

HeatHack Test Book!!!!

Contents

Admonitions formatting

Test of panels

Test of sidebars

Markdown Files

Notebooks with MyST Markdown

Demonstration that we can produce the graphs we want easily in the Jupyter Book

This is a test of Github Pages rendering, before we commit to Jupyter Book. It's mostly the default Jupyter book with a few simple format tests added to the beginning.

This is public, but of no use to anyone but the developers yet (sorry).

See [the Jupyter Book documentation](#) for documentation.

Admonitions formatting

Note

Notes have a nice format with a relevant icon and a specific colour.

Warning

See, this one is a bit different because it's a warning. We want a bunch of possibilities like this that aren't types they have natively: Key Concept (which is a some science), Fun Fact!, and Further Readings.

Admonitions are the general case where you can say lots in the "header", but how do you set the icon and colour - where does the "class" go and what are the ones that are already defined?

Key Concepts

Here's some text about some science.

Further Readings

How do we make this one different?

Fun Fact!

And this one - can it be fun?

Test of panels

At some point we want to know, how hard is it to put a
vscript simulation in (for now, any simple one will
do)?

Test of sidebars

This is a place to look at the side bar rendering. Maybe
we can have classes like admonitions do. We might
want to have some of our admonition types be side
bars instead.

>Lorem ipsum dolor sit amet, consectetur adipiscing
elit, sed do eiusmod tempor incididunt ut labore et
dolore magna aliqua. Ut enim ad minim veniam, quis
nostrud exercitation ullamco laboris nisi ut aliquip ex
ea commodo consequat. Duis aute irure dolor in
reprehenderit in voluptate velit esse cillum dolore eu
fugiat nulla pariatur. Excepteur sint occaecat cupidatat
non proident, sunt in culpa qui officia deserunt mollit
anim id est laborum.

>Lorem ipsum dolor sit amet,
consectetur adipiscing elit, sed
do eiusmod tempor incididunt
ut labore et dolore magna
aliqua. Ut enim ad minim
veniam, quis nostrud
exercitation ullamco laboris nisi
ut aliquip ex ea commodo
consequat. Duis aute irure
dolor in reprehenderit in
voluptate velit esse cillum
dolore eu fugiat nulla pariatur.
Excepteur sint occaecat
cupidatat non proident, sunt in
culpa qui officia deserunt mollit
anim id est laborum.

>Lorem ipsum dolor sit amet,
consectetur adipiscing elit, sed
do eiusmod tempor incididunt
ut labore et dolore magna
aliqua. Ut enim ad minim
veniam, quis nostrud
exercitation ullamco laboris nisi
ut aliquip ex ea commodo
consequat. Duis aute irure
dolor in reprehenderit in
voluptate velit esse cillum
dolore eu fugiat nulla pariatur.
Excepteur sint occaecat
cupidatat non proident, sunt in
culpa qui officia deserunt mollit
anim id est laborum.

**Here's a
sidebar!**

>Lorem ipsum
dolor sit amet,
consectetur
adipiscing elit,
sed do eiusmod
tempor
incidunt ut
labore et dolore
magna aliqua. Ut
enim ad minim
veniam, quis
nostrud
exercitation
ullamco laboris
nisi ut aliquip ex
ea commodo
consequat. Duis
aute irure dolor
in reprehenderit
in voluptate velit
esse cillum
dolore eu fugiat
nulla pariatur.
Excepteur sint
occaecat
cupidatat non
proident, sunt in
culpa qui officia
deserunt mollit
anim id est
laborum.

lore magna aliqua. Ut enim ad minim veniam, quis
strud exercitation ullamco laboris nisi ut aliquip ex
a commodo consequat. Duis aute irure dolor in
prehenderit in voluptate velit esse cillum dolore eu
giat nulla pariatur. Excepteur sint occaecat cupidatat
on proident, sunt in culpa qui officia deserunt mollit
rim id est laborum.

rem ipsum dolor sit amet, consectetur adipiscing
it, sed do eiusmod tempor incididunt ut labore et
lore magna aliqua. Ut enim ad minim veniam, quis
strud exercitation ullamco laboris nisi ut aliquip ex
a commodo consequat. Duis aute irure dolor in
prehenderit in voluptate velit esse cillum dolore eu
giat nulla pariatur. Excepteur sint occaecat cupidatat
on proident, sunt in culpa qui officia deserunt mollit
rim id est laborum.

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 the 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

This 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

`{cite}`holdgraf_evidence_2014`` will render like is: [\[HdHPK14\]](#).

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:

[\[dHPK14\]](#) 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.

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.

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 [JupyterText](#) 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).

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 treat it as a MyST Markdown Notebook, run the following command:

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

Demonstration that we can reproduce the graphs we want easily in the Jupyter Book

This is a test file for some of the kinds of plots we will want in HeatHack-Sessions, so we can see what builds look like and how much we can automate. Although this uses csv files uploaded to Github manually, it may be possible to automate temp/RH feed download and the production of books specific to the venues we serve.

A basic plot showing a thingspeak temperature feed marked up for 16C, the child care commission minimum (not relevant for this particular data, just a test). This plot is also useful for assessing temperature control, especially on a short test for overshoot that comes holding a building at a temperature - cheapest in the summer. We'll want similar plots showing suggested RH bounds for the comfort of people and for organs/oil paintings and so on.

A more pan, zoom, etc - not beautiful, but even this basic level of plot would work. I wonder whether they'll be worried by the rogue readings. We could probably move based on improbably fast temperature ranges.

In our current thingspeak feeds, temperature is field1 and RH is field2 - we may be able to assign better names in future. I think I had to change the time format so that's either some scripting or a configuration change on the platform.

Plotly express is syntactic sugar over graph_objects; dropping down into the graph_objects themselves allows more possibilities.

Information about non-plotly approaches:

<https://jupyterbook.org/interactive/interactive.html> One consideration is whether they're going to need internet access to look at graphs - they might not have that when they're together if they meet on the premises. It might sound like it might be useful in that situation.

TODO: find out if we can hide the code.

```
import plotly.express as px
import plotly.graph_objects as go

import pandas as pd

filename = "thingspeak-feed"
dfthingspeak = pd.read_csv(filename +
".csv")
dfthingspeak["timestamp"] =
pd.to_datetime(dfthingspeak['created_at'])

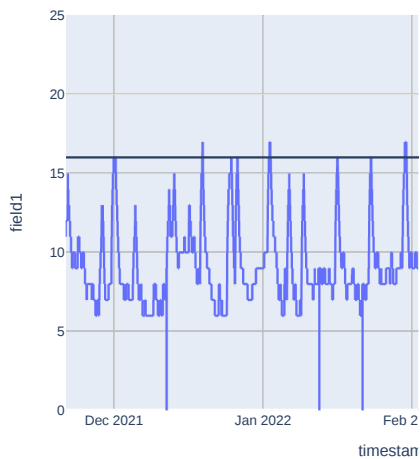
fig = px.line(dfthingspeak, x='timestamp',
y='field1', range_x=['2021-11-21', '2022-
04-07'],range_y=[0,25], title="Temperature
in a worship space: " + filename)

fig.add_hline(y=16)

fig.show()
```



Temperature in a worship space: thingspeak-fe



imple demonstration of data from two data frames on
e same plot - with the wrinkle that one frame is from
lascar logger. We will be roughly exploring the
alibration of the RH sensors by running batches of 10
HT22s alongside a few Lascars over an RH range
rd showing groups the results, so they can judge
ow much to trust the data.

ascars aren't configurable for what they export. I've
moved a Unicode character this couldn't deal with
egree symbol) and used Excel to change the data
rmat. These things should be fixable in code, but we
on't use Lascars enough for that to be a priority task.
ny processing we need to do on Thingspeak feeds is
priority, though.

ve need every line labelled. Doing that requires us to
ll out each line into a separate command to add it, I
ink - this might be an odd way of combining express
rd graph_object?

```
import plotly.express as px
import plotly.graph_objects as go

import pandas as pd
dfthingspeak = pd.read_csv("thingspeak-
feed.csv")
dfthingspeak["timestamp"] =
pd.to_datetime(dfthingspeak['created_at'])

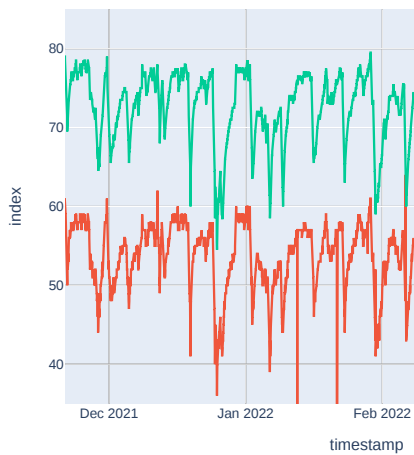
dflascar = pd.read_csv("lascar-data.csv")
dflascar['timestamp'] =
pd.to_datetime(dfascar['Time'])

# fig = px.line(dfthingspeak,
x='timestamp', y='field2', range_x=['2021-
11-21', '2022-04-07'],range_y=[35,85],
title="Comparing RH as measured by
different devices")
fig = px.line(dfthingspeak, x='timestamp',
range_x=['2021-11-21','2022-04-
07'],range_y=[35,85], title="Comparing RH
in a worship space as measured by
different devices side by side")
fig.add_scatter(x =
dfthingspeak['timestamp'], y =
dfthingspeak['field2'], name =
'thingspeak')
fig.add_scatter(x = dfascar['timestamp'],
y = dfascar['RH'], name = 'lascar')

fig.show()
```



Comparing RH in a worship space as measure



vertical lines are useful for the start and end time of events. It would be better rendered as a separate background shading when the space is occupied.

Perhaps we can set up a worksheet where they put in their usual weekly schedule with a descriptive short writing to render these. We could use diary export, but if their diary doesn't have a busy/free option, there's too much risk of personal data being in there, and there would be too many diary systems to deal with.

ODO: It would be helpful if there were a dropdown control for choosing to view a day or a week, and then click specific day or week. That sort of control could be used to choose the group and venue, as well, so we're only producing one master book for everyone.

```
import plotly.express as px
```

```
import pandas as pd
df = pd.read_csv("thingspeak-feed.csv")
df["timestamp"] =
pd.to_datetime(df['created_at'])
```

*#animation_frame and animation_group
should make it possible to add a range
slider??*

```
fig = px.line(df, x='timestamp',
y='field1', range_x=['2021-12-24', '2021-12-26'], range_y=[0, 20], title="Midnight  
mass and Christmas morning services in a  
worship space.")
fig.add_vline(x='2021-12-24 23:00')
fig.add_vline(x='2021-12-25 00:00')
fig.add_vline(x='2021-12-25 10:00')
fig.add_vline(x='2021-12-25 11:30')
```

```
fig.show()
```



Midnight mass and Christmas morning services

