Welcome to your Jupyter Book

This is a small sample book to give you a feel for how book content is structured.

> **Note**
>
> Here is a note!

And here is a code block:

```
e = mc^2
```

Check out the content pages bundled with this sample book to see more.

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**.

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.

What are 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.

Using a directive

At its simplest, you can insert a directive into your book's content like so:

```
```{mydirectivename}
My directive content
```
```

This will only work if a directive with name `mydirectivename` already exists (which it doesn't). There are many pre-defined directives associated with Jupyter Book. For example, to insert a note box into your content, you can use the following directive:

```
```{note}
Here is a note
```
```

This results in:

> **Note**
>
> Here is a note

In your built book.

For more information on writing directives, see the MyST documentation.

Using a role

Roles are very similar to directives, but they are less-complex and written entirely on one line. You can insert a role into your book's content with this pattern:

```
Some content {rolename}`and here is my role's content!`
```

Again, roles will only work if `rolename` is a valid role's name. For example, the `doc` role can be used to refer to another page in your book. You can refer directly to another page by its relative path. For example, the role syntax `{doc}`intro`` will result in: Welcome to your Jupyter Book.

For more information on writing roles, see the MyST documentation.

Adding a citation

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:

````
```{bibliography}
```
````

Resulting in a rendered bibliography that looks like:

[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.

Executing code in your markdown files

If you'd like to include computational content inside these markdown files, you can use MyST Markdown to define cells that will be executed when your book is built. Jupyter Book uses *jupytext* to do this.

First, add Jupytext metadata to the file. For example, to add Jupytext metadata to this markdown page, run this command:

```
jupyter-book myst init markdown.md
```

Once a markdown file has Jupytext metadata in it, you can add the following directive to run the code at build time:

````
```{code-cell}
print("Here is some code to execute")
```
````

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.

For more information about executing computational content with Jupyter Book, see The MyST-NB documentation.

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.

Markdown + notebooks

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

Markedly Structured Text

You can also \(add_{math}\) and
\[math^{blocks} \]
or
\[\begin{split} \begin{aligned} \mbox{mean} la_{tex} \\ \\ math blocks \end{aligned} \end{split}\]
But make sure you $Escape $your $dollar signs $you want to keep!

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.

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:

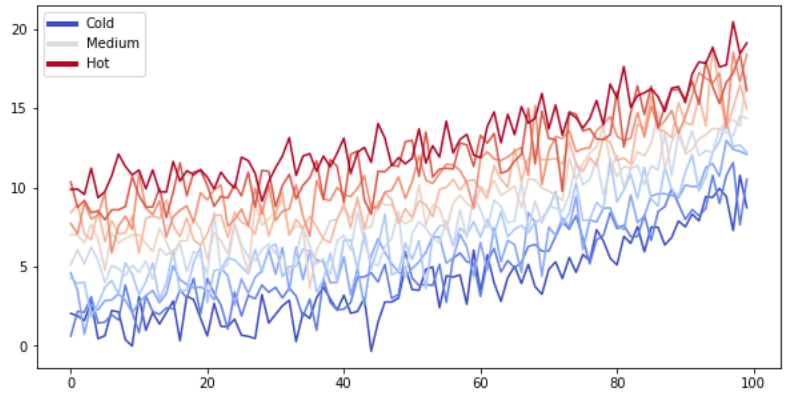
```
from matplotlib import rcParams, cycler
import matplotlib.pyplot as plt
import numpy as np
plt.ion()
```

```
# 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

---