Welcome to your Jupyter Book

Contents

This is a small sample 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.

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**. This is a simple file to help you get started and show off some syntax.

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.

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.

Citations test by Byamba

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:

[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.

Learn more

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

Content with notebooks

Skip to main content

Markdown + notebooks

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

You can also add_{math} and

$$math^{blocks}$$

or

$$\text{mean} la_{tex}$$

$$mathblocks$$

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.

Skip to main content

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()
```

```
---------------------------------------------------------------------------
ModuleNotFoundError                       Traceback (most recent call last)
Cell In[1], line 1
----> 1 from matplotlib import rcParams, cycler
      2 import matplotlib.pyplot as plt
      3 import numpy as np

ModuleNotFoundError: No module named 'matplotlib'
```

```
# 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

Notebooks with MyST Markdown

Skip to main content

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.

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.

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!

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

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

Integers

```
# Declare an integer variable
age = 25

# Perform arithmetic operations
result = age + 5
print(result)  # Output: 30
```

```
30
```

```
age = 40
result = age + 100
```

```
print(result)
```

```
140
```

Floats

```
# Declare a float variable
price = 19.99

# Perform arithmetic operations
discounted_price = price * 0.8
print(discounted_price)  # Output: 15.992
```

```
15.991999999999999
```

```
# Perform arithmetic operations
discounted_price = price * 0.1
print(discounted_price)  # Output: 15.992
```

```
1.9989999999999999
```

Strings

```
# Declare a string variable
greeting = "Hello, World!"

# Concatenate strings
welcome_message = greeting + " Welcome to Python!"
print(welcome_message)  # Output: Hello, World! Welcome to Python!
```

```
Hello, World! Welcome to Python!
```

```
"Byamba" == "byamba "
```

```
False
```

```
"Byamba" == "Byamba"
```

```
True
```

Lists

```
# Declare a list variable
fruits = ["apple", "banana", "cherry"]

# Access elements of a list
print(fruits[0])  # Output: apple

# Modify elements of a list
fruits[1] = "blueberry"
print(fruits)  # Output: ['apple', 'blueberry', 'cherry']
```

```
apple
['apple', 'blueberry', 'cherry']
```

```
print(fruits[0])
```

```
apple
```

```
numbers=[1,2,5, 0.8, "dog"]
```

```
print(numbers[4])
```

```
dog
```

```
numbers[0]
```

```
1
```

Tuples

```
# Declare a tuple variable
coordinates = (12.5, 45.3)

# Access elements of a tuple
print(coordinates[1])  # Output: 45.3

# Tuples are immutable; the following line will raise an error
# coordinates[1] = 50
```

```
45.3
```

Dictionaries

```
# Declare a dictionary variable
person = {"name": "Alice", "age": 30, "comment":"it is very nice"}

# Access elements of a dictionary
print(person["name"])  # Output: Alice

# Modify elements of a dictionary
person["age"] = 31
print(person)  # Output: {'name': 'Alice', 'age': 31}
```

```
Alice
{'name': 'Alice', 'age': 31, 'comment': 'it is very nice'}
```

```
person['comment']
```

```
'it is very nice'
```

```
person.keys()
```

```
dict_keys(['name', 'age', 'comment'])
```

Skip to main content

```
dict_values(['Alice', 31, 'it is very nice'])
```

Booleans

```
# Declare a boolean variable
is_active = True

# Use boolean in a conditional statement
if is_active:
    print("User is active.")  # Output: User is active.
```

```
User is active.
```

```
6 > 5
```

```
True
```

```
4 < 1
```

```
False
```

```
is_active = False


if is_active:
    print("yes")
else:
    print("no")
```

```
no
```

```
person['name']=='Byamba'
```

Skip to main content

```
False
```

```
person['name']=='Alice'
```

```
True
```

```
person['age']==41
```

```
False
```