnail . . . 54
12.1.2 Using a Cell Tag to Select a Thumbnail . . . 55
12.1.3 Using Cell Metadata to Select a Thumbnail . . . 56
12.1.4 Choosing from Multiple Outputs . . . 57
12.1.5 A Notebook without Thumbnail . . . 58
12.1.6 Specifying Thumbnails in `conf.py` . . . 58
12.2 Gallery With Links (HTML only) . . . 60

13 Using `toctree` In A Notebook **60**
13.1 Yet Another Notebook . . . 61

14 Custom CSS **61**
14.1 For All Pages . . . 62
14.2 For All RST files . . . 62
14.3 For All Notebooks . . . 62
14.4 For a Single Notebook . . . 63

15 Normal reStructuredText Files **63**
15.1 Links to Notebooks (and Other Sphinx Source Files) . . . 63

15.2 Links to Notebooks, Ye Olde Way . . . 64
15.3 Sphinx Directives for Info/Warning Boxes . . . 65
15.4 Domain Objects . . . 65
15.5 References . . . 65
15.5.1 Citations . . . 66
15.5.2 Footnote citations . . . 66
15.6 Thumbnail Link Galleries (HTML only) . . . 67
15.7 Thumbnail Galleries . . . 67
15.7.1 Dummy Notebook 1 for Gallery . . . 67
15.7.2 Dummy Notebook 2 for Gallery . . . 67

16 Using Markdown Files **68**
16.1 Links to Notebooks (and Other Sphinx Source Files) . . . 69
16.2 Math . . . 69
16.3 Tables . . . 69
16.4 Images . . . 69

17 External Links **69**

18 Contributing **72**
18.1 Development Installation . . . 72
18.2 Building the Documentation . . . 72
18.3 Building Themes . . . 73
18.4 Testing . . . 73

19 References **74**

20 Version History **74**

`nbsphinx` is a Sphinx[1] extension that provides a source parser for `*.ipynb` files. Custom Sphinx directives are used to show Jupyter Notebook[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`
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):
`https://nbsphinx.readthedocs.io/`

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

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

All of the following content was generated from Jupyter notebooks, except for the sections *Normal reStructuredText Files* (page 63), *Contributing* (page 72), *References* (page 74) and *Version History* (page 74), which were generated from Sphinx's built-in reStructuredText[3] format. The sections *Custom Notebook*

[1] `https://www.sphinx-doc.org/`
[2] `https://jupyter.org/`
[3] `https://www.sphinx-doc.org/en/master/usage/restructuredtext/basics.html`

Formats (page 51), *Dummy Notebook 2 for Gallery* (page 67) and *Using Markdown Files* (page 68) are using alternative storage formats for Jupyter notebooks, see *Custom Notebook Formats* (page 51) for details.

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 PyPI[4] (using `pip`). If you want to try the latest development version, have a look at the section *Contributing* (page 72).

1.1 `nbsphinx` Packages

[5]

If you are using the `conda` package manager (e.g. with Miniforge[6], Miniconda[7] or Anaconda[8]), you can install `nbsphinx` from the conda-forge[9] channel:

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

[10]

You can of course also install `nbsphinx` with `pip`, Python's own package manager:

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

Depending on your Python installation, you may have to use `python` instead of `python3`. If you have installed the module already, you can use the `--upgrade` flag to get the newest release.

There are more packages available. For an overview, see repology[11].

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 Miniforge/Mambaforge[12].

[4] https://pypi.org/project/nbsphinx
[5] https://anaconda.org/conda-forge/nbsphinx
[6] https://github.com/conda-forge/miniforge
[7] https://docs.conda.io/en/latest/miniconda.html
[8] https://www.anaconda.com/download#downloads
[9] https://conda-forge.org/
[10] https://pypi.org/project/nbsphinx
[11] https://repology.org/project/python:nbsphinx/versions
[12] https://github.com/conda-forge/miniforge

1.2.2 Sphinx

You'll need Sphinx[13] 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[14]:

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

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

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

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[15].

1.2.4 pandoc

The stand-alone program pandoc[16] 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[17]:

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

If that doesn't work out for you, have a look at `pandoc`'s installation instructions[18].

Note

The use of `pandoc` in `nbsphinx` is temporary, but will likely stay that way for a long time, see issue #36[19].

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).

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
```

[13] https://www.sphinx-doc.org/
[14] https://anaconda.org/conda-forge/sphinx
[15] https://pip.pypa.io/en/latest/installing/
[16] https://pandoc.org/
[17] https://anaconda.org/conda-forge/pandoc
[18] https://pandoc.org/installing.html
[19] https://github.com/spatialaudio/nbsphinx/issues/36

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[20] and nbsphinx issue #24[21].

1.2.6 Jupyter Kernel

If you want to execute your notebooks during the Sphinx build process (see *Controlling Notebook Execution* (page 44)), you need an appropriate Jupyter kernel[22] 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
```

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

Note

If your *automatic builds* (page 9) on `https://readthedocs.org` are failing due to an error like the one below, add `ipykernel` to `docs/requirements.txt` or `doc/environment.yml` to resolve.

```
jupyter_client.kernelspec.nosuchkernel: no such kernel named python3
```

`doc/installation.ipynb` ends here.

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

2 Usage

2.1 Project Setup

In the directory with your notebook files, run this command (assuming you have Sphinx[23] and `nbsphinx` installed already):

```
python3 -m sphinx.cmd.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 make sure that the `extensions` setting at least contains `'nbsphinx'` (but you might want to add other useful extensions as well):

[20] https://github.com/ContinuumIO/anaconda-issues/issues/1430
[21] https://github.com/spatialaudio/nbsphinx/issues/24
[22] https://jupyter.readthedocs.io/en/latest/projects/kernels.html
[23] https://www.sphinx-doc.org/

```
extensions = [
    'nbsphinx',
]
```

For an example, see this project's `conf.py` file.

Once your `conf.py` is in place, edit the file named `index.rst` and add the file names of your notebooks (without the `.ipynb` extension) to the toctree[24] directive. For an example, see this project's `doc/index.rst` file.

Alternatively, you can delete the file `index.rst` and replace it with your own notebook called `index.ipynb` which will serve as main page. In this case you can create the main *toctree* (page 60) in `index.ipynb`.

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* (page 44).

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
```

[24] https://www.sphinx-doc.org/en/master/usage/restructuredtext/directives.html#directive-toctree
[25] http://personal.psu.edu/jcc8//software/latexmk-jcc/
[26] https://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
```

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 automagically. 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* (page 8)). 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.org

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 GitLab/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 automagically 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[28] which of those should be created.

Note

If you want to execute notebooks (see *Controlling Notebook Execution* (page 44)), you'll need to install the appropriate Jupyter kernel. In the examples below, the IPython kernel is installed from the packet `ipykernel`.

[27] `https://pypi.org/project/sphinx-autobuild`
[28] `https://readthedocs.org/dashboard/`

2.4.1 Using `requirements.txt`

1. Create a file named `.readthedocs.yml` in the main directory of your repository with the following contents:

```yaml
version: 2
build:
  os: ubuntu-22.04
  tools:
    python: "3"
python:
  install:
    - requirements: doc/requirements.txt
```

   For further options see https://docs.readthedocs.io/en/latest/config-file/.

2. Create a file named `doc/requirements.txt` (or whatever you chose in the previous step) containing the required `pip` packages:

```
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
```

2.4.2 Using `conda`

1. Create a file named `.readthedocs.yml` in the main directory of your repository with the following contents:

```yaml
version: 2
formats: all
conda:
  environment: doc/environment.yml
```

   For further options see https://docs.readthedocs.io/en/latest/config-file/.

2. Create a file named `doc/environment.yml` (or whatever you chose in the previous step) describing a conda environment[29] like this:

```yaml
channels:
  - conda-forge
dependencies:
  - python>=3
  - pandoc
  - ipykernel
  - pip
  - 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.

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

[29] `https://docs.conda.io/projects/conda/en/latest/user-guide/tasks/manage-environments.html`

Note

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

2.5 Automatic Creation of HTML and PDF output on GitLab Pages

When using GitLab pages, you can use nbsphinx by adding the file `.gitlab-ci.yml` to your repo and copying the following lines into this file:

```
image: python:3-slim

variables:
  PIP: python3 -m pip
  SPHINX: python3 -m sphinx -W --keep-going --color

build-docs:
  stage: build
  script:
    - apt-get update -y
    - apt-get install -y --no-install-recommends pandoc
    - $PIP install -r doc/requirements.txt
    - $SPHINX -d doctrees doc html -b html
    - $SPHINX -d doctrees doc linkcheck -b linkcheck
  artifacts:
    when: always
    paths:
      - html
      - linkcheck/output.*

pages:
  stage: deploy
  variables:
    GIT_STRATEGY: none
  script:
    - mv html public
  artifacts:
    paths:
      - public
  rules:
    - if: $CI_COMMIT_REF_NAME == $CI_DEFAULT_BRANCH
```

2.6 HTML Themes

The `nbsphinx` extension does *not* provide its own theme, you can use any of the available themes or create a custom one[30], 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[31])
2. How the `nbsphinx` documentation looks when using this theme

[30] `https://www.sphinx-doc.org/en/master/development/theming.html#creating-themes`
[31] `https://www.sphinx-doc.org/en/master/usage/theming.html#builtin-themes`

3. How to enable this theme using either `requirements.txt` or `readthedocs.yml` and theme-specific settings (in some cases)

2.6.1 Sphinx's Built-In Themes

- `agogo`: example[32], usage[33]
- alabaster[34]: example[35], usage[36]
- `bizstyle`: example[37], usage[38]
- `classic`: example[39], usage[40]
- `haiku`: example[41], usage[42]
- `nature`: example[43], usage[44]
- `pyramid`: example[45], usage[46]
- `scrolls`: example[47], usage[48]
- `sphinxdoc`: example[49], usage[50]
- `traditional`: example[51], usage[52]

2.6.2 3rd-Party Themes

- basicstrap[53]: example[54], usage[55]
- better[56]: example[57], usage[58]
- bootstrap[59]: example[60], usage[61]
- bootstrap-astropy[62]: example[63], usage[64]

[32] https://nbsphinx.readthedocs.io/en/agogo-theme/
[33] https://github.com/spatialaudio/nbsphinx/compare/agogo-theme%5E...agogo-theme
[34] https://alabaster.readthedocs.io/
[35] https://nbsphinx.readthedocs.io/en/alabaster-theme/
[36] https://github.com/spatialaudio/nbsphinx/compare/alabaster-theme%5E...alabaster-theme
[37] https://nbsphinx.readthedocs.io/en/bizstyle-theme/
[38] https://github.com/spatialaudio/nbsphinx/compare/bizstyle-theme%5E...bizstyle-theme
[39] https://nbsphinx.readthedocs.io/en/classic-theme/
[40] https://github.com/spatialaudio/nbsphinx/compare/classic-theme%5E...classic-theme
[41] https://nbsphinx.readthedocs.io/en/haiku-theme/
[42] https://github.com/spatialaudio/nbsphinx/compare/haiku-theme%5E...haiku-theme
[43] https://nbsphinx.readthedocs.io/en/nature-theme/
[44] https://github.com/spatialaudio/nbsphinx/compare/nature-theme%5E...nature-theme
[45] https://nbsphinx.readthedocs.io/en/pyramid-theme/
[46] https://github.com/spatialaudio/nbsphinx/compare/pyramid-theme%5E...pyramid-theme
[47] https://nbsphinx.readthedocs.io/en/scrolls-theme/
[48] https://github.com/spatialaudio/nbsphinx/compare/scrolls-theme%5E...scrolls-theme
[49] https://nbsphinx.readthedocs.io/en/sphinxdoc-theme/
[50] https://github.com/spatialaudio/nbsphinx/compare/sphinxdoc-theme%5E...sphinxdoc-theme
[51] https://nbsphinx.readthedocs.io/en/traditional-theme/
[52] https://github.com/spatialaudio/nbsphinx/compare/traditional-theme%5E...traditional-theme
[53] https://pythonhosted.org/sphinxjp.themes.basicstrap/
[54] https://nbsphinx.readthedocs.io/en/basicstrap-theme/
[55] https://github.com/spatialaudio/nbsphinx/compare/basicstrap-theme%5E...basicstrap-theme
[56] https://sphinx-better-theme.readthedocs.io/
[57] https://nbsphinx.readthedocs.io/en/better-theme/
[58] https://github.com/spatialaudio/nbsphinx/compare/better-theme%5E...better-theme
[59] https://sphinx-bootstrap-theme.readthedocs.io/
[60] https://nbsphinx.readthedocs.io/en/bootstrap-theme/
[61] https://github.com/spatialaudio/nbsphinx/compare/bootstrap-theme%5E...bootstrap-theme
[62] https://github.com/astropy/astropy-sphinx-theme
[63] https://nbsphinx.readthedocs.io/en/astropy-theme/
[64] https://github.com/spatialaudio/nbsphinx/compare/astropy-theme%5E...astropy-theme

- cloud/redcloud/greencloud[65]: example[66], usage[67]
- dask_sphinx_theme[68]: example[69], usage[70]
- furo[71]: example[72], usage[73]
- guzzle_sphinx_theme[74]: example[75], usage[76]
- insipid[77]: example[78], usage[79]
- maisie_sphinx_theme[80]: example[81], usage[82]
- pangeo[83]: example[84], usage[85]
- pydata_sphinx_theme[86]: example[87], usage[88]
- python_docs_theme[89]: example[90], usage[91]
- pytorch_sphinx_theme[92]: example[93], usage[94]
- sizzle[95]: example[96], usage[97]
- sphinx_book_theme[98]: example[99], usage[100]
- sphinx_holoviz_theme[101]: example[102], usage[103]
- sphinx_immaterial[104]: example[105], usage[106]

[65] https://cloud-sptheme.readthedocs.io/
[66] https://nbsphinx.readthedocs.io/en/cloud-theme/
[67] https://github.com/spatialaudio/nbsphinx/compare/cloud-theme%5E...cloud-theme
[68] https://github.com/dask/dask-sphinx-theme
[69] https://nbsphinx.readthedocs.io/en/dask-theme/
[70] https://github.com/spatialaudio/nbsphinx/compare/dask-theme%5E...dask-theme
[71] https://github.com/pradyunsg/furo
[72] https://nbsphinx.readthedocs.io/en/furo-theme/
[73] https://github.com/spatialaudio/nbsphinx/compare/furo-theme%5E...furo-theme
[74] https://github.com/guzzle/guzzle_sphinx_theme
[75] https://nbsphinx.readthedocs.io/en/guzzle-theme/
[76] https://github.com/spatialaudio/nbsphinx/compare/guzzle-theme%5E...guzzle-theme
[77] https://github.com/mgeier/insipid-sphinx-theme/
[78] https://nbsphinx.readthedocs.io/en/insipid-theme/
[79] https://github.com/spatialaudio/nbsphinx/compare/insipid-theme%5E...insipid-theme
[80] https://github.com/maisie-dev/maisie-sphinx-theme
[81] https://nbsphinx.readthedocs.io/en/maisie-theme/
[82] https://github.com/spatialaudio/nbsphinx/compare/maisie-theme%5E...maisie-theme
[83] https://github.com/pangeo-data/sphinx_pangeo_theme/
[84] https://nbsphinx.readthedocs.io/en/pangeo-theme/
[85] https://github.com/spatialaudio/nbsphinx/compare/pangeo-theme%5E...pangeo-theme
[86] https://pydata-sphinx-theme.readthedocs.io/
[87] https://nbsphinx.readthedocs.io/en/pydata-theme/
[88] https://github.com/spatialaudio/nbsphinx/compare/pydata-theme%5E...pydata-theme
[89] https://github.com/python/python-docs-theme
[90] https://nbsphinx.readthedocs.io/en/python-docs-theme/
[91] https://github.com/spatialaudio/nbsphinx/compare/python-docs-theme%5E...python-docs-theme
[92] https://github.com/shiftlab/pytorch_sphinx_theme
[93] https://nbsphinx.readthedocs.io/en/pytorch-theme/
[94] https://github.com/spatialaudio/nbsphinx/compare/pytorch-theme%5E...pytorch-theme
[95] https://docs.red-dove.com/sphinx_sizzle_theme/
[96] https://nbsphinx.readthedocs.io/en/sizzle-theme/
[97] https://github.com/spatialaudio/nbsphinx/compare/sizzle-theme%5E...sizzle-theme
[98] https://sphinx-book-theme.readthedocs.io/
[99] https://nbsphinx.readthedocs.io/en/sphinx-book-theme/
[100] https://github.com/spatialaudio/nbsphinx/compare/sphinx-book-theme%5E...sphinx-book-theme
[101] https://github.com/pyviz-dev/sphinx_holoviz_theme
[102] https://nbsphinx.readthedocs.io/en/holoviz-theme/
[103] https://github.com/spatialaudio/nbsphinx/compare/holoviz-theme%5E...holoviz-theme
[104] https://github.com/jbms/sphinx-immaterial
[105] https://nbsphinx.readthedocs.io/en/immaterial-theme/
[106] https://github.com/spatialaudio/nbsphinx/compare/immaterial-theme%5E...immaterial-theme

- sphinx_material[107]: example[108], usage[109]
- sphinx_py3doc_enhanced_theme[110]: example[111], usage[112]
- sphinx_rtd_theme[113]: example[114], usage[115]
- typlog[116]: example[117], usage[118]

If you know of another Sphinx theme that should be included here, please open an issue on Github[119]. An overview of many more themes can be found at https://sphinx-themes.org/.

2.7 Using Notebooks with Git

Git[120] 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* (page 44)).

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* (page 44) for how this behavior can be customized.

In the Jupyter Notebook application, you can manually clear all outputs by selecting "Cell" → "All Output" → "Clear" from the menu. In JupyterLab, the menu items are "Edit" → "Clear All Outputs".

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
- https://tillahoffmann.github.io/2017/04/17/versioning-jupyter-notebooks-with-git.html
- http://timstaley.co.uk/posts/making-git-and-jupyter-notebooks-play-nice/
- https://web.archive.org/web/20191003081426/https://pascalbugnion.net/blog/ipython-notebooks-and-git.html
- https://github.com/choldgraf/nbclean
- https://jamesfolberth.org/articles/2017/08/07/git-commit-hook-for-jupyter-notebooks/

[107] https://github.com/bashtage/sphinx-material
[108] https://nbsphinx.readthedocs.io/en/material-theme/
[109] https://github.com/spatialaudio/nbsphinx/compare/material-theme%5E...material-theme
[110] https://github.com/ionelmc/sphinx-py3doc-enhanced-theme
[111] https://nbsphinx.readthedocs.io/en/py3doc-enhanced-theme/
[112] https://github.com/spatialaudio/nbsphinx/compare/py3doc-enhanced-theme%5E...py3doc-enhanced-theme
[113] https://github.com/readthedocs/sphinx_rtd_theme
[114] https://nbsphinx.readthedocs.io/en/rtd-theme/
[115] https://github.com/spatialaudio/nbsphinx/compare/rtd-theme%5E...rtd-theme
[116] https://github.com/typlog/sphinx-typlog-theme
[117] https://nbsphinx.readthedocs.io/en/typlog-theme/
[118] https://github.com/spatialaudio/nbsphinx/compare/typlog-theme%5E...typlog-theme
[119] https://github.com/spatialaudio/nbsphinx/issues
[120] https://git-scm.com/

- https://github.com/ResearchSoftwareActions/EnsureCleanNotebooksAction

doc/usage.ipynb ends here.

The following section was generated from doc/configuration.ipynb

3 Configuration

The following configuration values can be used in the `conf.py` file, see *Project Setup* (page 7).

3.1 Sphinx Configuration Values

All configuration values are described in the Sphinx documentation[121], here we mention only the ones which may be relevant in combination with `nbsphinx`.

3.1.1 exclude_patterns

Sphinx builds all potential source files (reST files, Jupyter notebooks, ...) that are in the source directory (including any sub-directories), whether you want to use them or not. If you want certain source files not to be built, specify them in exclude_patterns[122]. For example, you might want to ignore source files in your build directory:

```
exclude_patterns = ['_build']
```

Note that the directory `.ipynb_checkpoints` is automatically added to `exclude_patterns` by `nbsphinx`.

3.1.2 extensions

This is the only required value. You have to add `'nbsphinx'` to the list of extensions[123], otherwise it won't work.

Other interesting extensions are:

- `'sphinx.ext.mathjax'` (Sphinx loads this by default) for *math formulas* (page 20)
- `'sphinxcontrib.bibtex'` for *bibliographic references* (page 65)
- `'sphinxcontrib.rsvgconverter'` for *SVG->PDF conversion in LaTeX output* (page 24)
- `'sphinx_copybutton'` for adding "copy to clipboard" buttons[124] to all text/code boxes

3.1.3 html_css_files

See *Custom CSS* (page 61) and html_css_files[125].

[121] https://www.sphinx-doc.org/en/master/usage/configuration.html
[122] https://www.sphinx-doc.org/en/master/usage/configuration.html#confval-exclude_patterns
[123] https://www.sphinx-doc.org/en/master/usage/configuration.html#confval-extensions
[124] https://sphinx-copybutton.readthedocs.io/
[125] https://www.sphinx-doc.org/en/master/usage/configuration.html#confval-html_css_files

3.1.4 html_sourcelink_suffix

By default, a `.txt` suffix is added to source files. This is only relevant if the chosen HTML theme supports source links and if html_show_sourcelink[126] is `True`.

Jupyter notebooks with the suffix `.ipynb.txt` are normally not very useful, so if you want to avoid the additional suffix, set html_sourcelink_suffix[127] to the empty string:

```
html_sourcelink_suffix = ''
```

3.1.5 latex_additional_files

latex_additional_files[128] can be useful if you are using BibTeX files, see *References* (page 65).

3.1.6 mathjax3_config

The configuration value mathjax3_config[129] can be useful to enable *Automatic Equation Numbering* (page 21).

For Sphinx versions below 4.0.0, which used MathJax version 2, the relevant configuration value was called `mathjax_config`.

3.1.7 pygments_style

Use pygments_style[130] to change the color/font theme that's used for syntax highlighting in source code.

This affects both *code cells* (page 29) and *code blocks in Markdown cells* (page 23) (unless overwritten by the html_theme[131]).

3.1.8 suppress_warnings

Warnings can be really helpful to detect small mistakes, and you should consider invoking Sphinx with the -W[132] option, which turns warnings into errors. However, warnings can also be annoying, especially if you are fully aware of the "problem", but you simply don't care about it for some reason. In this case, you can use suppress_warnings[133] to silence specific types of warnings.

If you want to suppress all warnings from `nbsphinx`, use this:

```
suppress_warnings = [
    'nbsphinx',
]
```

You can also be more specific:

[126] https://www.sphinx-doc.org/en/master/usage/configuration.html#confval-html_show_sourcelink
[127] https://www.sphinx-doc.org/en/master/usage/configuration.html#confval-html_sourcelink_suffix
[128] https://www.sphinx-doc.org/en/master/usage/configuration.html#confval-latex_additional_files
[129] https://www.sphinx-doc.org/en/master/usage/extensions/math.html#confval-mathjax3_config
[130] https://www.sphinx-doc.org/en/master/usage/configuration.html#confval-pygments_style
[131] https://www.sphinx-doc.org/en/master/usage/configuration.html#confval-html_theme
[132] https://www.sphinx-doc.org/en/master/man/sphinx-build.html#cmdoption-sphinx-build-W
[133] https://www.sphinx-doc.org/en/master/usage/configuration.html#confval-suppress_warnings

```
suppress_warnings = [
    'nbsphinx.localfile',
    'nbsphinx.gallery',
    'nbsphinx.thumbnail',
    'nbsphinx.notebooktitle',
    'nbsphinx.ipywidgets',
]
```

3.2 nbsphinx Configuration Values

3.2.1 nbsphinx_allow_errors

If `True`, the build process is continued even if an exception occurs.

See *Ignoring Errors* (page 46).

3.2.2 nbsphinx_assume_equations

If `False`, do not force loading MathJax on HTML pages generated from notebooks.

3.2.3 nbsphinx_codecell_lexer

Default Pygments lexer for syntax highlighting in code cells. If available, this information is taken from the notebook metadata instead.

3.2.4 nbsphinx_custom_formats

See *Custom Notebook Formats* (page 51).

3.2.5 nbsphinx_epilog

See *Prolog and Epilog* (page 49).

3.2.6 nbsphinx_execute

Whether to execute notebooks before conversion or not. Possible values: `'always'`, `'never'`, `'auto'` (default).

See *Explicitly Dis-/Enabling Notebook Execution* (page 46).

3.2.7 nbsphinx_execute_arguments

Kernel arguments used when executing notebooks.

See *Configuring the Kernels* (page 48).

3.2.8 nbsphinx_input_prompt

Input prompt for code cells. %s is replaced by the execution count.

To get a prompt similar to the Classic Notebook, use

```
nbsphinx_input_prompt = 'In [%s]:'
```

3.2.9 nbsphinx_kernel_name

Use a different kernel than stored in the notebook metadata, e.g.:

```
nbsphinx_kernel_name = 'python3'
```

See *Configuring the Kernels* (page 48).

3.2.10 nbsphinx_output_prompt

Output prompt for code cells. %s is replaced by the execution count.

To get a prompt similar to the Classic Notebook, use

```
nbsphinx_output_prompt = 'Out[%s]:'
```

3.2.11 nbsphinx_prolog

See *Prolog and Epilog* (page 49).

3.2.12 nbsphinx_prompt_width

Width of input/output prompts (HTML only).

If a prompt is wider than that, it protrudes into the left margin.

Any CSS length can be specified.

3.2.13 nbsphinx_requirejs_options

Options for loading RequireJS. See *nbsphinx_requirejs_path* (page 18).

3.2.14 nbsphinx_requirejs_path

URL or local path to override the default URL for RequireJS[134].

If you use a local file, it should be located in a directory listed in html_static_path[135].

Set to empty string to disable loading RequireJS.

[134] https://requirejs.org/

[135] https://www.sphinx-doc.org/en/master/usage/configuration.html#confval-html_static_path

3.2.15 nbsphinx_responsive_width

If the browser window is narrower than this, input/output prompts are on separate lines (HTML only).

Any CSS length can be specified.

3.2.16 nbsphinx_thumbnails

A dictionary mapping from a document name (i.e. source file without suffix but with subdirectories) – optionally containing wildcards – to a thumbnail path to be used in a *thumbnail gallery* (page 53). Thumbnails specified in notebooks will override those provided in this dictionary.

See *Specifying Thumbnails* (page 58).

3.2.17 nbsphinx_timeout

Controls when a cell will time out. The timeout is given in seconds. Given `-1`, cells will never time out, which is also the default.

See *Cell Execution Timeout* (page 49).

3.2.18 nbsphinx_widgets_options

Options for loading Jupyter widgets resources. See *nbsphinx_widgets_path* (page 19).

3.2.19 nbsphinx_widgets_path

URL or local path to override the default URL for Jupyter widgets resources. See *Interactive Widgets (HTML only)* (page 36).

If you use a local file, it should be located in a directory listed in html_static_path[136].

For loading the widgets resources, RequireJS is needed, see *nbsphinx_requirejs_path* (page 18).

If `nbsphinx_widgets_path` is not specified, widgets resources are only loaded if at least one notebook actually uses widgets. If you are loading the relevant JavaScript code by some other means already, you can set this option to the empty string to avoid loading it a second time.

`doc/configuration.ipynb` ends here.

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

4 Markdown Cells

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

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

- Red
- Green
- Blue

Note: JupyterLab and JupyterNotebook uses a different Markdown parser than nbsphinx (which currently uses Pandoc). In case that your Bulletpoints do render in the notebook and do not render with nbsphinx, please add one blank line before the bulletpoints. ***

[136] `https://www.sphinx-doc.org/en/master/usage/configuration.html#confval-html_static_path`

1. One
2. Two
3. Three

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

4.1 Equations

Inline equations like $e^{i\pi} = -1$ can be created by putting a LaTeX expression between two dollar signs, like this: `$\text{e}^{i\pi} = -1$`.

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[137]:

> 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.

Equations can also be displayed on their own line like this:

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

This can be done by simply using one of the LaTeX math environments, like so:

```
\begin{equation}
\int\limits_{-\infty}^\infty f(x) \delta(x - x_0) dx = f(x_0)
\end{equation}
```

Note

For equations to be shown in HTML output, you have to specify a math extension[138] in your *extensions* (page 15) setting, e.g.:

```
extensions = [
    'nbsphinx',
    'sphinx.ext.mathjax',
    # ... other useful extensions ...
]
```

[137] https://pandoc.org/MANUAL.html#math
[138] https://www.sphinx-doc.org/en/master/usage/extensions/math.html

4.1.1 Automatic Equation Numbering

This is not automatically enabled in Jupyter notebooks, but you can install a notebook extension in order to enable equation numbering: https://jupyter-contrib-nbextensions.readthedocs.io/en/latest/nbextensions/equation-numbering/readme.html.

Automatic Equation Numbering is enabled on https://nbviewer.jupyter.org/, see e.g. the latest version of this very notebook at the link https://nbviewer.jupyter.org/github/spatialaudio/nbsphinx/blob/master/doc/markdown-cells.ipynb#Automatic-Equation-Numbering.

When using `nbsphinx`, you can use the following `mathjax3_config` setting in your `conf.py` file to enable automatic equation numbering in HTML output.

```
mathjax3_config = {
    'tex': {'tags': 'ams', 'useLabelIds': True},
}
```

This works for Sphinx version 4 (and higher), which uses MathJax version 3. For older Sphinx versions, the corresponding configuration looks like this:

```
mathjax_config = {
    'TeX': {'equationNumbers': {'autoNumber': 'AMS', 'useLabelIds': True}},
}
```

In LaTeX output, the equations are numbered by default.

You can use `\label{...}` to give a unique label to an equation:

$$\phi = \frac{1 + \sqrt{5}}{2} \tag{2}$$

```
\begin{equation}
\phi = \frac{1 + \sqrt{5}}{2}
\label{golden-mean}
\end{equation}
```

If automatic equation numbering is enabled, you can later reference that equation using its label. You can use `\eqref{golden-mean}` for a reference with parentheses: (2), or `\ref{golden-mean}` for a reference without them: 2.

In HTML output, these equation references only work for equations within a single HTML page. In LaTeX output, equations from other notebooks can be referenced, e.g. (08.15).

4.1.2 Manual Equation Numbering

If you prefer to assign equation numbers (or some kind of names) manually, you can do so with `\tag{...}`:

$$a^2 + b^2 = c^2 \tag{99.4}$$

```
\begin{equation}
a^2 + b^2 = c^2
\tag{99.4}
\label{pythagoras}
\end{equation}
```

The above equation has the number 99.4.

4.2 Citations

According to https://nbconvert.readthedocs.io/en/latest/latex_citations.html, nbconvert supports citations using a special HTML-based syntax. nbsphinx supports the same syntax.

Example: [Kluyver *et al.*, 2016].

```
<cite data-cite="kluyver2016jupyter">(Kluyver et al. 2016)</cite>
```

You don't actually have to use <cite>, any inline HTML tag can be used, e.g. : [Pérez *et al.*, 2011].

```
<strong data-cite="perez2011python">Python: An Ecosystem for Scientific Computing</strong>
```

In-text citations like Kluyver *et al.* [2016] can be created like this:

```
<cite data-cite-t="kluyver2016jupyter">Kluyver et al. (2016)</cite>
```

You'll also have to define a list of references, see *the section about references* (page 65).

There is also a Notebook extension which may or may not be useful: https://github.com/takluyver/cite2c.

4.2.1 Footnote citations

Since version 2.0.0 of sphinxcontrib-bibtex, footnote citations[139] are possible. This generates footnotes for all foot-citations up to the point of the *bibliography directive* (page 66), which is typically placed at the end of the source file.

Depending on whether the documentation is rendered into HTML or into LaTeX/PDF, the citations are either placed into a bibliography as ordinary citations (HTML output) or placed into the footnotes of the citation's respective page (PDF).

Example:[140].

```
<cite data-footcite="perez2011python">(Pérez et al. 2011)</cite>
```

Example for an in-text citation: Pérez *et al.*[140].

```
<cite data-footcite-t="perez2011python">Pérez et al. (2011)</cite>
```

As footnote references are restricted to their own Jupyter notebook or other source file, a raw nbconvert cell of reST format (see *the section about raw cells* (page 40)) can be added to the notebook, containing the

```
.. footbibliography::
```

directive.

Alternatively, one can use the *nbsphinx epilog* (page 49) by setting it to, e.g.,

```
nbsphinx_epilog = r"""
.. footbibliography::
"""
```

[139] https://sphinxcontrib-bibtex.readthedocs.io/en/latest/usage.html#role-footcite

[140] Fernando Pérez, Brian E. Granger, and John D. Hunter. Python: an ecosystem for scientific computing. *Computing in Science Engineering*, 13(2):13–21, 2011. doi:10.1109/MCSE.2010.119[141].

[141] https://doi.org/10.1109/MCSE.2010.119

4.3 Code

We can also write code with nice syntax highlighting:

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

4.4 Tables

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

4.5 Images

Local image:

```
![Jupyter notebook icon](images/notebook_icon.png)
```

Remote image:

```
![Python logo (remote)](https://www.python.org/static/img/python-logo-large.png)
```

4.5.1 Using the HTML <img> tag

The aforementioned Markdown syntax for including images doesn't allow specifying the image size.

If you want to control the size of the included image, you can use the HTML <img>[142] element with the `width` attribute like this:

```
<img src="images/notebook_icon.png" alt="Jupyter notebook icon" width="300">
```

In addition to the `src`, `alt`, `width` and `height` attributes, you can also use the `class` attribute, which is simply forwarded to the HTML output (and ignored in LaTeX output). All other attributes are ignored.

4.5.2 SVG support for LaTeX

LaTeX doesn't support SVG images, but there are Sphinx extensions that can be used for automatically converting SVG images for inclusion in LaTeX output.

Just include one of the following options in the list of *extensions* (page 15) in your `conf.py` file.

- `'sphinxcontrib.inkscapeconverter'` or `'sphinxcontrib.rsvgconverter'`: See https://github.com/missinglinkelectronics/sphinxcontrib-svg2pdfconverter for installation instructions.

  The external programs `inkscape` or `rsvg-convert` (Debian/Ubuntu package `librsvg2-bin`; `conda` package `librsvg`) are needed, respectively.

- `'sphinx.ext.imgconverter'`: This is a built-in Sphinx extension, see https://www.sphinx-doc.org/en/master/usage/extensions/imgconverter.html.

  This needs the external program `convert` from *ImageMagick*.

  The disadvantage of this extension is that SVGs are converted to bitmap images.

If one of those extensions is installed, SVG images can be used even for LaTeX output:

[142] https://www.w3.org/TR/html52/semantics-embedded-content.html#the-img-element

```
![Python logo](images/python_logo.svg)
```

Remote SVG images can also be used (and will be shown in the LaTeX output):

```
![Jupyter logo](https://jupyter.org/assets/homepage/main-logo.svg)
```

4.6 Cell Attachments

Images can also be embedded in the notebook itself. Just drag an image file into the Markdown cell you are just editing or copy and paste some image data from an image editor/viewer.

The generated Markdown code will look just like a "normal" image link, except that it will have an `attachment:` prefix:

```
![a stick figure](attachment:stickfigure.png)
```

These are cell attachments:

In the Jupyter Notebook, there is a speciall "Attachments" cell toolbar which you can use to see all attachments of a cell and delete them, if needed.

4.7 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[143] elements can be created like this:

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

[143] https://developer.mozilla.org/en-US/docs/Web/HTML/Element/audio

Example:

The HTML audio element is not supported!

HTML5 video[144] 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.

4.8 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[145]), 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.

Warning

While this works nicely with `nbsphinx`, JupyterLab and the Classic Jupyter Notebook, This doesn't work correctly in `nbconvert` and by extension on `https://nbviewer.jupyter.org/` and Github's notebook preview.

[144] `https://developer.mozilla.org/en-US/docs/Web/HTML/Element/video`
[145] `https://github.com/jupyter/notebook/issues/1292`

See https://github.com/jupyter/nbconvert/issues/1125.

Note

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

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

4.9 Links to Other Notebooks

Relative links to local notebooks can be used: *a link to a notebook in a subdirectory* (page 52), 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* (page 52), *another version of the same link* (page 52).

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* (page 53).

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!

And if there are double quotes in the section title, you'll have to replace them with `%22`, like e.g. `#That's-a-%22Strange%22-Section` in this link: *a section with "strange" characters* (page 53).

BTW, links to sections of the current notebook work, too, e.g. *beginning of this section* (page 27).

This can be done, as expected, like this:

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

It's also possible to create a *link to the beginning of the current page* (page 19), by simply using a `#` character:

```
[link to the beginning of the current page](#)
```

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

Links to files whose extension is in the configuration value source_suffix[146], will be converted to links to the generated HTML/LaTeX pages. Example: *A reStructuredText file* (page 63).

This was created with:

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

Links to sub-sections are also possible. Example: *Sphinx Directives* (page 65).

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.

4.11 Links to Local Files

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, links are created, but the files are not copied to the target directory.

4.12 Links to Domain Objects

Links to Sphinx domain objects[147] (such as a Python class or JavaScript function) are also possible. For example: *example_python_function()* (page 65).

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[148] extension!

In some situations, you might prefer to have the default Sphinx formatting and checking in place when linking to domain objects. In such a case, *raw cells in "reST" format* (page 42) could be an alternative worthwhile considering. They allow one to use any kind of Sphinx roles and directives inside a Jupyter Notebook.

`doc/markdown-cells.ipynb` ends here.

[146] https://www.sphinx-doc.org/en/master/config.html#confval-source_suffix
[147] https://www.sphinx-doc.org/en/master/usage/restructuredtext/domains.html
[148] https://www.sphinx-doc.org/en/master/ext/autodoc.html

5 Code Cells

5.1 Code, Output, Streams

An empty code cell:

[ ]:

Two empty lines:

[ ]:

Leading/trailing empty lines:

[1]:
```
# 2 empty lines before, 1 after
```

A simple output:

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

[2]:
```
42
```

The standard output stream:

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

```
Hello, world!
```

Normal output + standard output

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

```
Hello, world!
```

[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.

[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
```

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

Note

Using the IPython kernel, the order is actually mixed up, see https://github.com/ipython/ipykernel/issues/280.

5.2 Special Display Formats

See IPython example notebook[149].

5.2.1 Local Image Files

[6]:
```
from IPython.display import Image
i = Image(filename='images/notebook_icon.png')
i
```

[6]:

[7]:
```
display(i)
```

See also *SVG support for LaTeX* (page 24).

[8]:
```
from IPython.display import SVG
SVG(filename='images/python_logo.svg')
```

[8]:

[149] https://nbviewer.jupyter.org/github/ipython/ipython/blob/main/examples/IPython%20Kernel/Rich%20Output.ipynb

5.2.2 Image URLs

[9]:
```
Image(url='https://www.python.org/static/img/python-logo-large.png')
```

[9]:
```
<IPython.core.display.Image object>
```

[10]:
```
Image(url='https://www.python.org/static/img/python-logo-large.png', embed=True)
```

[10]:

[11]:
```
Image(url='https://jupyter.org/assets/homepage/main-logo.svg')
```

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

5.2.3 Math

[12]:
```
from IPython.display import Math
eq = Math(r'\int\limits_{-\infty}^\infty f(x) \delta(x - x_0) dx = f(x_0)')
eq
```

[12]:

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

[13]:
```
display(eq)
```

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

[14]:
```
from IPython.display import Latex
Latex(r'This is a \LaTeX{} equation: $a^2 + b^2 = c^2$')
```

[14]:

This is a LaTeX equation: $a^2 + b^2 = c^2$

[15]:
```
%%latex
\begin{equation}
\int\limits_{-\infty}^\infty f(x) \delta(x - x_0) dx = f(x_0)
\end{equation}
```

$$\int_{-\infty}^{\infty} f(x)\delta(x - x_0)dx = f(x_0) \tag{3}$$

(continues on next page)

(continued from previous page)

5.2.4 Plots

> Make sure to use at least version 0.1.6 of the `matplotlib-inline` package (which is an automatic dependency of the `ipython` package).

By default, the plots created with the "inline" backend have the wrong size. More specifically, PNG plots (the default) will be slightly larger than SVG and PDF plots.

This can be fixed easily by creating a file named `matplotlibrc` (in the directory where your Jupyter notebooks live, e.g. in this directory: `matplotlibrc`) and adding the following line:

```
figure.dpi: 96
```

If you are using Git to manage your files, don't forget to commit this local configuration file to your repository. Different directories can have different local configurations. If a given configuration should apply to multiple directories, symbolic links can be created in each directory.

For more details, see Default Values for Matplotlib's "inline" Backend[150].

By default, plots are generated in the PNG format. In most cases, it looks better if SVG plots are used for HTML output and PDF plots are used for LaTeX/PDF. This can be achieved by setting *nbsphinx_execute_arguments* (page 17) in your `conf.py` file like this:

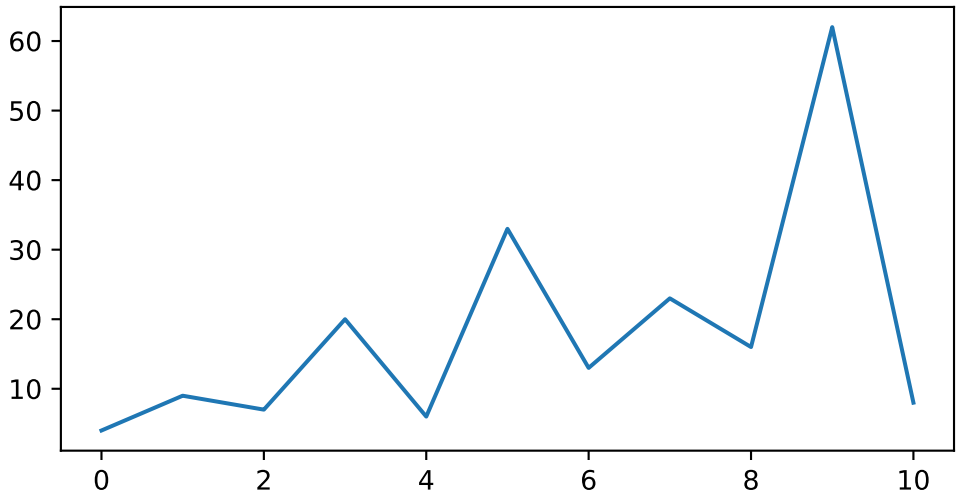
```
nbsphinx_execute_arguments = [
    "--InlineBackend.figure_formats={'svg', 'pdf'}",
]
```

In the following example, `nbsphinx` should use an SVG image in the HTML output and a PDF image for LaTeX/PDF output (other Jupyter clients like JupyterLab will still show the default PNG format).

[16]:
```
import matplotlib.pyplot as plt
```

[17]:
```
fig, ax = plt.subplots(figsize=[6, 3])
ax.plot([4, 9, 7, 20, 6, 33, 13, 23, 16, 62, 8]);
```

[150] `https://nbviewer.jupyter.org/github/mgeier/python-audio/blob/master/plotting/matplotlib-inline-defaults.ipynb`

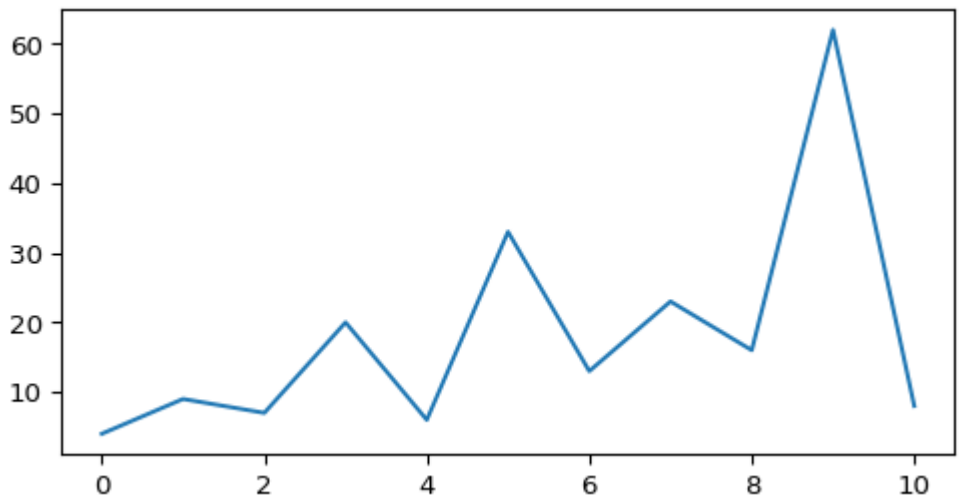


For comparison, this is how it would look in PNG format ...

[18]:
```
%config InlineBackend.figure_formats = ['png']
```

[19]:
```
fig
```

[19]:

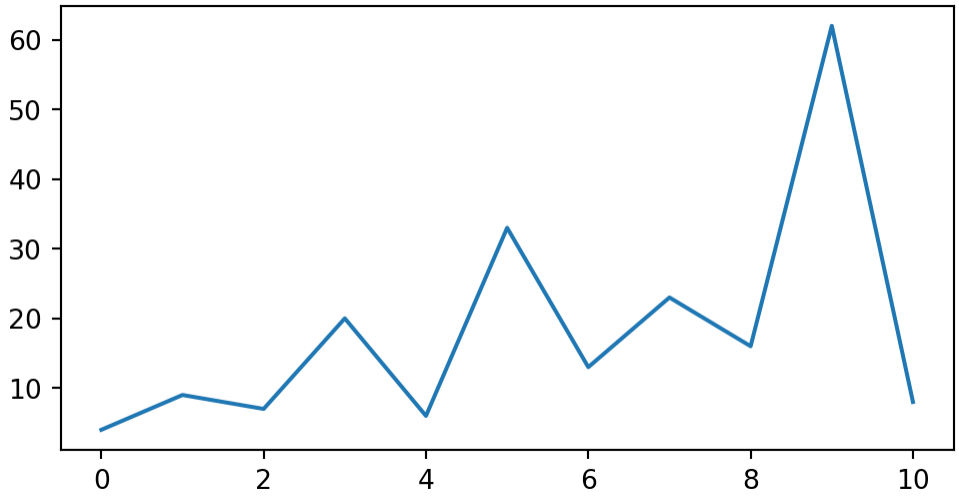


... and in `'png2x'` (a.k.a. `'retina'`) format:

[20]:
```
%config InlineBackend.figure_formats = ['png2x']
```

[21]:
```
fig
```

[21]:



5.2.5 Pandas Dataframes

Pandas dataframes[151] should be displayed as nicely formatted HTML tables (if you are using HTML output).

[22]:
```
import numpy as np
import pandas as pd
```

[23]:
```
df = pd.DataFrame(np.random.randint(0, 100, size=[10, 4]),
                  columns=[r'$\alpha$', r'$\beta$', r'$\gamma$', r'$\delta$'])
df
```

[23]:

	α	β	γ	δ
0	35	2	46	14
1	55	42	37	86
2	16	68	2	52
3	72	99	48	64
4	70	24	7	54
5	26	3	10	73
6	44	45	29	47
7	52	0	24	39
8	32	33	40	52
9	80	20	70	5

5.2.6 Markdown Content

[24]:
```
from IPython.display import Markdown
```

[25]:
```
md = Markdown("""
# Markdown

It *should* show up as **formatted** text
with things like [links] and images.

[links]: https://jupyter.org/
```

(continues on next page)

[151] https://pandas.pydata.org/pandas-docs/stable/user_guide/dsintro.html#dataframe

(continued from previous page)

```
![Jupyter notebook icon](images/notebook_icon.png)

## Markdown Extensions

There might also be mathematical equations like
$a^2 + b^2 = c^2$
and even tables:

A     | B     | A and B
------|-------|--------
False | False | False
True  | False | False
False | True  | False
True  | True  | True

""")
md
```

[25]:

5.2.6.1 Markdown

It *should* show up as **formatted** text with things like links[152] and images.

Markdown Extensions

There might also be mathematical equations like $a^2 + b^2 = c^2$ and even tables:

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

5.2.7 YouTube Videos

[26]:

```
from IPython.display import YouTubeVideo
YouTubeVideo('9_OIs49m56E')
```

[26]:

5.2.8 Interactive Widgets (HTML only)

The basic widget infrastructure is provided by the ipywidgets[153] module. More advanced widgets are available in separate packages, see for example `https://jupyter.org/widgets`.

The JavaScript code which is needed to display Jupyter widgets is loaded automatically (using RequireJS). If you want to use non-default URLs or local files, you can use the *nbsphinx_widgets_path* (page 19) and *nbsphinx_requirejs_path* (page 18) settings.

[27]:
```
import ipywidgets as w
```

[28]:
```
slider = w.IntSlider()
slider.value = 42
slider
```

[28]:
```
IntSlider(value=42)
```

A widget typically consists of a so-called "model" and a "view" into that model.

If you display a widget multiple times, all instances act as a "view" into the same "model". That means that their state is synchronized. You can move either one of these sliders to try this out:

[29]:
```
slider
```

[29]:
```
IntSlider(value=42)
```

You can also link different widgets.

Widgets can be linked via the kernel (which of course only works while a kernel is running) or directly in the client (which even works in the rendered HTML pages).

Widgets can be linked uni- or bi-directionally.

Examples for all 4 combinations are shown here:

[153] `https://ipywidgets.readthedocs.io/`

[30]:
```
link = w.IntSlider(description='link')
w.link((slider, 'value'), (link, 'value'))
jslink = w.IntSlider(description='jslink')
w.jslink((slider, 'value'), (jslink, 'value'))
dlink = w.IntSlider(description='dlink')
w.dlink((slider, 'value'), (dlink, 'value'))
jsdlink = w.IntSlider(description='jsdlink')
w.jsdlink((slider, 'value'), (jsdlink, 'value'))
w.VBox([link, jslink, dlink, jsdlink])
```

[30]:
```
VBox(children=(IntSlider(value=42, description='link'), IntSlider(value=0, description='jslink
↪'), IntSlider(va…
```

[31]:
```
tabs = w.Tab()
for idx, obj in enumerate([df, fig, eq, i, md, slider]):
    out = w.Output()
    with out:
        display(obj)
    tabs.children += out,
    tabs.set_title(idx, obj.__class__.__name__)
tabs
```

[31]:
```
Tab(children=(Output(), Output(), Output(), Output(), Output(), Output()), selected_index=0,␣
↪titles=('DataFram…
```

Other Languages

The examples shown here are using Python, but the widget technology can also be used with different Jupyter kernels (i.e. with different programming languages).

5.2.8.1 Troubleshooting

To obtain more information if widgets are not displayed as expected, you will need to look at the error message in the web browser console.

> To figure out how to open the web browser console, you may look at the web browser documentation:
>
> Chrome: https://developer.chrome.com/docs/devtools/open/#shortcuts
>
> Firefox: https://developer.mozilla.org/en-US/docs/Tools/Web_Console#opening-the-web-console

The error is most probably linked to the JavaScript files not being loaded or loaded in the wrong order within the HTML file. To analyze the error, you can inspect the HTML file within the web browser (e.g.: right-click on the page and select *View Page Source*) and look at the `<head>` section of the page. That section should contain some JavaScript libraries. Those relevant for widgets are:

```
<!-- require.js is a mandatory dependency for jupyter-widgets -->
<script crossorigin="anonymous" integrity="sha256-Ae2Vz/4ePdIu6ZyI/5ZGsYnb+m0JlOmKPjt6XZ9JJkA=
↪" src="https://cdnjs.cloudflare.com/ajax/libs/require.js/2.3.4/require.min.js"></script>
<!-- jupyter-widgets JavaScript -->
<script type="text/javascript" src="https://unpkg.com/@jupyter-widgets/html-manager@^0.18.0/
↪dist/embed-amd.js"></script>
<!-- JavaScript containing custom Jupyter widgets -->
<script src="../_static/embed-widgets.js"></script>
```

The two first elements are mandatory. The third one is required only if you designed your own widgets but did not publish them on npm.js.

If those libraries appear in a different order, the widgets won't be displayed.

Here is a list of possible solutions:

- If the widgets are **not displayed**, see #519[154].
- If the widgets are **displayed multiple times**, see #378[155].

5.2.9 Arbitrary JavaScript Output (HTML only)

[32]:
```
%%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>
```

5.2.10 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.

[33]:
```
display({
    'text/x-python': 'print("Hello, world!")',
    'text/x-haskell': 'main = putStrLn "Hello, world!"',
}, raw=True)
```

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

5.3 ANSI Colors

The standard output and standard error streams may contain ANSI escape sequences[156] to change the text and background colors.

[34]:
```
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')
```

```
ABCDEFGHIJKLMNOPQRSTUV
```

```
BEWARE: ugly colors!
```

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

[35]:
```
text = ' XYZ '
formatstring = '\x1b[{}m' + text + '\x1b[m'

print(' ' * 6 + ' ' * len(text) +
```

(continues on next page)

[154] https://github.com/spatialaudio/nbsphinx/issues/519
[155] https://github.com/spatialaudio/nbsphinx/issues/378
[156] https://en.wikipedia.org/wiki/ANSI_escape_code

(continued from previous page)

```
      ''.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  XYZ
  31  XYZ  XYZ       XYZ  XYZ  XYZ  XYZ  XYZ  XYZ
1;31  XYZ  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  XYZ
  33  XYZ  XYZ  XYZ  XYZ       XYZ  XYZ  XYZ  XYZ
1;33  XYZ  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  XYZ
  35  XYZ  XYZ  XYZ  XYZ  XYZ  XYZ       XYZ  XYZ
1;35  XYZ  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  XYZ
  37  XYZ  XYZ  XYZ  XYZ  XYZ  XYZ  XYZ  XYZ
1;37  XYZ  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[157].

[36]:
```
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
    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 9101112131415161718192021222324252627282930313233343 5
  0  X X X X X X X X
 16
 52
 88
124
160
196
232
```

You can even use 24-bit RGB colors:

[37]:
```
start = 255, 0, 0
end = 0, 0, 255
length = 79
out = []
```

(continues on next page)

[157] https://web.archive.org/web/20190109005413/http://bitmote.com/index.php?post/2012/11/19/Using-ANSI-Color-Codes-to-Colorize-Your-Bash-Prompt-on-Linux

(continued from previous page)

```
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]}mX\x1b[m'.format(rgb=rgb))
print(''.join(out))
```

```
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
```

doc/code-cells.ipynb ends here.

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

6 Raw Cells

Any Jupyter Notebook consists of cells of three different types: *Code cells* (page 29), *Markdown cells* (page 19) , and/or Raw cells. While most Jupyter Notebook users are very familiar with Code cells and Markdown cells in Jupyter Notebooks, Raw cells are less frequently used. For Jupyter Notebook, they are introduced here[158] and for JupyterLab here[159]. The Raw cells are also sometimes referred to as Raw NBConvert cells in the context of nbconvert[160]. The Raw 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.

6.1 Usage

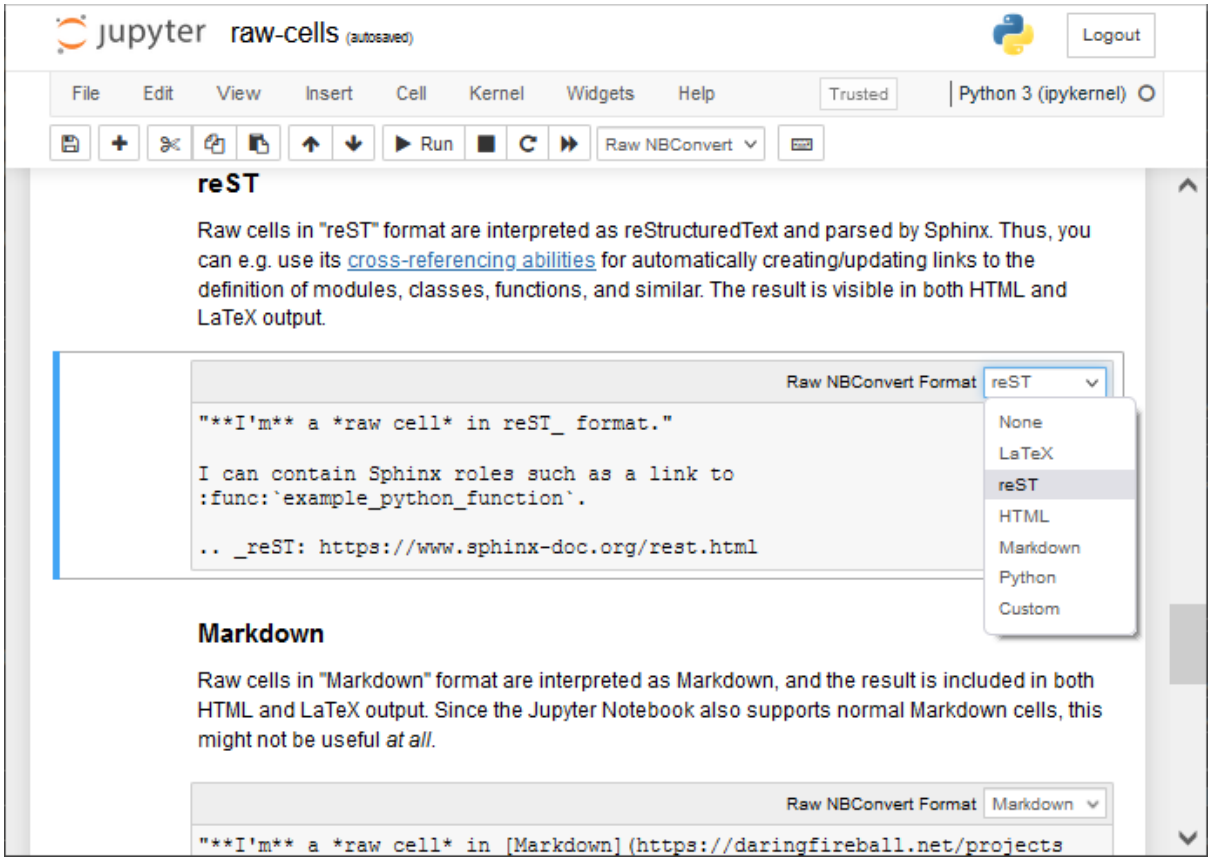
Raw cells are created differently depending on the user interface.

6.1.1 Jupyter Notebook

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

1. Select "Raw NBConvert" in the Menu Toolbar (just below the two menus "Widgets" and "Help").
2. Click on the "Raw NBConvert Format" dropdown menu within the cell and select "reST".

[158] https://jupyter-notebook.readthedocs.io/en/stable/notebook.html?highlight=raw#raw-cells
[159] https://jupyterlab.readthedocs.io/en/stable/extension/notebook.html?highlight=raw#model
[160] https://nbconvert.readthedocs.io/en/latest/architecture.html?highlight=raw#a-detailed-pipeline-exploration

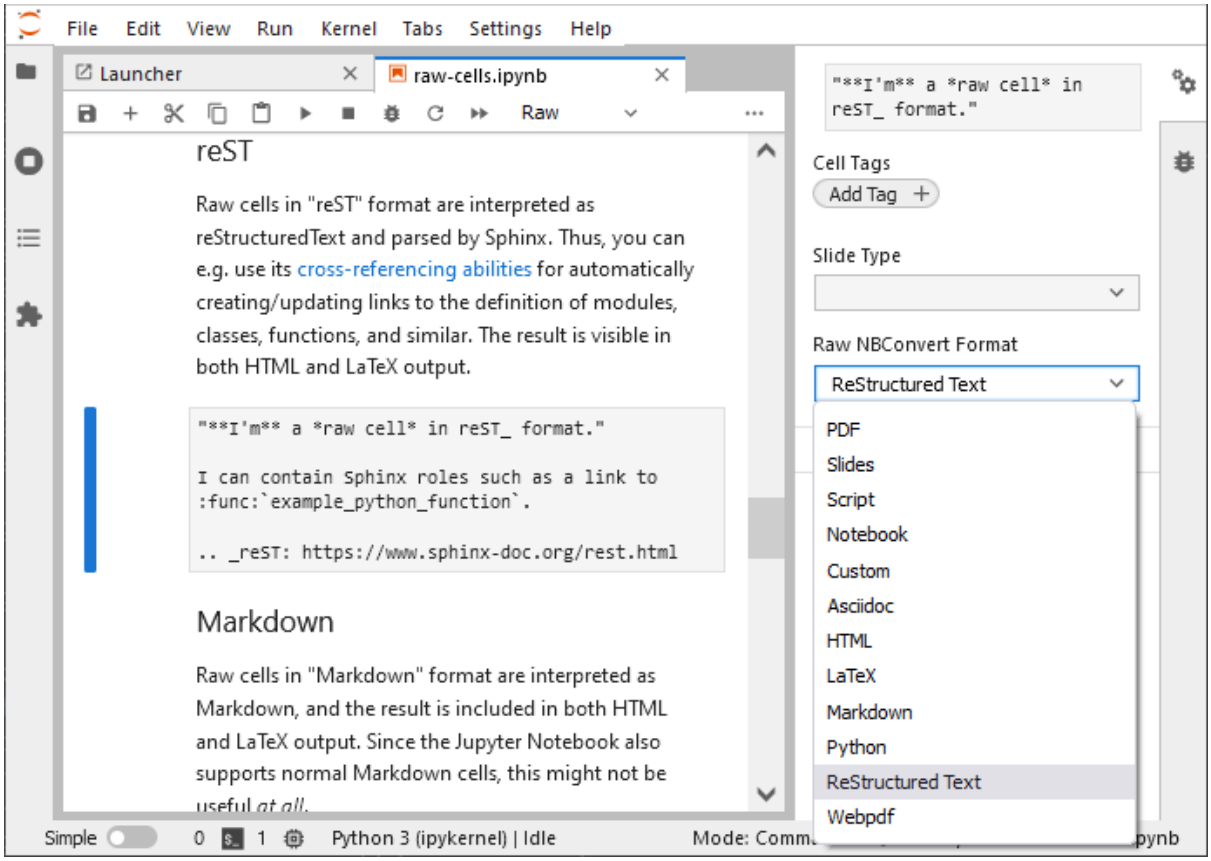


6.1.2 JupyterLab

To select a desired format from within JupyterLab, first activate the right sidebar by clicking on View in the Menu Toolbar. Then you ensure that in front of Show Right Sidebar there is a tick. Once the Right Sidebar is shown, you are ready to go.

Now you select the cell containing your special code and choose options from the following dropdown menus:

1. Select “Raw” in the Notebook Toolbar (just next to the symbols that run cells or reload the kernel).
2. Click on “Raw NBConvert Format” in the Right Sidebar and select “reStructured Text”.



6.2 Available Raw Cell Formats

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

6.2.1 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."

6.2.2 reST

Raw cells in "reST" format are interpreted as reStructuredText and parsed by Sphinx. Thus, you can e.g. use its cross-referencing abilities[161] for automatically creating/updating links to the definition of modules, classes, functions, and similar. The result is visible in both HTML and LaTeX output.

"**I'm** a *raw cell* in reST[162] format."

I can contain Sphinx roles such as a link to `example_python_function()` (page 65).

[161] https://www.sphinx-doc.org/en/master/usage/restructuredtext/domains.html#cross-referencing-python-objects
[162] https://www.sphinx-doc.org/rest.html

6.2.3 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 Markdown[163] format."

6.2.4 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.

6.2.5 LaTeX

Raw cells in "LaTeX" format are only visible in LaTeX output.

I'm a *raw cell* in LaTeX format.

6.2.6 Python

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

doc/raw-cells.ipynb ends here.

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

7 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:

[2]:
```
answer
```

[2]:
```
42
```

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.

doc/hidden-cells.ipynb ends here.

[163] https://daringfireball.net/projects/markdown/

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

8 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

8.1 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.

8.1.1 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.

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

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

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

[2]:
```
42
```

8.1.2 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.

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

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

[4]:
```
42
```

8.1.3 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 or per-cell basis – see *Ignoring Errors* (page 46) (*per cell* (page 47)).
2. Execute the notebook beforehand and save the results, like it's done in this example notebook:

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

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

ZeroDivisionError: division by zero
```

8.1.4 Client-specific Outputs

When `nbsphinx` executes notebooks, it uses the `nbconvert` module to do so. Certain Jupyter clients might produce output that differs from what `nbconvert` would produce. To preserve those original outputs, the notebook has to be executed and saved before running Sphinx.

For example, the JupyterLab help system shows the help text as cell outputs, while executing with `nbconvert` doesn't produce any output.

[6]:
```
sorted?
```

```
Signature: sorted(iterable, /, *, key=None, reverse=False)
Docstring:
Return a new list containing all items from the iterable in ascending order.

A custom key function can be supplied to customize the sort order, and the
reverse flag can be set to request the result in descending order.
Type:      builtin_function_or_method
```

8.1.5 Interactive Input

If your code asks for user input, it probably doesn't work when executed by Sphinx/`nbsphinx`. You'll probably get an error like this:

```
StdinNotImplementedError: raw_input was called, but this frontend does not support input␣
↪requests.
```

In this case, you can run the notebook interactively, provide the desired inputs and then save the notebook including its cell outputs.

[7]:
```
name = input('What... is your name?')
quest = input('What... is your quest?')
color = input('What... is your favorite color?')
```

```
What… is your name? Sir Lancelot of Camelot
What… is your quest? To seek the Holy Grail
What… is your favorite color? Blue
```

..... doc/pre-executed.ipynb ends here.

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

8.2 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:

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

doc/never-execute.ipynb ends here.

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

8.3 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 four options:

1. Manually execute the notebook in question and save the results, see *the pre-executed example notebook* (page 44).
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
   },
   ```

4. Allow errors on a per-cell basis using the raises-exception tag, see *Ignoring Errors on a Cell-by-Cell Basis* (page 47).

This very notebook is an example for the third 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.

[1]:
```
nonsense
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[1], line 1
----> 1 nonsense

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

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

```
---------------------------------------------------------------------------
ZeroDivisionError                         Traceback (most recent call last)
Cell In[2], line 1
----> 1 42 / 0

ZeroDivisionError: division by zero
```

[3]:
```
6 * 7
```

[3]:
```
42
```

..... doc/allow-errors.ipynb ends here.

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

8.4 Ignoring Errors on a Per-Cell Basis

Instead of ignoring errors for all notebooks or for some selected notebooks (see *the previous notebook* (page 46)), you can be more fine-grained and just allow errors on certain code cells by tagging them with the `raises-exception` tag.

[1]:
```
'no problem'
```

[1]:
```
'no problem'
```

The following code cell has the `raises-exception` tag.

[2]:
```
problem
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[2], line 1
----> 1 problem

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

The following code cell is executed even though the previous cell raised an exception.

[3]:
```
'no problem'
```

[3]:
```
'no problem'
```

Note

The behavior of the `raises-exception` tag doesn't match its name. While it does *allow* exceptions, it does not check if an exception is actually raised!

This will hopefully be fixed at some point, see https://github.com/jupyter/nbconvert/issues/730.

..... doc/allow-errors-per-cell.ipynb ends here.

The following section was generated from doc/configuring-kernels.ipynb

8.5 Configuring the Kernels

8.5.1 Kernel Name

If we have multiple kernels installed, we can choose to override the kernel saved in the notebook using *nbsphinx_kernel_name* (page 18):

```
nbsphinx_kernel_name = 'python-upstream-dev'
```

which uses the kernel named `python-upstream-dev` instead of the kernel name stored in the notebook.

8.5.2 Kernel Arguments

We can pass arguments to the kernel by using *nbsphinx_execute_arguments* (page 17), for example to set *plot options* (page 32):

```
nbsphinx_execute_arguments = [
    "--InlineBackend.figure_formats={'svg', 'pdf'}",
]
```

8.5.3 Environment Variables

The contents of `os.environ` after the execution of `conf.py` will be passed as environment variables to the kernel. As an example, `MY_DUMMY_VARIABLE` has been set in `conf.py` like this:

```
import os
os.environ['MY_DUMMY_VARIABLE'] = 'Hello from conf.py!'
```

... and it can be checked in the notebook like this:

[1]:
```
import os
os.environ['MY_DUMMY_VARIABLE']
```

[1]:
```
'Hello from conf.py!'
```

This is useful if we want to edit `PYTHONPATH` in order to compile the documentation without installing the project:

```
import os

src = os.path.abspath('../src')
os.environ['PYTHONPATH'] = src
```

If you are using `https://mybinder.org/` and you want to define environment variables, you should create a file `.binder/start` in your repository (see Binder docs[164]) containing definitions like this:

```
#!/bin/bash
export MY_DUMMY_VARIABLE="Hello from .binder/start!"
exec "$@"
```

doc/configuring-kernels.ipynb ends here.

[164] https://mybinder.readthedocs.io/en/latest/using/config_files.html#start-run-code-before-the-user-sessions-starts

The following section was generated from doc/timeout.ipynb

8.6 Cell Execution Timeout

By default, code cells will be executed until they are finished, even if that takes a very long time. In some cases they might never finish.

If you would like to only use a finite amount of time per cell, you can choose a 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 (which is the default).

Alternatively, you can manually execute the notebook in question and save the results, see *the pre-executed example notebook* (page 44).

doc/timeout.ipynb ends here.

doc/executing-notebooks.ipynb ends here.

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

9 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[165] markup. Particularly, the only[166] and raw[167] directives can be used to have different content for HTML and LaTeX output.

Those strings are also processed by the Jinja2[168] templating engine. This means you can run Python-like code within those strings. You have access to the current Sphinx build environment[169] 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[170] for more information.

Warning

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

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

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:

[165] https://www.sphinx-doc.org/rest.html
[166] https://www.sphinx-doc.org/en/master/usage/restructuredtext/directives.html#directive-only
[167] https://docutils.sourceforge.io/docs/ref/rst/directives.html#raw-data-pass-through
[168] https://jinja.palletsprojects.com/
[169] https://www.sphinx-doc.org/en/master/extdev/envapi.html
[170] https://jinja.palletsprojects.com/templates/

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

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

An alternative work-around would be to surround LaTeX braces with Jinja braces like this:

```
{{ '{' }}
```

The string within will not be touched by Jinja.

Another special Jinja syntax is {%, which is also often used in fancy TeX/LaTeX code. A work-around for this situation would be to use

```
{{ '{%' }}
```

9.1 Examples

You can include a simple static string, using reST[171] markup if you like:

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

Generated by nbsphinx_ from a Jupyter_ notebook.

.. _nbsphinx: https://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 = r"""
{% set docname = env.doc2path(env.docname, base=None) %}

.. only:: html

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

.. raw:: latex

    \nbsphinxstartnotebook{The following section was created from
    \texttt{\strut{}{{ docname }}}:}
"""

nbsphinx_epilog = r"""
.. raw:: latex

    \nbsphinxstopnotebook{\hfill End of notebook.}
"""
```

[171] https://www.sphinx-doc.org/rest.html

Note the use of the `\nbsphinxstartnotebook` and `\nbsphinxstopnotebook` commands. Those make sure there is not too much space between the "prolog" and the beginning of the notebook and, respectively, between the end of the notebook and the "epilog". They also avoid page breaks, in order for the "prolog"/"epilog" not to end up on the page before/after the notebook.

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

`doc/prolog-and-epilog.ipynb` ends here.

The following section was generated from `doc/custom-formats.pct.py`

10 Custom Notebook Formats

By default, Jupyter notebooks are stored in files with the suffix `.ipynb`, which use the JSON format for storage.

However, there are libraries available which allow storing notebooks in different formats, using different file suffixes.

To use a custom notebook format in `nbsphinx`, you can specify the `nbsphinx_custom_formats` option in your `conf.py` file. You have to provide the file extension and a conversion function that takes the contents of a file (as a string) and returns a Jupyter notebook object.

```
nbsphinx_custom_formats = {
    '.mysuffix': 'mylibrary.converter_function',
}
```

The converter function can be given as a string (recommended) or as a function object.

If a conversion function takes more than a single string argument, you can specify the function name plus a dictionary with keyword arguments which will be passed to the conversion function in addition to the file contents.

```
nbsphinx_custom_formats = {
    '.mysuffix': ['mylibrary.converter_function', {'some_arg': 42}],
}
```

You can of course use multiple formats by specifying multiple conversion functions.

10.1 Example: Jupytext

One example for a library which provides a custom conversion function is jupytext[172], which allows storing the contents of Jupyter notebooks in Markdown and R-Markdown, as well as plain Julia, Python and R files.

Since its conversion function takes more than a single string argument, we have to pass a keyword argument, e.g.:

```
nbsphinx_custom_formats = {
    '.Rmd': ['jupytext.reads', {'fmt': 'Rmd'}],
}
```

This very page is an example of a notebook stored in the `py:percent` format (see docs[173]):

[1]:
```
!head -20 custom-formats.pct.py
```

[172] https://github.com/mwouts/jupytext
[173] https://jupytext.readthedocs.io/en/latest/formats-scripts.html#the-percent-format

```
# %% [markdown]
# # Custom Notebook Formats
#
# By default, Jupyter notebooks are stored in files with the suffix `.ipynb`,
# which use the JSON format for storage.
#
# However, there are libraries available which allow storing notebooks
# in different formats, using different file suffixes.
#
# To use a custom notebook format in `nbsphinx`, you can specify the
# `nbsphinx_custom_formats` option in your `conf.py` file.
# You have to provide the file extension
# and a conversion function that takes the contents of a file (as a string)
# and returns a Jupyter notebook object.
#
# ```python
# nbsphinx_custom_formats = {
#     '.mysuffix': 'mylibrary.converter_function',
# }
# ```
```

To select a suitable conversion function, we use the following setting in `conf.py`:

```
nbsphinx_custom_formats = {
    '.pct.py': ['jupytext.reads', {'fmt': 'py:percent'}],
    '.md': ['jupytext.reads', {'fmt': 'Rmd'}],
}
```

Another example is *this gallery example page* (page 67).

doc/custom-formats.pct.py ends here.

The following section was generated from doc/subdir/a-notebook-in-a-subdir.ipynb

11 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:

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

[1]:

Warning

There may be problems with images in output cells if your source directory contains symbolic links, see issue #49[174].

A link to a notebook in the same sub-directory: *link* (page 60).

A link to a notebook in the parent directory: *link* (page 19).

A link to a local file: link.

A random equation:

$$F_n = F_{n-1} + F_{n-2} \tag{08.15}$$

11.1 A Sub-Section

This is just for testing inter-notebook links, see *this section* (page 27).

11.2 That's a "Strange" Section

This is for testing links to a section title containing quotes.

`doc/subdir/a-notebook-in-a-subdir.ipynb` ends here.

The following section was generated from `doc/subdir/gallery.ipynb`

12 Creating Thumbnail Galleries

Inspired by Sphinx-Gallery[175], you can create thumbnail galleries from a list of Jupyter notebooks (or other Sphinx source files).

`nbsphinx` provides CSS styles for galleries, but like all styles you can tweak them with your own CSS files loaded via html_css_files[176]. If you want to disable all the original styling, you can create a file named `nbsphinx-gallery.css` somewhere in your html_static_path[177], which will replace the original CSS file (adding it to `html_css_files` is not necessary in this case, because it is automatically added by `nbsphinx`).

The following sections present two different ways of creating thumbnail galleries in Jupyter notebooks and show how thumbnail images can be selected. Thumbnail galleries can also be created in *reStructuredText files* (page 67).

The following section was generated from `doc/gallery/gallery-with-nested-documents.ipynb`

12.1 Gallery With Nested Documents

You can create *thumbnail galleries in reST files* (page 67), but you can also create such galleries in Jupyter notebooks by adding the `nbsphinx-gallery` cell tag or metadata, which is used just like the *nbsphinx-toctree* (page 60) cell tag/metadata. For possible options, see the *toctree* (page 60) notebook.

[174] https://github.com/spatialaudio/nbsphinx/issues/49
[175] https://sphinx-gallery.github.io/
[176] https://www.sphinx-doc.org/en/master/usage/configuration.html#confval-html_css_files
[177] https://www.sphinx-doc.org/en/master/usage/configuration.html#confval-html_static_path

Note

In LaTeX output this behaves just like `toctree`, i.e. no thumbnail gallery is shown, but the linked files are included in the document.

Like with `toctree` you should avoid adding content after a gallery (except other toctrees and galleries) because this content would appear in the LaTeX output *after* the content of all included source files, which is probably not what you want.

The following cell has the `nbsphinx-gallery` tag, which creates a thumbnail gallery. The *first* section title in that cell (if available) is used as `caption` (unless it is already given in the metadata).

The notebooks in the following gallery describe different ways how to select which images are used as thumbnails. The notebooks are added as sub-sections under the current section, just like when using `toctree`. If you want to create a gallery from links to notebooks that are already included somewhere else, you can use *nbsphinx-link-gallery* (page 60).

The following section was generated from `doc/gallery/default-thumbnail.ipynb`

12.1.1 Default Thumbnail

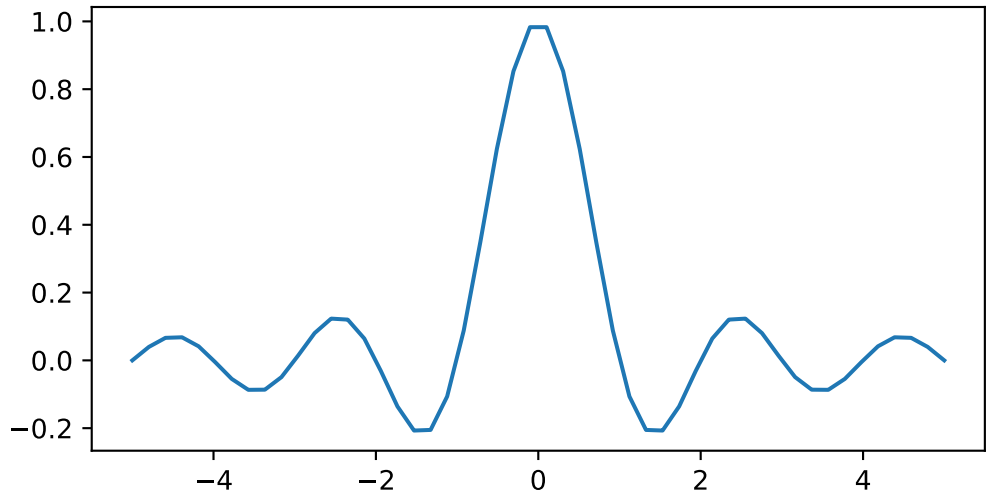
By default, the last image output of a notebook will be used as its thumbnail. Without an image output, a placeholder will be used. See *a notebook with no thumbnail* (page 58) for an example.

However, if a thumbnail is explicitly assigned by *Using Cell Metadata to Select a Thumbnail* (page 56), *Using a Cell Tag to Select a Thumbnail* (page 55) or *Specifying a Thumbnail File* (page 58), these methods will take precedence.

[1]:
```
import matplotlib.pyplot as plt
import numpy as np
```

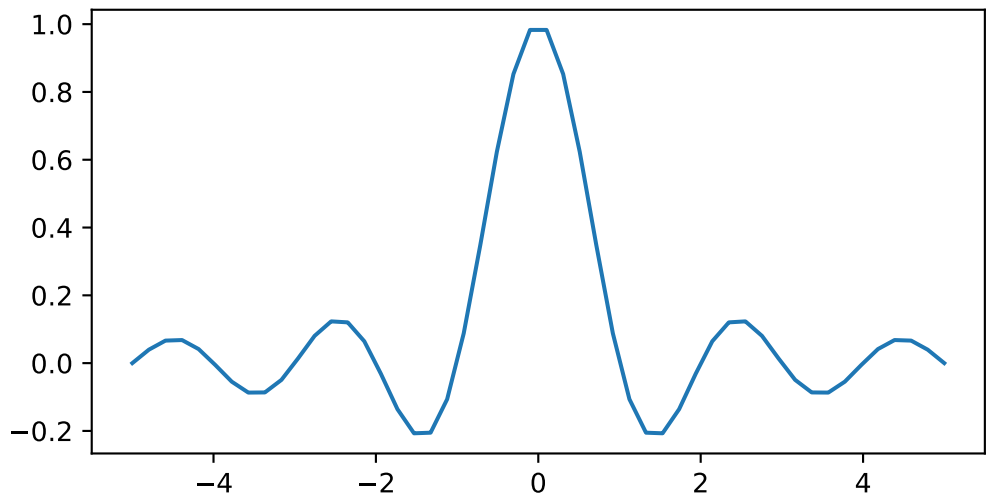
Although the next cell contains an image (a plot), it won't be used as the thumbnail because it's not the last in the notebook, and we haven't explicitly tagged it.

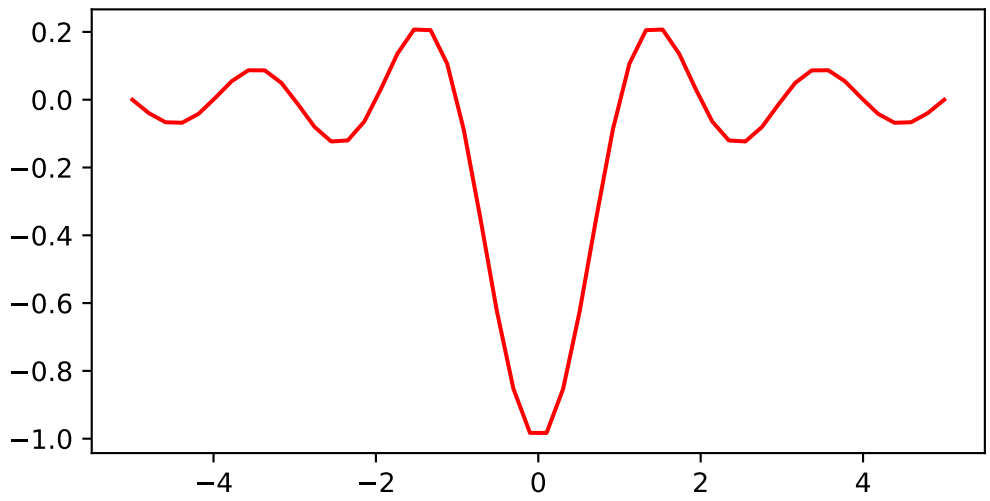
[2]:
```
fig, ax = plt.subplots(figsize=[6, 3])
x = np.linspace(-5, 5, 50)
ax.plot(x, np.sinc(x));
```



But the next cell is the last containing an image in the notebook, so its last image output will be used as the thumbnail.

[3]:
```
display(fig)
fig, ax = plt.subplots(figsize=[6, 3])
x = np.linspace(-5, 5, 50)
ax.plot(x, -np.sinc(x), color='red');
```





doc/gallery/default-thumbnail.ipynb ends here.

The following section was generated from doc/gallery/cell-tag.ipynb

12.1.2 Using a Cell Tag to Select a Thumbnail

You can select any code cell (with appropriate output) by tagging it with the `nbsphinx-thumbnail` tag.

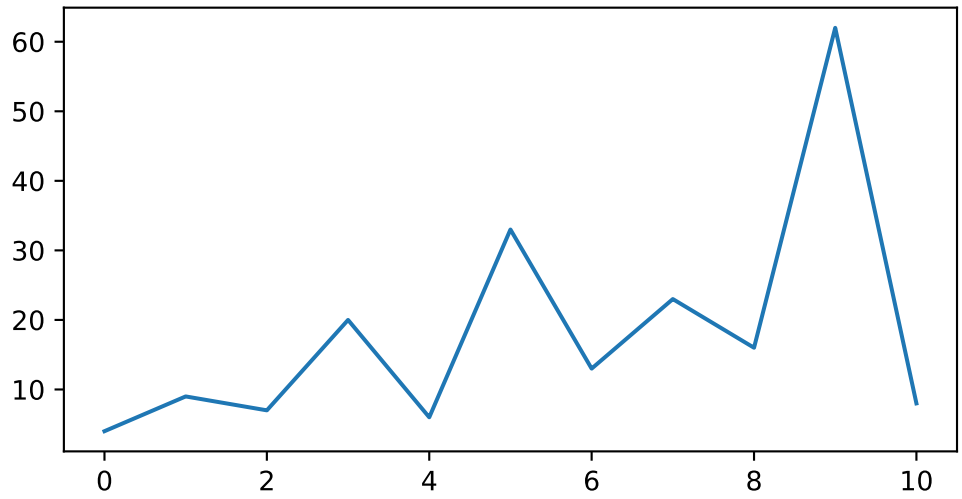
If there are multiple outputs in the selected cell, the last one is used. See *Choosing from Multiple Outputs* (page 57) for how to select a specific output. If you want to show a tooltip, have a look at *Using Cell Metadata to Select a Thumbnail* (page 56).

[1]:
```
import matplotlib.pyplot as plt
```

The following cell has the `nbsphinx-thumbnail` tag, which will take precedence over the default of the last image in the notebook:

[2]:
```
fig, ax = plt.subplots(figsize=[6, 3])
ax.plot([4, 9, 7, 20, 6, 33, 13, 23, 16, 62, 8])
```

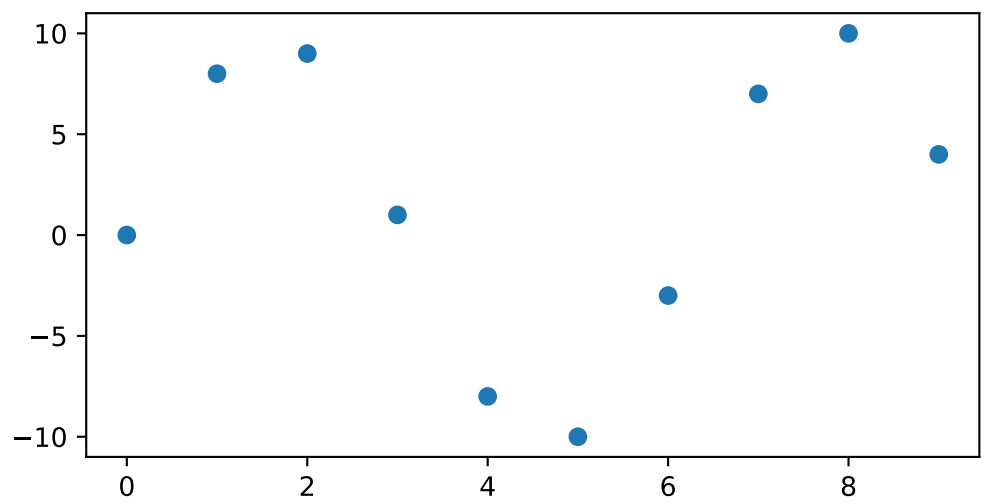
[2]:
```
[<matplotlib.lines.Line2D at 0x7fabddb72c50>]
```



Although the next cell has an image, it won't be used as the thumbnail, due to the tag on the one above.

[3]:
```
fig, ax = plt.subplots(figsize=[6, 3])
ax.scatter(range(10), [0, 8, 9, 1, -8, -10, -3, 7, 10, 4])
```

[3]:
```
<matplotlib.collections.PathCollection at 0x7fabddc4a890>
```



doc/gallery/cell-tag.ipynb ends here.

The following section was generated from doc/gallery/cell-metadata.ipynb

12.1.3 Using Cell Metadata to Select a Thumbnail

If the *nbsphinx-thumbnail* (page 55) cell tag is not enough, you can use cell metadata to specify more options.

The last cell in this notebook has this metadata:

```
{
    "nbsphinx-thumbnail": {
        "tooltip": "This tooltip message was defined in cell metadata"
    }
}
```

If there are multiple outputs in the selected cell, the last one is used. See *Choosing from Multiple Outputs* (page 57) for how to select a specific output.

[1]:
```
import matplotlib.pyplot as plt
import numpy as np
```

[2]:
```
plt.rcParams['image.cmap'] = 'coolwarm'
plt.rcParams['image.origin'] = 'lower'
```

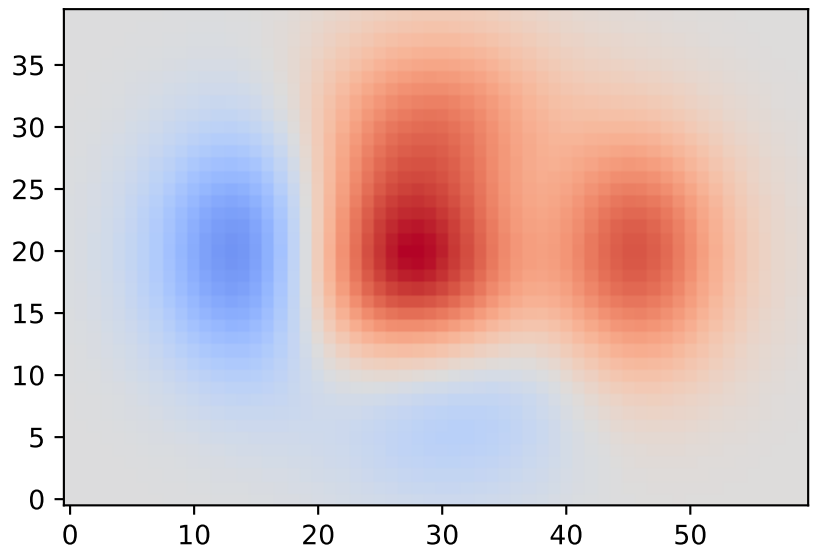
Some example data stolen from `https://matplotlib.org/examples/pylab_examples/pcolor_demo.html`:

[3]:
```
x, y = np.meshgrid(np.arange(-3, 3, 0.1), np.arange(-2, 2, 0.1))
z = (1 - x / 2 + x ** 5 + y ** 3) * np.exp(-x ** 2 - y ** 2)
```

[4]:
```
zmax = np.max(np.abs(z))
```

[5]:
```
fig, ax = plt.subplots(figsize=[5, 3.5])
ax.imshow(z, vmin=-zmax, vmax=zmax)
```

[5]:
```
<matplotlib.image.AxesImage at 0x7fa6ddcf3a50>
```



..... doc/gallery/cell-metadata.ipynb ends here.

The following section was generated from `doc/gallery/multiple-outputs.ipynb` .....

12.1.4 Choosing from Multiple Outputs

By default, the last output of the selected cell is used as a thumbnail. If that's what you want, you can simply use the *nbsphinx-thumbnail* (page 55) cell tag.

If you want to specify one of multiple outputs, you can add a (zero-based) `"output-index"` to your `"nbsphinx-thumbnail"` cell metadata.

The following cell has this metadata, selecting the third output to be used as thumbnail in *the gallery* (page 53).

```
{
    "nbsphinx-thumbnail": {
        "output-index": 2
```

(continues on next page)

(continued from previous page)

```
    }
}
```

[1]:
```
from IPython.display import Image

display(Image(url='https://jupyter.org/assets/homepage/main-logo.svg'))
print('Hello!')
display(Image(filename='../images/notebook_icon.png'))
display(Image(url='https://www.python.org/static/img/python-logo-large.png', embed=True))
```

```
<IPython.core.display.Image object>
```

```
Hello!
```

doc/gallery/multiple-outputs.ipynb ends here.

The following section was generated from doc/gallery/no-thumbnail.ipynb

12.1.5 A Notebook without Thumbnail

This notebook doesn't contain any thumbnail metadata.

It should be displayed with the default thumbnail image in the *gallery* (page 53).

doc/gallery/no-thumbnail.ipynb ends here.

The following section was generated from doc/gallery/thumbnail-from-conf-py.ipynb

12.1.6 Specifying Thumbnails in `conf.py`

This notebook doesn't contain a `nbsphinx-thumbnail` *cell tag* (page 55) nor *cell metadata* (page 56). Instead, in the file conf.py, a thumbnail is specified (via the *nbsphinx_thumbnails* (page 19) option), which will be used in the *gallery* (page 53).

The keys in the `nbsphinx_thumbnails` dictionary can contain wildcards, which behave very similarly to the html_sidebars[178] option.

The thumbnail files can be local image files somewhere in the source directory, but you'll need to create at least one *link* (page 28) to them in order to copy them to the HTML output directory.

You can also use files from the `_static` directory (which contains all files in your html_static_path[179]).

If you want, you can also use files from the `_images` directory, which contains all notebook outputs.

To demonstrate this feature, we are creating an image file here:

[1]:
```
import matplotlib.pyplot as plt
```

[2]:
```
fig, ax = plt.subplots()
ax.plot([4, 8, 15, 16, 23, 42])
fig.savefig('a-local-file.png')
plt.close()  # avoid plotting the figure
```

Please note that the previous cell doesn't have any outputs, but it has generated a file named `a-local-file.png` in the notebook's directory.

We have to create a link to this file (which is a good idea anyway): `a-local-file.png`.

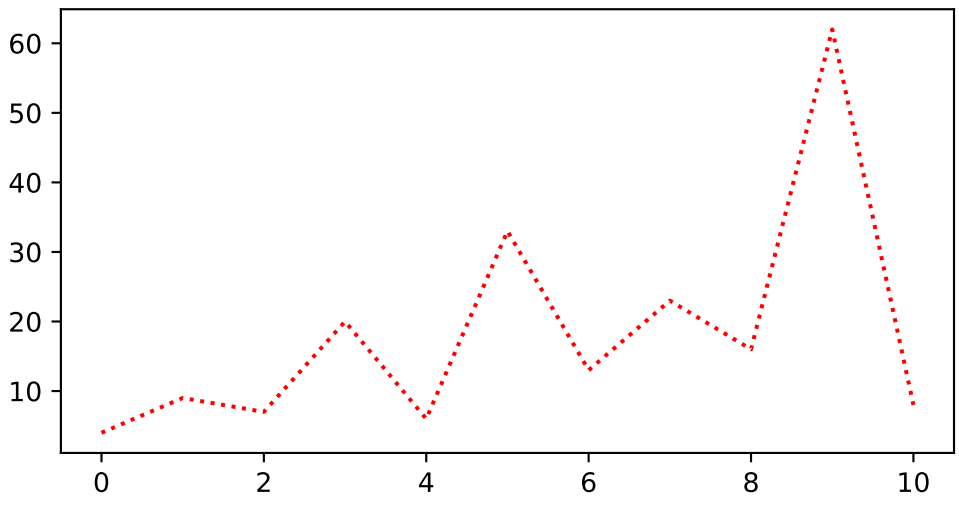
Now we can use this file in our conf.py like this:

```
nbsphinx_thumbnails = {
    'gallery/thumbnail-from-conf-py': 'gallery/a-local-file.png',
}
```

Please note that the notebook name does *not* contain the `.ipynb` suffix.

Note that the following plot is *not* used as a thumbnail because the `nbsphinx_thumbnails` setting overrides *the default behavior* (page 54).

[3]:
```
fig, ax = plt.subplots(figsize=[6, 3])
ax.plot([4, 9, 7, 20, 6, 33, 13, 23, 16, 62, 8], 'r:');
```



doc/gallery/thumbnail-from-conf-py.ipynb ends here.
doc/gallery/gallery-with-nested-documents.ipynb ends here.

[178] https://www.sphinx-doc.org/en/master/usage/configuration.html#confval-html_sidebars
[179] https://www.sphinx-doc.org/en/master/usage/configuration.html#confval-html_static_path

The following section was generated from doc/gallery/gallery-with-links.ipynb

12.2 Gallery With Links (HTML only)

Contrary to *nbsphinx-gallery* (page 53), the cell tag/metadata `nbsphinx-link-gallery` creates a gallery from notebooks (and other source) files without including them as sub-sections. Other than that, it works in a similar way, but only the options `"name"`, `"caption"` and `"reversed"` are supported. In LaTeX output, this has no effect. The cell is ignored and nothing is added to the LaTeX document.

In reST files (and *raw reST cells* (page 42)), the *nblinkgallery* (page 67) directive can be used.

The following Markdown cell has the `nbsphinx-link-gallery` tag, which turns the contained links into a gallery and uses the first section title as `caption`:

doc/gallery/gallery-with-links.ipynb ends here.

doc/subdir/gallery.ipynb ends here.

The following section was generated from doc/subdir/toctree.ipynb

13 Using `toctree` In A Notebook

In Sphinx-based documentation, there is typically a file called `index.rst` which contains one or more toctree[180] directives. Those can be used to pull in further source files (which themselves can contain further `toctree` directives).

With `nbsphinx` it is possible to get a similar effect within a Jupyter notebook using the `nbsphinx-toctree` cell tag or cell metadata. Markdown cells with `nbsphinx-toctree` tag/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 selected cell, it is used as `toctree` caption (but it also works without a title).

Note

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

If you are satisfied with the default settings, you can simply use `nbsphinx-toctree` as a cell tag.

Alternatively, you can store `nbsphinx-toctree` cell metadata. Use ...

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

... for the default settings, ...

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

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

[180] https://www.sphinx-doc.org/en/master/usage/restructuredtext/directives.html#directive-toctree

```
{
  "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 a the Sphinx documentation[181]. All options can be used – except `:glob:`, which can only be used in *rst files* (page 63) and in *raw reST cells* (page 42).

Note

In 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` and contains a link to the notebook *yet-another.ipynb* (page 61) 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 (which also will only be visible in the HTML output).

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

13.1 Yet Another Notebook

This notebook is only here to show how (sub-)toctrees can be created with Markdown cell metadata. See *there* (page 60).

`doc/yet-another.ipynb` ends here.

`doc/subdir/toctree.ipynb` ends here.

The following section was generated from `doc/custom-css.ipynb`

14 Custom CSS

If you are not satisfied with the CSS styles provided by `nbsphinx` and by your Sphinx theme, don't worry, you can add your own styles easily.

[181] `https://www.sphinx-doc.org/en/master/usage/restructuredtext/directives.html#directive-toctree`

14.1 For All Pages

Just create your own CSS file, e.g. `my-own-style.css`, and put it into the `_static/` sub-directory of your source directory.

You'll also have to set the config values html_static_path[182] and html_css_files[183] in your `conf.py`, e.g. like this:

```
html_static_path = ['_static']
html_css_files = ['my-own-style.css']
```

14.2 For All RST files

If you want your style to only apply to `*.rst` files (and not Jupyter notebooks or other source files), you can use rst_prolog[184] with the raw[185] directive in your `conf.py` like this:

```
rst_prolog = """
.. raw:: html

    <style>
        h1 {
            color: fuchsia;
        }
    </style>
"""
```

14.3 For All Notebooks

Similarly, if you want your style to only apply to notebooks, you can use *nbsphinx_prolog* (page 49) like this:

```
nbsphinx_prolog = """
.. raw:: html

    <style>
        h1 {
            color: chartreuse;
        }
    </style>
"""
```

[182] https://www.sphinx-doc.org/en/master/usage/configuration.html#confval-html_static_path
[183] https://www.sphinx-doc.org/en/master/usage/configuration.html#confval-html_css_files
[184] https://www.sphinx-doc.org/en/master/usage/configuration.html#confval-rst_prolog
[185] https://docutils.sourceforge.io/docs/ref/rst/directives.html#raw-data-pass-through

14.4 For a Single Notebook

For styles that should affect only the current notebook, you can simply insert `<style>` tags into Markdown cells like this:

```
<style>
    .nbinput .prompt,
    .nboutput .prompt {
        display: none;
    }
</style>
```

This CSS example removes the input and output prompts from code cells, see the following cell:

[1]:
```
6 * 7
```

[1]:
```
42
```

..... doc/custom-css.ipynb ends here.

15 Normal reStructuredText Files

This is a normal RST file.

> **Note:** Those still work!

15.1 Links to Notebooks (and Other Sphinx Source Files)

Links to Sphinx source files can be created like normal Sphinx hyperlinks[186], just using a relative path to the local file: *link* (page 52).

```
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* (page 52).

```
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[187], like this: *link* (page 52). 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[188], like this *link* (page 52).

[186] https://www.sphinx-doc.org/en/master/usage/restructuredtext/basics.html#external-links
[187] https://docutils.sourceforge.io/docs/ref/rst/restructuredtext.html#anonymous-hyperlinks
[188] https://docutils.sourceforge.io/docs/ref/rst/restructuredtext.html#embedded-uris-and-aliases

```
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* (page 53).

```
For example, see this subsection_.

.. _subsection: subdir/a-notebook-in-a-subdir.ipynb#A-Sub-Section
```

15.2 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:[189]. This has some disadvantages:

- It is arguably a bit more clunky.
- 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* (page 52).

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

But you can also provide *your own link title* (page 52).

```
: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)* (page 63).

Links to subsections are also possible, e.g. *A Sub-Section* (page 53) (the subsection title is used as link text) and *alternative text* (page 53).

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!

[189] https://www.sphinx-doc.org/en/master/usage/restructuredtext/roles.html#role-ref

15.3 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!

15.4 Domain Objects

`example_python_function`(*foo*)

This is just for testing domain object links.

Parameters
`foo` (*str*[190]) – Example string parameter

See also:

Links to Domain Objects (page 28)

15.5 References

There are different ways of handling references, for example you could use the standard Sphinx citations[191], but it might be more practical to use the sphinxcontrib.bibtex[192] extension.

After installing the sphinxcontrib.bibtex[193] extension, you have to enable it in your `conf.py` and select the BibTeX file(s) you want to use:

```
extensions = [
    'nbsphinx',
    'sphinxcontrib.bibtex',
    # Probably more extensions here ...
]

bibtex_bibfiles = ['my-references.bib']
bibtex_reference_style = 'author_year'
```

[190] https://docs.python.org/3/library/stdtypes.html#str
[191] https://www.sphinx-doc.org/en/master/usage/restructuredtext/basics.html#citations
[192] https://sphinxcontrib-bibtex.readthedocs.io/
[193] https://sphinxcontrib-bibtex.readthedocs.io/

Afterwards all the references defined in the bibliography file(s) can be used throughout the Jupyter notebooks and other source files as detailed in the following.

15.5.1 Citations

You can create citations like [Pérez *et al.*, 2011]:

```
:cite:`perez2011python`
```

You can also create so-called in-text citations, where the names of the authors, for example Pérez *et al.* [2011], are part of the sentence:

```
:cite:t:`perez2011python`
```

You can create similar citations in Jupyter notebooks with a special HTML syntax, see the section about *citations in Markdown cells* (page 22).

You can create a list of references in any reStructuredText file (or *reST cell* (page 42) in a notebook) like this:

```
.. bibliography::
```

For an example, see the file `doc/references.rst`.

15.5.2 Footnote citations

With a sphinxcontrib.bibtex[194] version of `>= 2.0.0` it is possible to create footnote bibliographies with footnote citations like[195].

```
:footcite:`perez2011python`
```

In-text citations like Kluyver *et al.*[197] can be created like this:

```
:footcite:t:`kluyver2016jupyter`
```

Also footnote citations can be used within Jupyter notebooks with a special HTML syntax, see the section about *footnote citations in Markdown cells* (page 22). Footnote citations are restricted to their own source file and the assembly of the bibliography is (analogously to normal citations) invoked with the

```
.. footbibliography::
```

directive. For example, a footnote bibliography might look like this (in HTML output):

In the LaTeX/PDF output, there is no list of references appearing right here. Instead, the footnote citations are placed into the footnotes of their respective pages.

[194] `https://sphinxcontrib-bibtex.readthedocs.io/`

[195] Fernando Pérez, Brian E. Granger, and John D. Hunter. Python: an ecosystem for scientific computing. *Computing in Science Engineering*, 13(2):13–21, 2011. doi:10.1109/MCSE.2010.119[Page 66, 196].

[196] `https://doi.org/10.1109/MCSE.2010.119`

[197] Thomas Kluyver, Benjamin Ragan-Kelley, Fernando Pérez, Brian Granger, Matthias Bussonnier, Jonathan Frederic, Kyle Kelley, Jessica Hamrick, Jason Grout, Sylvain Corlay, Paul Ivanov, Damián Avila, Safia Abdalla, Carol Willing, and Jupyter Development Team. Jupyter Notebooks—a publishing format for reproducible computational workflows. In Fernando Loizides and Birgit Schmidt, editors, *Positioning and Power in Academic Publishing: Players, Agents and Agendas*, pages 87–90. IOS Press, 2016. doi:10.3233/978-1-61499-649-1-87[198].

[198] `https://doi.org/10.3233/978-1-61499-649-1-87`

15.6 Thumbnail Link Galleries (HTML only)

In some case it is desired to create thumbnail links to existing notebooks, already included in a `toctree`. This can be used e.g. to link to a subset of notebooks from API documentation to highlight the use of some functionality. For this there is a dedicated `nblinkgallery` directive.

The following example gallery was created using:

```
.. nblinkgallery::
    :caption: A few links
    :name: rst-link-gallery

    gallery/multiple-outputs
    gallery/no-thumbnail
    gallery/cell-metadata
    orphan
```

See also:

Link Galleries in Jupyter Notebooks (page 60)

15.7 Thumbnail Galleries

With `nbsphinx` you can create thumbnail galleries in notebook files as described in *Gallery With Nested Documents* (page 53). If you like, you can also create such galleries in reST files using the `nbgallery` directive. It takes the same parameters as the toctree[199] directive.

Note: The notes regarding LaTeX in *Gallery With Nested Documents* (page 53) and *Using toctree In A Notebook* (page 60) also apply here!

The following example gallery was created using:

```
.. nbgallery::
    :caption: This is a thumbnail gallery:
    :name: rst-gallery
    :glob:
    :reversed:

    gallery/*-rst
```

The following section was generated from `doc/gallery/uno-rst.ipynb`

15.7.1 Dummy Notebook 1 for Gallery

This is a dummy file just to fill *the gallery in the reST file* (page 67).

The thumbnail image is assigned in conf.py.

`doc/gallery/uno-rst.ipynb` ends here.

The following section was generated from `doc/gallery/due-rst.pct.py`

15.7.2 Dummy Notebook 2 for Gallery

This is a dummy file just to fill *the gallery in the reST file* (page 67).

[199] `https://www.sphinx-doc.org/en/master/usage/restructuredtext/directives.html#directive-toctree`

The thumbnail image is assigned in conf.py.

The source file is, for no particular reason, a Python script adhering to the `py:percent` format. It is parsed with the help of Jupytext[200], see *Custom Notebook Formats* (page 51).

[1]:
```
from pathlib import Path
```

[2]:
```
filename = 'due-rst.pct.py'

print(Path(filename).read_text())
```

```
# %% [markdown]
# # Dummy Notebook 2 for Gallery
#
# This is a dummy file just to fill
# [the gallery in the reST file](../a-normal-rst-file.rst#thumbnail-galleries).
#
# The thumbnail image is assigned in [conf.py](../conf.py).

# %% [markdown]
# The source file is, for no particular reason,
# a Python script adhering to the `py:percent` format.
# It is parsed with the help of [Jupytext](https://jupytext.readthedocs.io/),
# see [Custom Notebook Formats](../custom-formats.ipynb).

# %%
from pathlib import Path

# %%
filename = 'due-rst.pct.py'

print(Path(filename).read_text())
```

doc/gallery/due-rst.pct.py ends here.

The following section was generated from doc/a-markdown-file.md

16 Using Markdown Files

Sphinx on its own doesn't know how to handle Markdown files, but there are extensions that enable their usage as Sphinx source files. For an example, see the Sphinx documentation[201].

Alternatively, when using `nbsphinx` it is also possible to use Markdown files via *custom notebook formats* (page 51).

You only need to install the jupytext[202] package and add a configuration setting to `conf.py`, which can be used to select one of several Markdown flavors supported by jupytext[203] (here we are using R Markdown):

```
nbsphinx_custom_formats = {
    '.md': ['jupytext.reads', {'fmt': 'Rmd'}],
}
```

This very page was generated from a Markdown file using these settings.

[200] https://jupytext.readthedocs.io/
[201] https://www.sphinx-doc.org/en/master/usage/markdown.html
[202] https://jupytext.readthedocs.io/
[203] https://jupytext.readthedocs.io/en/latest/formats-markdown.html

16.1 Links to Notebooks (and Other Sphinx Source Files)

Links to other Sphinx source files can be created like in *Markdown cells of notebooks* (page 27).

16.2 Math

Mathematical equations can be used just like in *Markdown cells of notebooks* (page 20).

Inline like this: $e^{i\pi} = -1$.

Or as a separate block:

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

16.3 Tables

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

16.4 Images

`doc/a-markdown-file.md` ends here.

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

17 External Links

nbconvert

The official conversion tool of the Jupyter project. It can be used to convert notebooks to HTML, LaTeX and many other formats.

Its `--execute` flag can be used to automatically execute notebooks before conversion.

`https://nbconvert.readthedocs.io/`

`https://github.com/jupyter/nbconvert`

RunNotebook (notebook_sphinxext.py)

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

No LaTeX support.

https://github.com/ngoldbaum/RunNotebook

There are some forks:

- `https://bitbucket.org/yt_analysis/yt-doc/src/default/extensions/notebook_sphinxext.py` (not available anymore)
- https://github.com/matthew-brett/perrin-academy/blob/master/sphinxext/notebook_sphinxext.py

nbsite

Build a tested, sphinx-based website from notebooks.

https://nbsite.holoviz.org/

ipypublish

A workflow for creating and editing publication ready scientific reports and presentations, from one or more Jupyter Notebooks, without leaving the browser!

https://ipypublish.readthedocs.io/

https://github.com/chrisjsewell/ipypublish

jupyterbook

Jupyter Book is an open source project for building beautiful, publication-quality books and documents from computational material.

https://jupyterbook.org/

https://github.com/executablebooks/jupyter-book

MyST-NB

A collection of tools for working with Jupyter Notebooks in Sphinx.

The primary tool this package provides is a Sphinx parser for `ipynb` files. This allows you to directly convert Jupyter Notebooks into Sphinx documents. It relies heavily on the MyST parser[204].

https://myst-nb.readthedocs.io/

https://github.com/executablebooks/MyST-NB

notebook-to-pdf

This Jupyter notebook extension allows you to save your notebook as a PDF.

Three new features compared to the official "save as PDF" extension:

1. produce a PDF with the smallest number of page breaks,
2. the original notebook is attached to the PDF; and
3. this extension does not require LaTex.

https://github.com/betatim/notebook-as-pdf

nbinteract

Create interactive webpages from Jupyter Notebooks

https://www.nbinteract.com/

[204] `https://github.com/executablebooks/MyST-Parser`

https://github.com/SamLau95/nbinteract

nb_pdf_template

An extended `nbconvert` template for LaTeX output.

https://github.com/t-makaro/nb_pdf_template

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

https://sphinx-gallery.readthedocs.io/

sphinx-nbexamples

https://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.

bookbook

Uses `nbconvert` to create a sequence of HTML or a concatenated LaTeX file from a sequence of notebooks.

https://github.com/takluyver/bookbook

jupyter-sphinx

Jupyter Sphinx is a Sphinx extension that executes embedded code in a Jupyter kernel, and embeds outputs of that code in the output document. It has support for rich output such as images, Latex math and even javascript widgets.

https://jupyter-sphinx.readthedocs.io/

https://github.com/jupyter/jupyter-sphinx

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 (not available anymore)

https://sphinx-ipynb.readthedocs.io/

Converting reStructuredText to Notebooks

https://github.com/nthiery/rst-to-ipynb

https://github.com/QuantEcon/sphinxcontrib-jupyter

Converting Notebooks to HTML for Blog Posts

http://dongweiming.github.io/divingintoipynb_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)

doc/links.ipynb ends here.

18 Contributing

If you find bugs, errors, omissions or other things that need improvement, please create an issue or a pull request at https://github.com/spatialaudio/nbsphinx/. Contributions are always welcome!

18.1 Development Installation

Make sure that the necessary prerequisites[205] are installed. Then, instead of `pip`-installing the latest release from PyPI[206], you should get the newest development version (a.k.a. "master") with Git:

```
git clone https://github.com/spatialaudio/nbsphinx.git
cd nbsphinx
python3 -m pip install -e .
```

... where `-e` stands for `--editable`.

When installing this way, you can quickly try other Git branches (in this example the branch is called "another-branch"):

```
git checkout another-branch
```

If you want to go back to the "master" branch, use:

```
git checkout master
```

To get the latest changes from Github, use:

```
git pull --ff-only
```

18.2 Building the Documentation

If you make changes to the documentation, you should create the HTML pages locally using Sphinx and check if they look OK.

Initially, you might need to install a few packages that are needed to build the documentation:

```
python3 -m pip install -r doc/requirements.txt
```

To (re-)build the HTML files, use:

[205] https://nbsphinx.readthedocs.io/installation.html#nbsphinx-Prerequisites
[206] https://pypi.org/project/nbsphinx/

```
python3 setup.py build_sphinx
```

If you want to check the LaTeX output, use:

```
python3 setup.py build_sphinx -b latex
```

Again, you'll probably have to use `python` instead of `python3`. The generated files will be available in the directories `build/sphinx/html/` and `build/sphinx/latex/`, respectively.

18.3 Building Themes

The `nbsphinx` documentation is available in over 30 different HTML themes[207], with each having its own branch ending in `-theme`.

To simplify the building and testing of themes, which is especially needed when changing CSS, we provide you with command line tool to build all themes or a user specified subset. The tool is located at `theme_comparison.py` and can be run with:

```
python3 theme_comparison.py
```

Before doing that, the required dependencies can be obtained with:

```
python3 theme_comparison.py --requirements
```

This will create a list of dependencies in `theme_comparison/theme_requirements.txt`. The dependencies can then be installed with:

```
python3 -m pip install -r theme_comparison/theme_requirements.txt
```

If you just want to build a subset of the themes (e.g. `alabaster` and `sphinx_rtd_theme`), simply run:

```
python3 theme_comparison.py alabaster rtd
```

For more information run:

```
python3 theme_comparison.py --help
```

18.4 Testing

Unfortunately, the currently available automated tests are very limited. Contributions to improve the testing situation are of course also welcome!

The `nbsphinx` documentation also serves as a test case. However, the resulting HTML/LaTeX/PDF files have to be inspected manually to check whether they look as expected.

Sphinx's warnings can help spot problems, therefore it is recommended to use the `-W` flag to turn Sphinx warnings into errors while testing:

```
python3 setup.py build_sphinx -W
```

This flag is also used for continuous integration on Github Actions (see the files `.github/workflows/*.yml`) and CircleCI (see the file `.circleci/config.yml`).

Sphinx has a `linkcheck` builder that can check whether all URLs used in the documentation are still valid. This is also part of the continuous integration setup on CircelCI.

[207] `https://nbsphinx.readthedocs.io/usage.html#HTML-Themes`

19 References

By default, in the LaTeX/PDF output the list of references will not appear here, but instead at the end of the document. For a possible work-around (which is also used here) see https://github.com/mcmtroffaes/sphinxcontrib-bibtex/issues/156.

The list of references may look something like this:

[KRKP+16] Thomas Kluyver, Benjamin Ragan-Kelley, Fernando Pérez, Brian Granger, Matthias Bussonnier, Jonathan Frederic, Kyle Kelley, Jessica Hamrick, Jason Grout, Sylvain Corlay, Paul Ivanov, Damián Avila, Safia Abdalla, Carol Willing, and Jupyter Development Team. Jupyter Notebooks—a publishing format for reproducible computational workflows. In Fernando Loizides and Birgit Schmidt, editors, *Positioning and Power in Academic Publishing: Players, Agents and Agendas*, pages 87–90. IOS Press, 2016. doi:10.3233/978-1-61499-649-1-87[208].

[PGH11] Fernando Pérez, Brian E. Granger, and John D. Hunter. Python: an ecosystem for scientific computing. *Computing in Science Engineering*, 13(2):13–21, 2011. doi:10.1109/MCSE.2010.119[209].

Warning: With `docutils` versions 0.18 and 0.19, the HTML output after the bibliography is broken, see https://github.com/mcmtroffaes/sphinxcontrib-bibtex/issues/309. This problem has been fixed in `docutils` version 0.20.

20 Version History

Version 0.9.3 – 2023-08-27 – PyPI Page 74,[210] **– diff**[211]

- Fix gallery regression in Sphinx 7.2

Version 0.9.2 – 2023-05-24 – PyPI[212] **– diff**[213]

- Improve support for `sphinx_immaterial` theme
- Improve support for links starting with #
- Add support for in-text citations
- LaTeX: Add support for admonition titles

Version 0.9.1 – 2023-03-14 – PyPI[214] **– diff**[215]

- pandoc: disable "smart" option only for version 2.0+

Version 0.9.0 – 2023-03-12 – PyPI[216] **– diff**[217]

- Split `nbsphinx.py` (a Python *module*) into:
  - `nbsphinx/__init__.py` (a Python *package*)
  - A standalone CSS file `nbsphinx-code-cells.css_t`
  - LaTeX style file `nbsphinx.sty`

[208] https://doi.org/10.3233/978-1-61499-649-1-87
[209] https://doi.org/10.1109/MCSE.2010.119
[210] https://pypi.org/project/nbsphinx/0.9.3/
[211] https://github.com/spatialaudio/nbsphinx/compare/0.9.2...0.9.3
[212] https://pypi.org/project/nbsphinx/0.9.2/
[213] https://github.com/spatialaudio/nbsphinx/compare/0.9.1...0.9.2
[214] https://pypi.org/project/nbsphinx/0.9.1/
[215] https://github.com/spatialaudio/nbsphinx/compare/0.9.0...0.9.1
[216] https://pypi.org/project/nbsphinx/0.9.0/
[217] https://github.com/spatialaudio/nbsphinx/compare/0.8.12...0.9.0

- Add custom HTML/CSS for thumbnail galleries
  - Separate CSS file `nbsphinx-gallery.css`
  - The CSS from Sphinx Gallery (`'sphinx_gallery.load_style'`) cannot be used anymore
- Use the last image in a notebook as the default thumbnail

Version 0.8.12 – 2023-01-19 – PyPI[Page 75, 218] **– diff**[219]

- Implement "link" galleries (without nested sub-documents)

Version 0.8.11 – 2022-12-29 – PyPI[220] **– diff**[221]

- LaTeX: apply code cell border style to all code blocks

Version 0.8.10 – 2022-11-13 – PyPI[222] **– diff**[223]

- Fix handling of `source_suffix`
- A few LaTeX fixes

Version 0.8.9 – 2022-06-04 – PyPI[224] **– diff**[225]

- CSS: support tables in widgets
- Avoid empty "raw" directive

Version 0.8.8 – 2021-12-31 – PyPI[226] **– diff**[227]

- Support for the `sphinx_codeautolink` extension
- Basic support for the `text` builder

Version 0.8.7 – 2021-08-10 – PyPI[228] **– diff**[229]

- Fix assertion error in LaTeX build with Sphinx 4.1.0+

Version 0.8.6 – 2021-06-03 – PyPI[230] **– diff**[231]

- Support for Jinja2 version 3

Version 0.8.5 – 2021-05-12 – PyPI[232] **– diff**[233]

- Freeze Jinja2 version to 2.11 (for now, until a bugfix is found)
- Add `theme_comparison.py` tool for creating multiple versions (with different HTML themes) of the docs at once

Version 0.8.4 – 2021-04-29 – PyPI[234] **– diff**[235]

- Support for `mathjax3_config` (for Sphinx >= 4)

[218] https://pypi.org/project/nbsphinx/0.8.12/
[219] https://github.com/spatialaudio/nbsphinx/compare/0.8.11...0.8.12
[220] https://pypi.org/project/nbsphinx/0.8.11/
[221] https://github.com/spatialaudio/nbsphinx/compare/0.8.10...0.8.11
[222] https://pypi.org/project/nbsphinx/0.8.10/
[223] https://github.com/spatialaudio/nbsphinx/compare/0.8.9...0.8.10
[224] https://pypi.org/project/nbsphinx/0.8.9/
[225] https://github.com/spatialaudio/nbsphinx/compare/0.8.8...0.8.9
[226] https://pypi.org/project/nbsphinx/0.8.8/
[227] https://github.com/spatialaudio/nbsphinx/compare/0.8.7...0.8.8
[228] https://pypi.org/project/nbsphinx/0.8.7/
[229] https://github.com/spatialaudio/nbsphinx/compare/0.8.6...0.8.7
[230] https://pypi.org/project/nbsphinx/0.8.6/
[231] https://github.com/spatialaudio/nbsphinx/compare/0.8.5...0.8.6
[232] https://pypi.org/project/nbsphinx/0.8.5/
[233] https://github.com/spatialaudio/nbsphinx/compare/0.8.4...0.8.5
[234] https://pypi.org/project/nbsphinx/0.8.4/
[235] https://github.com/spatialaudio/nbsphinx/compare/0.8.3...0.8.4

- Force loading MathJax on HTML pages generated from notebooks (can be disabled with `nbsphinx_assume_equations = False`)

Version 0.8.3 – 2021-04-09 – PyPI[Page 76, 236] – diff[237]

- Increase `line_length_limit` (for *docutils* 0.17+)

Version 0.8.2 – 2021-02-28 – PyPI[238] – diff[239]

- Add support for `data-footcite` HTML attribute
- Disable automatic highlighting in notebooks, setting `highlight_language` is no longer needed

Version 0.8.1 – 2021-01-18 – PyPI[240] – diff[241]

- Minor fixes and documentation update

Version 0.8.0 – 2020-10-20 – PyPI[242] – diff[243]

- Don't overwrite Pygments background in notebook code cells. To get rid of those ugly greenish code blocks, remove `pygments_style = 'sphinx'` from your `conf.py`.
- Switch documentation to insipid[244] theme by default
- Require Python 3.6+

Version 0.7.1 – 2020-06-16 – PyPI[245] – diff[246]

- Avoid links on scaled images

Version 0.7.0 – 2020-05-08 – PyPI[247] – diff[248]

- Warnings can be suppressed with `suppress_warnings`.
- `<img>` tags are handled in Markdown cells; the `alt`, `width`, `height` and `class` attributes are supported.
- CSS: prompts protrude into left margin if `nbsphinx_prompt_width` is too small. If you want to hide the prompts, use custom CSS[249].

Version 0.6.1 – 2020-04-18 – PyPI[250] – diff[251]

- `.ipynb_checkpoints` is automatically added to `exclude_patterns`

Version 0.6.0 – 2020-04-03 – PyPI[252] – diff[253]

- Thumbnail galleries (inspired by `https://sphinx-gallery.github.io/`)
- `nbsphinx-toctree` as cell tag
- Keyword arguments in `nbsphinx_custom_formats`

[236] https://pypi.org/project/nbsphinx/0.8.3/
[237] https://github.com/spatialaudio/nbsphinx/compare/0.8.2...0.8.3
[238] https://pypi.org/project/nbsphinx/0.8.2/
[239] https://github.com/spatialaudio/nbsphinx/compare/0.8.1...0.8.2
[240] https://pypi.org/project/nbsphinx/0.8.1/
[241] https://github.com/spatialaudio/nbsphinx/compare/0.8.0...0.8.1
[242] https://pypi.org/project/nbsphinx/0.8.0/
[243] https://github.com/spatialaudio/nbsphinx/compare/0.7.1...0.8.0
[244] https://insipid-sphinx-theme.readthedocs.io/
[245] https://pypi.org/project/nbsphinx/0.7.1/
[246] https://github.com/spatialaudio/nbsphinx/compare/0.7.0...0.7.1
[247] https://pypi.org/project/nbsphinx/0.7.0/
[248] https://github.com/spatialaudio/nbsphinx/compare/0.6.1...0.7.0
[249] https://nbsphinx.readthedocs.io/en/0.7.0/custom-css.html
[250] https://pypi.org/project/nbsphinx/0.6.1/
[251] https://github.com/spatialaudio/nbsphinx/compare/0.6.0...0.6.1
[252] https://pypi.org/project/nbsphinx/0.6.0/
[253] https://github.com/spatialaudio/nbsphinx/compare/0.5.1...0.6.0

- Python 2 support has been dropped

Version 0.5.1 – 2020-01-28 – PyPI[Page 77, 254] – diff[255]

- This will be the last release supporting Python 2.x!
- Support for https://github.com/choldgraf/sphinx-copybutton
- Executed notebooks are now saved in the HTML output directory

Version 0.5.0 – 2019-11-20 – PyPI[256] – diff[257]

- Automatic support for Jupyter widgets, customizable with `nbsphinx_widgets_path` (and `nbsphinx_widgets_options`)

Version 0.4.3 – 2019-09-30 – PyPI[258] – diff[259]

- Add option `nbsphinx_requirejs_path` (and `nbsphinx_requirejs_options`)

Version 0.4.2 – 2019-01-15 – PyPI[260] – diff[261]

- Re-establish Python 2 compatibility (but the clock is ticking ...)

Version 0.4.1 – 2018-12-16 – PyPI[262] – diff[263]

- Fix issue #266

Version 0.4.0 – 2018-12-14 – PyPI[264] – diff[265]

- Support for "data-cite" HTML tags in Markdown cells
- Add option `nbsphinx_custom_formats`
- LaTeX macros `\nbsphinxstartnotebook` and `\nbsphinxstopnotebook`
- Support for cell attachments
- Add options `nbsphinx_input_prompt` and `nbsphinx_output_prompt`
- Re-design LaTeX output of code cells, fix image sizes

Version 0.3.5 – 2018-09-10 – PyPI[266] – diff[267]

- Disable `nbconvert` version 5.4 to avoid issue #878[268]

Version 0.3.4 – 2018-07-28 – PyPI[269] – diff[270]

- Fix issue #196 and other minor changes

Version 0.3.3 – 2018-04-25 – PyPI[271] – diff[272]

[254] https://pypi.org/project/nbsphinx/0.5.1/
[255] https://github.com/spatialaudio/nbsphinx/compare/0.5.0...0.5.1
[256] https://pypi.org/project/nbsphinx/0.5.0/
[257] https://github.com/spatialaudio/nbsphinx/compare/0.4.3...0.5.0
[258] https://pypi.org/project/nbsphinx/0.4.3/
[259] https://github.com/spatialaudio/nbsphinx/compare/0.4.2...0.4.3
[260] https://pypi.org/project/nbsphinx/0.4.2/
[261] https://github.com/spatialaudio/nbsphinx/compare/0.4.1...0.4.2
[262] https://pypi.org/project/nbsphinx/0.4.1/
[263] https://github.com/spatialaudio/nbsphinx/compare/0.4.0...0.4.1
[264] https://pypi.org/project/nbsphinx/0.4.0/
[265] https://github.com/spatialaudio/nbsphinx/compare/0.3.5...0.4.0
[266] https://pypi.org/project/nbsphinx/0.3.5/
[267] https://github.com/spatialaudio/nbsphinx/compare/0.3.4...0.3.5
[268] https://github.com/jupyter/nbconvert/issues/878
[269] https://pypi.org/project/nbsphinx/0.3.4/
[270] https://github.com/spatialaudio/nbsphinx/compare/0.3.3...0.3.4
[271] https://pypi.org/project/nbsphinx/0.3.3/
[272] https://github.com/spatialaudio/nbsphinx/compare/0.3.2...0.3.3

- Locally linked files are only copied for Jupyter notebooks (and not anymore for other Sphinx source files)

Version 0.3.2 – 2018-03-28 – PyPI[Page 78, 273] **– diff**[274]

- Links to local files are rewritten for all Sphinx source files (not only Jupyter notebooks)

Version 0.3.1 – 2018-01-17 – PyPI[275] **– diff**[276]

- Enable notebook translations (NB: The use of reST strings is temporary!)

Version 0.3.0 – 2018-01-02 – PyPI[277] **– diff**[278]

- Add options `nbsphinx_prolog` and `nbsphinx_epilog`
- Links from `*.rst` files to notebooks have to start with a slash

Version 0.2.18 – 2017-12-03 – PyPI[279] **– diff**[280]

- Fix issue #148

Version 0.2.17 – 2017-11-12 – PyPI[281] **– diff**[282]

- Fix issue #146

Version 0.2.16 – 2017-11-07 – PyPI[283] **– diff**[284]

- Fix issue #142

Version 0.2.15 – 2017-11-03 – PyPI[285] **– diff**[286]

- Links to subsections are now possible in all source files

Version 0.2.14 – 2017-06-09 – PyPI[287] **– diff**[288]

- Add option `nbsphinx_kernel_name`

Version 0.2.13 – 2017-01-25 – PyPI[289] **– diff**[290]

- Minor fixes

Version 0.2.12 – 2016-12-19 – PyPI[291] **– diff**[292]

- Basic support for widgets
- CSS is now "responsive", some new CSS classes

Version 0.2.11 – 2016-11-19 – PyPI[293] **– diff**[294]

[273] https://pypi.org/project/nbsphinx/0.3.2/
[274] https://github.com/spatialaudio/nbsphinx/compare/0.3.1...0.3.2
[275] https://pypi.org/project/nbsphinx/0.3.1/
[276] https://github.com/spatialaudio/nbsphinx/compare/0.3.0...0.3.1
[277] https://pypi.org/project/nbsphinx/0.3.0/
[278] https://github.com/spatialaudio/nbsphinx/compare/0.2.18...0.3.0
[279] https://pypi.org/project/nbsphinx/0.2.18/
[280] https://github.com/spatialaudio/nbsphinx/compare/0.2.17...0.2.18
[281] https://pypi.org/project/nbsphinx/0.2.17/
[282] https://github.com/spatialaudio/nbsphinx/compare/0.2.16...0.2.17
[283] https://pypi.org/project/nbsphinx/0.2.16/
[284] https://github.com/spatialaudio/nbsphinx/compare/0.2.15...0.2.16
[285] https://pypi.org/project/nbsphinx/0.2.15/
[286] https://github.com/spatialaudio/nbsphinx/compare/0.2.14...0.2.15
[287] https://pypi.org/project/nbsphinx/0.2.14/
[288] https://github.com/spatialaudio/nbsphinx/compare/0.2.13...0.2.14
[289] https://pypi.org/project/nbsphinx/0.2.13/
[290] https://github.com/spatialaudio/nbsphinx/compare/0.2.12...0.2.13
[291] https://pypi.org/project/nbsphinx/0.2.12/
[292] https://github.com/spatialaudio/nbsphinx/compare/0.2.11...0.2.12
[293] https://pypi.org/project/nbsphinx/0.2.11/
[294] https://github.com/spatialaudio/nbsphinx/compare/0.2.10...0.2.11

- Minor fixes

Version 0.2.10 – 2016-10-16 – PyPI[Page 79, 295] **– diff**[296]

- Enable JavaScript output cells

Version 0.2.9 – 2016-07-26 – PyPI[297] **– diff**[298]

- Add option `nbsphinx_prompt_width`

Version 0.2.8 – 2016-05-20 – PyPI[299] **– diff**[300]

- Add options `nbsphinx_execute` and `nbsphinx_execute_arguments`
- Separate "display priority" for HTML and LaTeX

Version 0.2.7 – 2016-05-04 – PyPI[301] **– diff**[302]

- Special CSS tuning for `sphinx_rtd_theme`
- Replace info/warning `<div>` elements with `nbinfo/nbwarning`

Version 0.2.6 – 2016-04-12 – PyPI[303] **– diff**[304]

- Support for LaTeX math environments in Markdown cells
- Add options `nbsphinx_timeout` and `nbsphinx_codecell_lexer`

Version 0.2.5 – 2016-03-15 – PyPI[305] **– diff**[306]

- Add option `nbsphinx_allow_errors` to globally ignore exceptions
- Separate class *nbsphinx.Exporter*

Version 0.2.4 – 2016-02-12 – PyPI[307] **– diff**[308]

- Support for "nbsphinx-toctree" cell metadata

Version 0.2.3 – 2016-01-22 – PyPI[309] **– diff**[310]

- Links from notebooks to local files can now be used

Version 0.2.2 – 2016-01-06 – PyPI[311] **– diff**[312]

- Support for links to sub-sections in other notebooks

Version 0.2.1 – 2016-01-04 – PyPI[313] **– diff**[314]

- No need to mention `source_suffix` and `source_parsers` in `conf.py`

[295] https://pypi.org/project/nbsphinx/0.2.10/
[296] https://github.com/spatialaudio/nbsphinx/compare/0.2.9...0.2.10
[297] https://pypi.org/project/nbsphinx/0.2.9/
[298] https://github.com/spatialaudio/nbsphinx/compare/0.2.8...0.2.9
[299] https://pypi.org/project/nbsphinx/0.2.8/
[300] https://github.com/spatialaudio/nbsphinx/compare/0.2.7...0.2.8
[301] https://pypi.org/project/nbsphinx/0.2.7/
[302] https://github.com/spatialaudio/nbsphinx/compare/0.2.6...0.2.7
[303] https://pypi.org/project/nbsphinx/0.2.6/
[304] https://github.com/spatialaudio/nbsphinx/compare/0.2.5...0.2.6
[305] https://pypi.org/project/nbsphinx/0.2.5/
[306] https://github.com/spatialaudio/nbsphinx/compare/0.2.4...0.2.5
[307] https://pypi.org/project/nbsphinx/0.2.4/
[308] https://github.com/spatialaudio/nbsphinx/compare/0.2.3...0.2.4
[309] https://pypi.org/project/nbsphinx/0.2.3/
[310] https://github.com/spatialaudio/nbsphinx/compare/0.2.2...0.2.3
[311] https://pypi.org/project/nbsphinx/0.2.2/
[312] https://github.com/spatialaudio/nbsphinx/compare/0.2.1...0.2.2
[313] https://pypi.org/project/nbsphinx/0.2.1/
[314] https://github.com/spatialaudio/nbsphinx/compare/0.2.0...0.2.1

Version 0.2.0 – 2015-12-27 – PyPI[Page 80, 315] **– diff**[316]

- Add support for `allow_errors` and `hidden` metadata
- Add custom reST template
- Add nbinput and nboutput directives with HTML+CSS and LaTeX formatting
- Turn nbsphinx into a Sphinx extension

Version 0.1.0 – 2015-11-29

Initial release

[315] `https://pypi.org/project/nbsphinx/0.2.0/`

[316] `https://github.com/spatialaudio/nbsphinx/compare/0.1.0...0.2.0`