# Python

Read the Docs team

Feb 13, 2024

# CONTENTS

This example shows a Jupyter Book project built and published on Read the Docs. You're encouraged to use it to get inspiration and copy & paste from the files in the source code repository. In the source repository, you will also find the relevant configuration and instructions for building Jupyter Book projects on Read the Docs.

If you are using Read the Docs for the first time, have a look at the official Read the Docs Tutorial. If you are using Jupyter Book for the first time, have a look at the official Jupyter Book documentation.

# Why run Jupyter Book with Read the Docs?

Read the Docs simplifies developing Jupyter Book projects by automating building, versioning, and hosting of your project for you. You might be familiar with Read the Docs for software documentation projects, but these features are just as relevant for science.

With Read the Docs, you can improve collaboration on your Jupyter Book project with Git (GitHub, GitLab, BitBucket etc.) and then connect the Git repository to Read the Docs. Once Read the Docs and the git repository are connected, your project will be built and published automatically every time you commit and push changes with git. Furthermore, if you open Pull Requests, you can preview the result as rendered by Jupyter Book.

## What is in this example?

Jupyter Book has a number of built-in features. This is a small example book to give you a feel for how book content is structured. It shows off a few of the major file types, as well as some sample content. It does not go in-depth into any particular topic - check out the Jupyter Book documentation for more information.

- *Examples of Markdown*
- *Rendering a notebook Jupyter Notebook*
- *A notebook written in MyST Markdown*

We have also added some popular features for Jupyter Book that really you shouldn't miss when building your own project with Jupyter Book and Read the Docs:

- *intersphinx to link to other documentation and Jupyter Book projects*
- *sphinx-examples to show examples and results side-by-side*
- *sphinx-hoverxref to preview cross-references*
- *sphinx-proof for logic and math, to write proofs, theorems, lemmas etc.*

## Table of Contents

Here is an automatically generated Tabel of Contents:

- *sphinx-examples: Code examples simplified*
- *sphinx-hoverxref: Preview tooltips for cross-references*
- *sphinx-proof: advanced proofs, theorems & more…*

# Part I

# Basic examples 

CHAPTER

ONE

# MARKDOWN FILES

Whether you write your book's content in Jupyter Notebooks (`.ipynb`) or in regular markdown files (`.md`), you'll write in the same flavor of markdown called **MyST Markdown**. This is a simple file to help you get started and show off some syntax.

## 1.1 What is MyST?

MyST stands for "Markedly Structured Text". It is a slight variation on a flavor of markdown called "CommonMark" markdown, with small syntax extensions to allow you to write **roles** and **directives** in the Sphinx ecosystem.

For more about MyST, see the MyST Markdown Overview.

## 1.2 Sample Roles and Directives

Roles and directives are two of the most powerful tools in Jupyter Book. They are kind of like functions, but written in a markup language. They both serve a similar purpose, but **roles are written in one line**, whereas **directives span many lines**. They both accept different kinds of inputs, and what they do with those inputs depends on the specific role or directive that is being called.

Here is a "note" directive:

---

**Note:** Here is a note

---

It will be rendered in a special box when you build your book.

Here is an inline directive to refer to a document: *Notebooks with MyST Markdown*.

## 1.3 Citations

You can also cite references that are stored in a `bibtex` file. For example, the following syntax: `{cite}`holdgraf_evidence_2014`` will render like this: [HdHPK14].

Moreover, you can insert a bibliography into your page with this syntax: The `{bibliography}` directive must be used for all the `{cite}` roles to render properly. For example, if the references for your book are stored in `references.bib`, then the bibliography is inserted with:

## 1.4 Learn more

This is just a simple starter to get you started. You can learn a lot more at jupyterbook.org.

CHAPTER

TWO

# CONTENT WITH NOTEBOOKS

You can also create content with Jupyter Notebooks. This means that you can include code blocks and their outputs in your book.

## 2.1 Markdown + notebooks

As it is markdown, you can embed images, HTML, etc into your posts!


You can also $add_{math}$ and

$$math^{blocks}$$

or

$$mean la_{tex}$$

$$mathblocks$$

But make sure you \$Escape \$your \$dollar signs \$you want to keep!

## 2.2 MyST markdown

MyST markdown works in Jupyter Notebooks as well. For more information about MyST markdown, check out the MyST guide in Jupyter Book, or see the MyST markdown documentation.

## 2.3 Code blocks and outputs

Jupyter Book will also embed your code blocks and output in your book. For example, here's some sample Matplotlib code:

```
from matplotlib import rcParams, cycler
import matplotlib.pyplot as plt
import numpy as np
plt.ion()
```

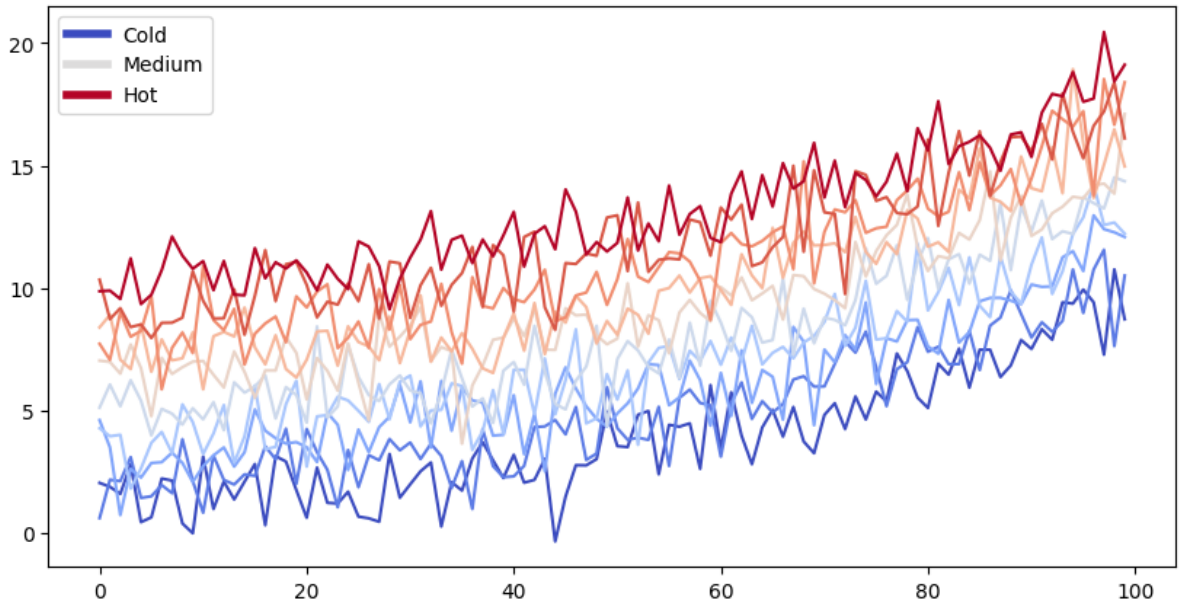
```
<contextlib.ExitStack at 0x7f07be290370>
```

```
# Fixing random state for reproducibility
np.random.seed(19680801)

N = 10
data = [np.logspace(0, 1, 100) + np.random.randn(100) + ii for ii in range(N)]
data = np.array(data).T
cmap = plt.cm.coolwarm
rcParams['axes.prop_cycle'] = cycler(color=cmap(np.linspace(0, 1, N)))


from matplotlib.lines import Line2D
custom_lines = [Line2D([0], [0], color=cmap(0.), lw=4),
                Line2D([0], [0], color=cmap(.5), lw=4),
                Line2D([0], [0], color=cmap(1.), lw=4)]

fig, ax = plt.subplots(figsize=(10, 5))
lines = ax.plot(data)
ax.legend(custom_lines, ['Cold', 'Medium', 'Hot']);
```




There is a lot more that you can do with outputs (such as including interactive outputs) with your book. For more information about this, see the Jupyter Book documentation

CHAPTER

THREE

# NOTEBOOKS WITH MYST MARKDOWN

Jupyter Book also lets you write text-based notebooks using MyST Markdown. See the Notebooks with MyST Markdown documentation for more detailed instructions. This page shows off a notebook written in MyST Markdown.

## 3.1 An example cell

With MyST Markdown, you can define code cells with a directive like so:

```
print(2 + 2)
```

```
4
```

When your book is built, the contents of any `{code-cell}` blocks will be executed with your default Jupyter kernel, and their outputs will be displayed in-line with the rest of your content.

**See also:**

Jupyter Book uses Jupytext to convert text-based files to notebooks, and can support many other text-based notebook files.

## 3.2 Create a notebook with MyST Markdown

MyST Markdown notebooks are defined by two things:

1. YAML metadata that is needed to understand if / how it should convert text files to notebooks (including information about the kernel needed). See the YAML at the top of this page for example.
2. The presence of `{code-cell}` directives, which will be executed with your book.

That's all that is needed to get started!

## 3.3 Quickly add YAML metadata for MyST Notebooks

If you have a markdown file and you'd like to quickly add YAML metadata to it, so that Jupyter Book will treat it as a MyST Markdown Notebook, run the following command:

```
jupyter-book myst init path/to/markdownfile.md
```

# Part II

# Cool extensions 

CHAPTER

FOUR

# INTERSPHINX: CROSS-REFERENCE OTHER PROJECTS

Behind the built-in Sphinx extension `intersphinx` is a powerful tool to reference sections in other Sphinx and Jupyter Book documentation projects.

You can configure mappings to external Sphinx projects in your Jupyter Book configuration, the `_config.yml` file. In this example project, we have configured `ebp` to reference `https://executablebooks.org/en/latest/`. In the following code examples, we refer to the configured `ebp` mapping and link directly to a section called `tools`.

```
We can link to pages in other documentation projects.
This is a link to the
[Executable Book project's list of tools they build](ebp:tools)
```

We can link to pages in other documentation projects. This is a link to the *Executable Book project's list of tools they build*

```
```{eval-rst}
We can link to pages in other documentation projects.
This is a link to the
:doc:`Executable Book project's list of tools they build <ebp:tools>`
```

We can link to pages in other documentation projects. This is a link to the Executable Book project's list of tools they build

---

**Note:** In the above `reStructuredText` example, we use `{eval-rst}` to write reST inside a `.md` file (i.e. the one you are reading now). You only need to use this directive if you are writing reST code in a `.md` file.

---
```

CHAPTER

FIVE

# SPHINX-EXAMPLES: CODE EXAMPLES SIMPLIFIED

In this example project, we have enabled the extension `sphinx-examples` in `_config.yml` to be able to display source code alongside its rendered result.

You can use it this way:

Use examples to show source code and rendered result

````
```{example} Example title
Here's my **example**!
```
````

Example title

```
Here's my **example**!
```

Here's my **example**!

CHAPTER

SIX

# SPHINX-HOVERXREF: PREVIEW TOOLTIPS FOR CROSS-REFERENCES

Using `sphinx-hoverxref`, we can preview cross-references in the documentation.

For instance, try placing your mouse over *this reference to the front page*. We have use a named "target" for the reference. Here are some examples of how to name your targets:

```
(foobar)=

# My Headline

Here is a paragraph, we link to this headline {hoverxref}`like this <foobar>`.
```

```
.. _foobar:

My Headline
===========

Here is a paragraph, we link to this headline :hoverxref:`like this <foobar>`
```

`sphinx-hoverxref` works especially well on Read the Docs, since the platform runs the necessary API backend for generating preview data. Aside from enabling the extention in your project's `_config.yml`, you don't have to do anyhing, it Just Works™.

CHAPTER

SEVEN

# SPHINX-PROOF: ADVANCED PROOFS, THEOREMS & MORE...

Easily render proofs, theorems, axioms, lemmas, definitions and much more with `sphinx-proof`. Read more in the *documentation of sphinx-proof*.

---

Proof. We'll omit the full proof.

But we will prove sufficiency of the asserted conditions.

To this end, let $y \in \mathbb{R}^n$ and let $S$ be a linear subspace of $\mathbb{R}^n$.

Let $\hat{y}$ be a vector in $\mathbb{R}^n$ such that $\hat{y} \in S$ and $y - \hat{y} \perp S$.

Let $z$ be any other point in $S$ and use the fact that $S$ is a linear subspace to deduce

$$\|y - z\|^2 = \|(y - \hat{y}) + (\hat{y} - z)\|^2 = \|y - \hat{y}\|^2 + \|\hat{y} - z\|^2$$

Hence $\|y - z\| \geq \|y - \hat{y}\|$, which completes the proof.

---

# BIBLIOGRAPHY

[HdHPK14] Christopher Ramsay Holdgraf, Wendy de Heer, Brian N. Pasley, and Robert T. Knight. Evidence for Predictive Coding in Human Auditory Cortex. In *International Conference on Cognitive Neuroscience*. Brisbane, Australia, Australia, 2014. Frontiers in Neuroscience.