2020

jupytercon

module 5:

more Jupyter Book

by Martina Vilas
@martinagvilas

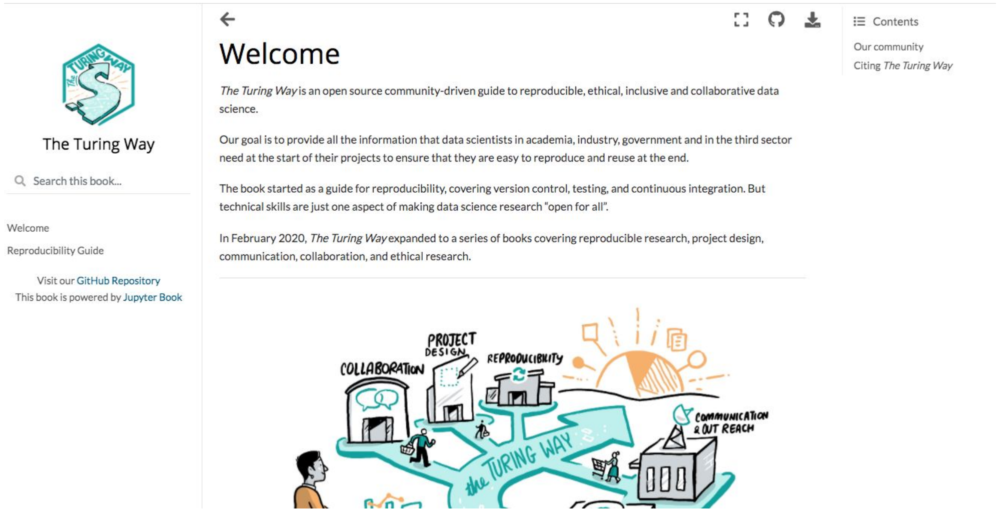
#JupyterCon2020

so far ...

- you have built a Jupyter Book (*module 3*) and modify its appearance using a configuration file (*module 4*)

so far ...

- you have built a Jupyter Book (*module 3*) and modify its appearance using a configuration file (*module 4*)



The Turing Way

Search this book...

Welcome

Reproducibility Guide

Visit our GitHub Repository
This book is powered by Jupyter Book

Welcome

The Turing Way is an open source community-driven guide to reproducible, ethical, inclusive and collaborative data science.

Our goal is to provide all the information that data scientists in academia, industry, government and in the third sector need at the start of their projects to ensure that they are easy to reproduce and reuse at the end.

The book started as a guide for reproducibility, covering version control, testing, and continuous integration. But technical skills are just one aspect of making data science research "open for all".

In February 2020, *The Turing Way* expanded to a series of books covering reproducible research, project design, communication, collaboration, and ethical research.

Contents

Our community

Citing *The Turing Way*

jupytercon

learning objectives of *module 5*

- show how we can add Jupyter Notebooks to our Jupyter Book

learning objectives of *module 5*

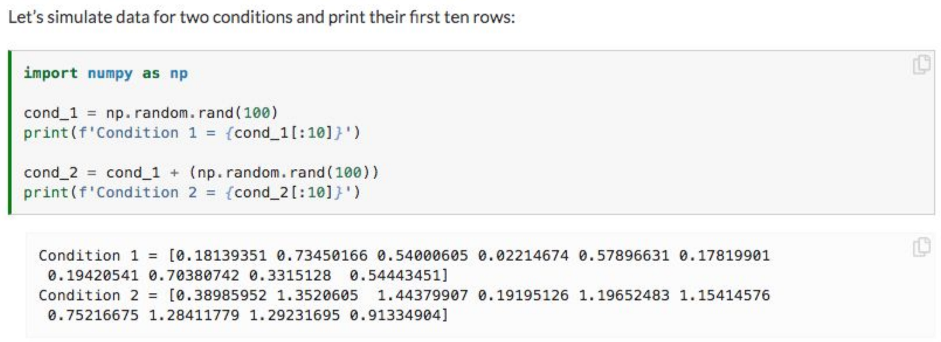
- show how we can add Jupyter Notebooks to our Jupyter Book
- explain how to make our Jupyter Notebooks interactive using Binder

learning objectives of *module 5*

- show how we can add Jupyter Notebooks to our Jupyter Book
- explain how to make our Jupyter Notebooks executable using Binder
- introduce how Jupyter Book supports a special flavor of markdown called MyST markdown

support of Jupyter Notebooks

support of Jupyter Notebooks


Let's simulate data for two conditions and print their first ten rows:

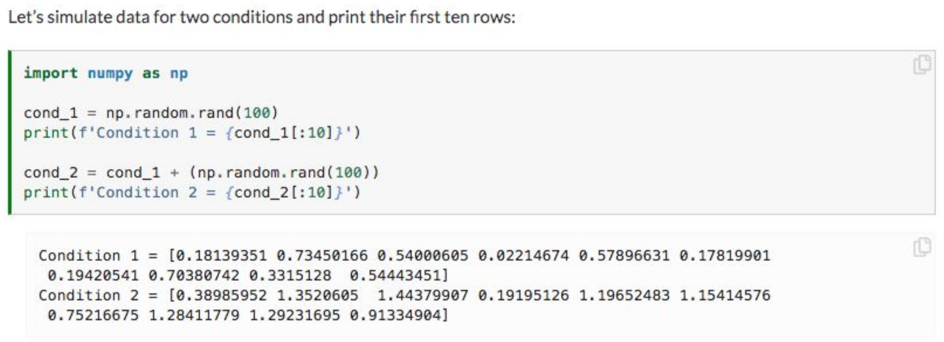
```
import numpy as np

cond_1 = np.random.rand(100)
print(f'Condition 1 = {cond_1[:10]}')

cond_2 = cond_1 + (np.random.rand(100))
print(f'Condition 2 = {cond_2[:10]}')
```

```
Condition 1 = [0.18139351 0.73450166 0.54000605 0.02214674 0.57896631 0.17819901
 0.19420541 0.70380742 0.3315128  0.54443451]
Condition 2 = [0.38985952 1.3520605  1.44379907 0.19195126 1.19652483 1.15414576
 0.75216675 1.28411779 1.29231695 0.91334904]
```

support of Jupyter Notebooks



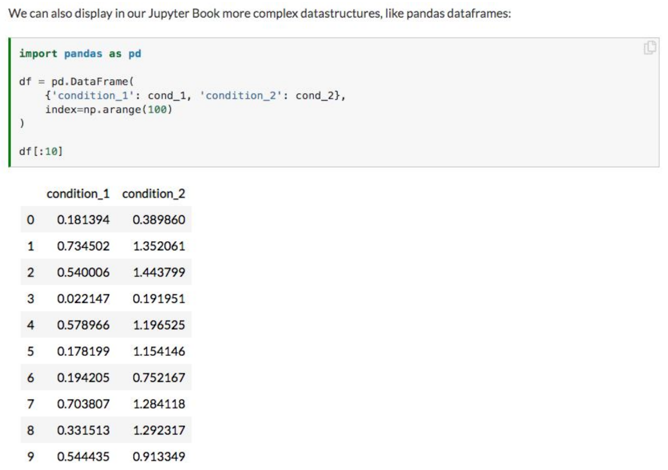
Let's simulate data for two conditions and print their first ten rows:

```
import numpy as np

cond_1 = np.random.rand(100)
print(f'Condition 1 = {cond_1[:10]}')

cond_2 = cond_1 + (np.random.rand(100))
print(f'Condition 2 = {cond_2[:10]}')
```

```
Condition 1 = [0.18139351 0.73450166 0.54000605 0.02214674 0.57896631 0.17819901
 0.19420541 0.70380742 0.3315128  0.54443451]
Condition 2 = [0.38985952 1.3520605  1.44379907 0.19195126 1.19652483 1.15414576
 0.75216675 1.28411779 1.29231695 0.91334904]
```



We can also display in our Jupyter Book more complex datastructures, like pandas dataframes:

```
import pandas as pd

df = pd.DataFrame(
    {'condition_1': cond_1, 'condition_2': cond_2},
    index=np.arange(100)
)

df[:10]
```

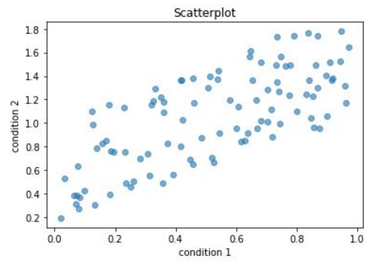
	condition_1	condition_2
0	0.181394	0.389860
1	0.734502	1.352061
2	0.540006	1.443799
3	0.022147	0.191951
4	0.578966	1.196525
5	0.178199	1.154146
6	0.194205	0.752167
7	0.703807	1.284118
8	0.331513	1.292317
9	0.544435	0.913349

support of Jupyter Notebooks

And of course, we can display plots as well!

```python
import matplotlib.pyplot as plt

plt.scatter(cond_1, cond_2, alpha=.6)
plt.xlabel('condition 1')
plt.ylabel('condition 2')
plt.title('Scatterplot')
plt.show()
```



Let's simulate data for two conditions and print their first ten rows:

```python
import numpy as np

cond_1 = np.random.rand(100)
print(f'Condition 1 = {cond_1[:10]}')

cond_2 = cond_1 + (np.random.rand(100))
print(f'Condition 2 = {cond_2[:10]}')
```

```
Condition 1 = [0.18139351 0.73450166 0.54000605 0.02214674 0.57896631 0.17819901
 0.19420541 0.70380742 0.3315128  0.54443451]
Condition 2 = [0.38985952 1.3520605  1.44379907 0.19195126 1.19652483 1.15414576
 0.75216675 1.28411779 1.29231695 0.91334904]
```

We can also display in our Jupyter Book more complex datastructures, like pandas dataframes:

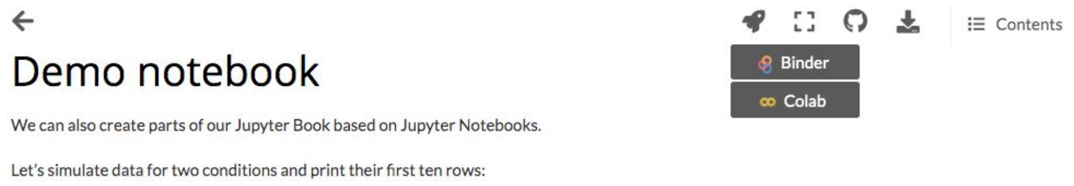
```python
import pandas as pd

df = pd.DataFrame(
    {'condition_1': cond_1, 'condition_2': cond_2},
    index=np.arange(100)
)

df[:10]
```

	condition_1	condition_2
0	0.181394	0.389860
1	0.734502	1.352061
2	0.540006	1.443799
3	0.022147	0.191951
4	0.578966	1.196525
5	0.178199	1.154146
6	0.194205	0.752167
7	0.703807	1.284118
8	0.331513	1.292317
9	0.544435	0.913349

support of Jupyter Notebooks and Binder

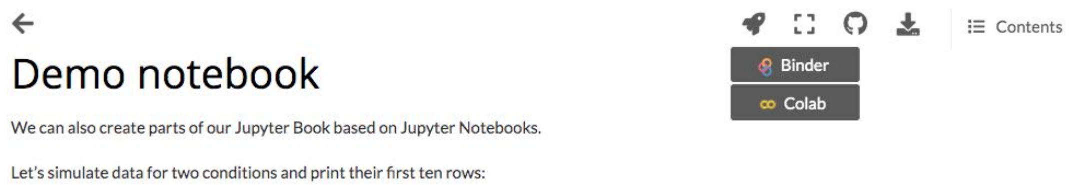


Demo notebook

We can also create parts of our Jupyter Book based on Jupyter Notebooks.

Let's simulate data for two conditions and print their first ten rows:

support of Jupyter Notebooks and Binder



Demo notebook

We can also create parts of our Jupyter Book based on Jupyter Notebooks.

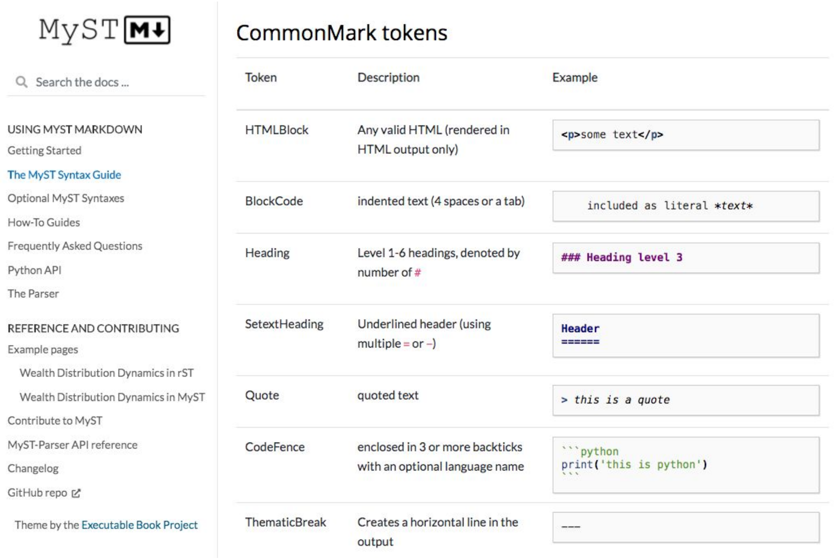
Let's simulate data for two conditions and print their first ten rows:

- You will need a GitHub repository that:
  - Hosts the Jupyter Notebook
  - Specifies a `requirements.txt`

support of MyST

- A language that supports:
    - CommonMark specification

https://myst-parser.readthedocs.io/en/latest/using/syntax.html


MyST
Search the docs ...
USING MYST MARKDOWN
Getting Started
The MyST Syntax Guide
Optional MyST Syntaxes
How-To Guides
Frequently Asked Questions
Python API
The Parser
REFERENCE AND CONTRIBUTING
Example pages
Wealth Distribution Dynamics in rST
Wealth Distribution Dynamics in MyST
Contribute to MyST
MyST-Parser API reference
Changelog
GitHub repo
Theme by the Executable Book Project

CommonMark tokens

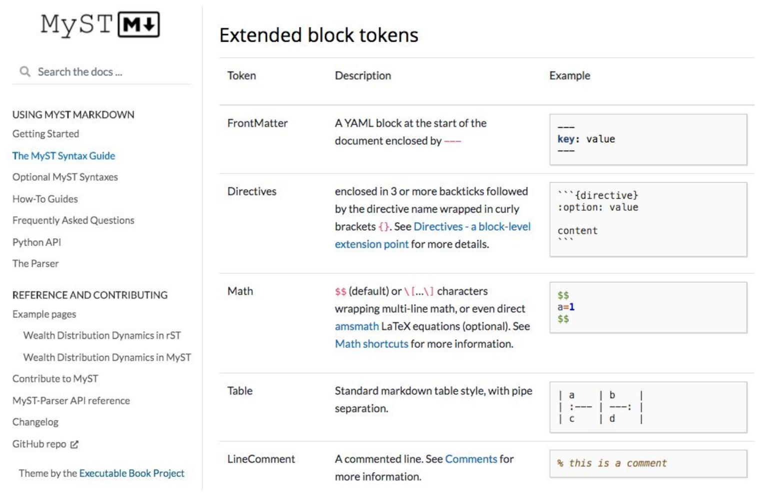
Token	Description	Example
HTMLBlock	Any valid HTML (rendered in HTML output only)	`<p>some text</p>`
BlockCode	indented text (4 spaces or a tab)	`    included as literal *text*`
Heading	Level 1-6 headings, denoted by number of #	`### Heading level 3`
SetextHeading	Underlined header (using multiple = or -)	`Header` `======`
Quote	quoted text	`> this is a quote`
CodeFence	enclosed in 3 or more backticks with an optional language name	` ```python` `print('this is python')` ` ``` `
ThematicBreak	Creates a horizontal line in the output	`---`

jupytercon

support of MyST

- A language that supports:
  - Features similar to those of `.rst` files that are used by Sphinx to convert your content to html
  - e.g. roles and directives

https://myst-parser.readthedocs.io/en/latest/using/syntax.html



MyST

Search the docs ...

USING MYST MARKDOWN
- Getting Started
- The MyST Syntax Guide
- Optional MyST Syntaxes
- How-To Guides
- Frequently Asked Questions
- Python API
- The Parser

REFERENCE AND CONTRIBUTING
- Example pages
  - Wealth Distribution Dynamics in rST
  - Wealth Distribution Dynamics in MyST
- Contribute to MyST
- MyST-Parser API reference
- Changelog
- GitHub repo

Theme by the Executable Book Project

Extended block tokens

Token	Description	Example

FrontMatter	A YAML block at the start of the document enclosed by `---`	`---`
		`key: value`
		`---`

Directives	enclosed in 3 or more backticks followed by the directive name wrapped in curly brackets `{}`. See Directives - a block-level extension point for more details.	`
		```
		{directive} `
		`:option: value`

		`content`
		`
		```
		 `

Math	`$$` (default) or `\[...\]` characters wrapping multi-line math, or even direct amsmath LaTeX equations (optional). See Math shortcuts for more information.	`$$`
		`a=1`
		`$$`

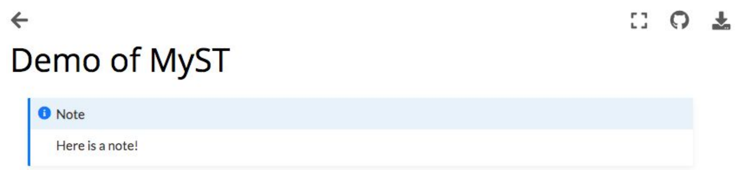
Table	Standard markdown table style, with pipe separation.	`| a    | b    |`
		`| :--- | ---: |`
		`| c    | d    |`

LineComment	A commented line. See Comments for more information.	`% this is a comment`

support of MyST

- A language that supports:
  - Features similar to those of .rst files that are used by Sphinx to convert your content to html
  - e.g. roles and directives

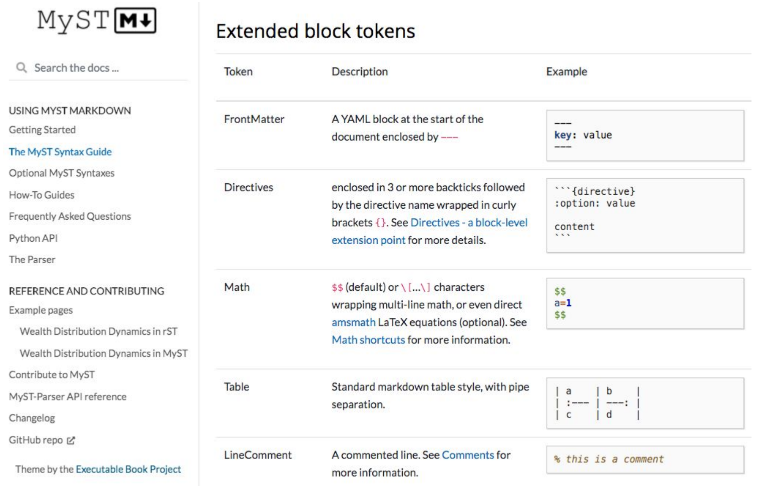
https://myst-parser.readthedocs.io/en/latest/using/syntax.html



Demo of MyST

Note

Here is a note!



MyST

Search the docs ...

USING MYST MARKDOWN
- Getting Started
- The MyST Syntax Guide
- Optional MyST Syntaxes
- How-To Guides
- Frequently Asked Questions
- Python API
- The Parser

REFERENCE AND CONTRIBUTING
- Example pages
  - Wealth Distribution Dynamics in rST
  - Wealth Distribution Dynamics in MyST
- Contribute to MyST
- MyST-Parser API reference
- Changelog
- GitHub repo

Theme by the Executable Book Project

Extended block tokens

Token	Description	Example

FrontMatter	A YAML block at the start of the document enclosed by `---`	`---`
		`key: value`
		`---`

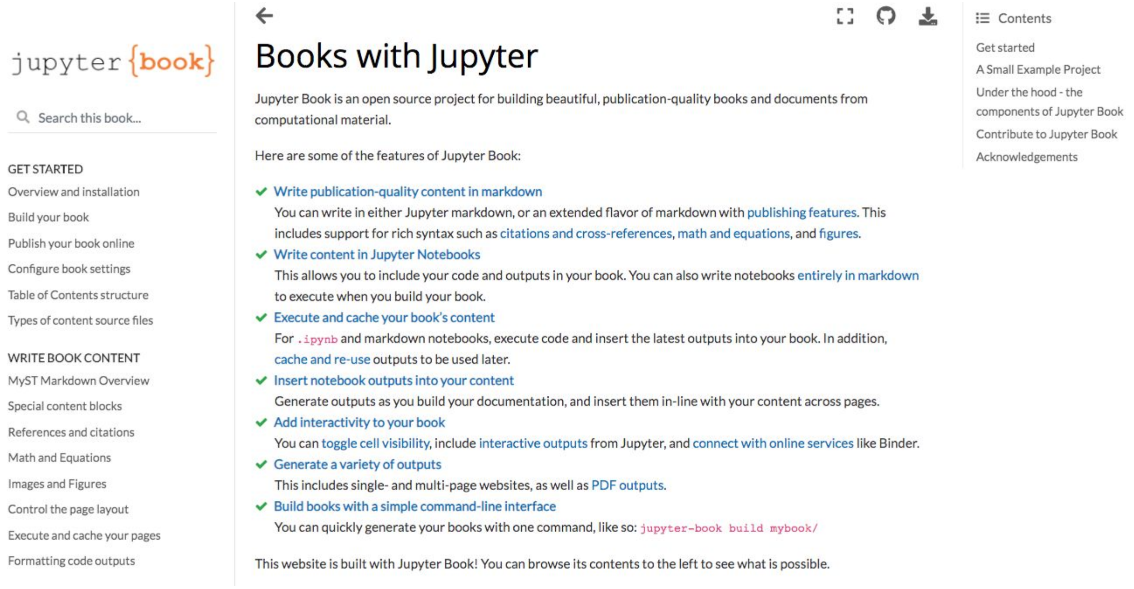
Directives	enclosed in 3 or more backticks followed by the directive name wrapped in curly brackets `{}`. See Directives - a block-level extension point for more details.	```` ```{directive} ````
		`:option: value`
		
		`content`
		```` ``` ````

Math	`$$` (default) or `\[...\]` characters wrapping multi-line math, or even direct amsmath LaTeX equations (optional). See Math shortcuts for more information.	`$$`
		`a=1`
		`$$`

Table	Standard markdown table style, with pipe separation.	`| a    | b    |`
		`| :--- | ---: |`
		`| c    | d    |`

LineComment	A commented line. See Comments for more information.	`% this is a comment`

check Jupyter Book's official documentation!



Books with Jupyter

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

Here are some of the features of Jupyter Book:

✓ Write publication-quality content in markdown
You can write in either Jupyter markdown, or an extended flavor of markdown with publishing features. This includes support for rich syntax such as citations and cross-references, math and equations, and figures.

✓ Write content in Jupyter Notebooks
This allows you to include your code and outputs in your book. You can also write notebooks entirely in markdown to execute when you build your book.

✓ Execute and cache your book's content
For `.ipynb` and markdown notebooks, execute code and insert the latest outputs into your book. In addition, cache and re-use outputs to be used later.

✓ Insert notebook outputs into your content
Generate outputs as you build your documentation, and insert them in-line with your content across pages.

✓ Add interactivity to your book
You can toggle cell visibility, include interactive outputs from Jupyter, and connect with online services like Binder.

✓ Generate a variety of outputs
This includes single- and multi-page websites, as well as PDF outputs.

✓ Build books with a simple command-line interface
You can quickly generate your books with one command, like so: `jupyter-book build mybook/`

This website is built with Jupyter Book! You can browse its contents to the left to see what is possible.

https://jupyterbook.org/

see you in *module 6*!