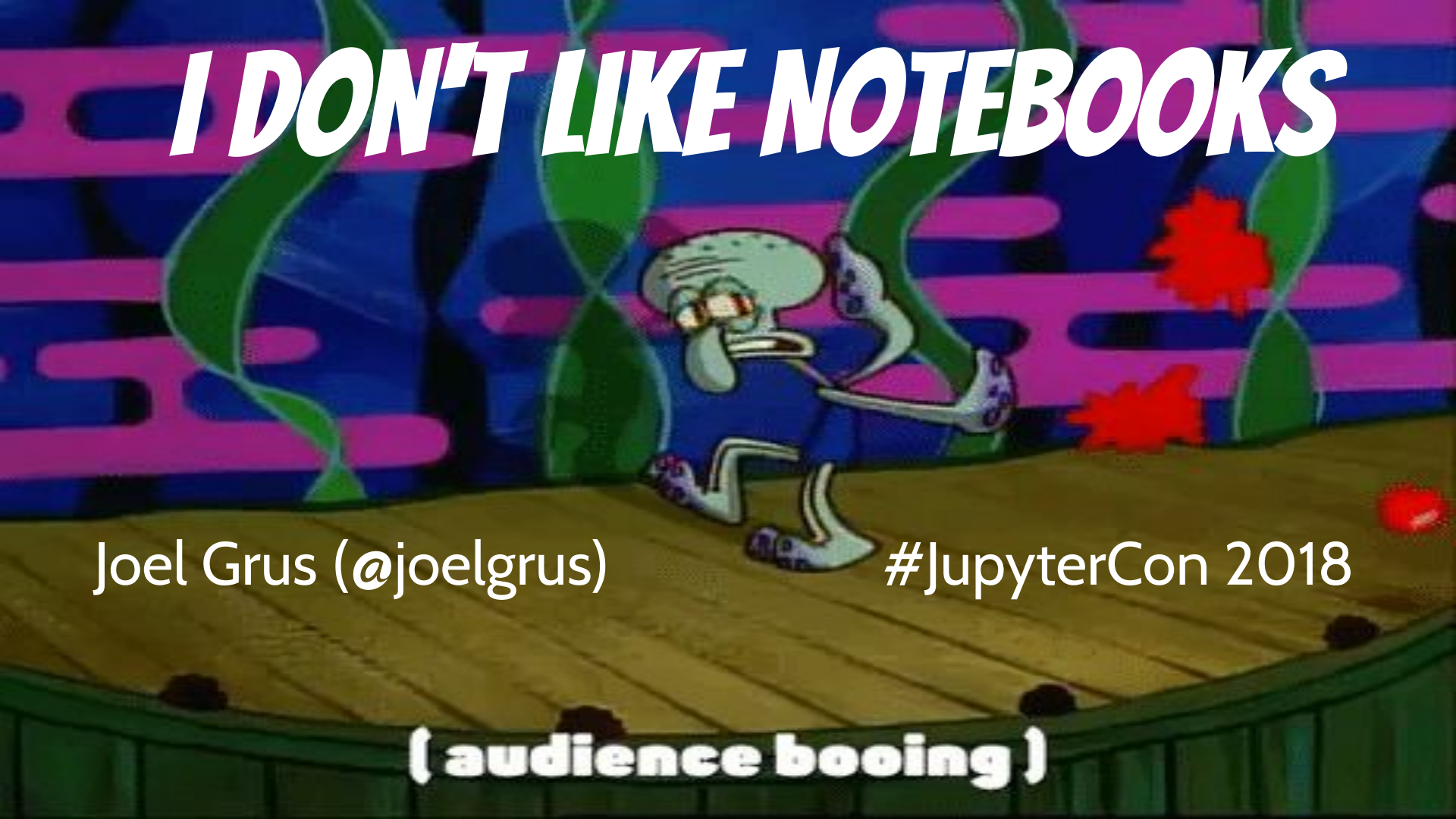
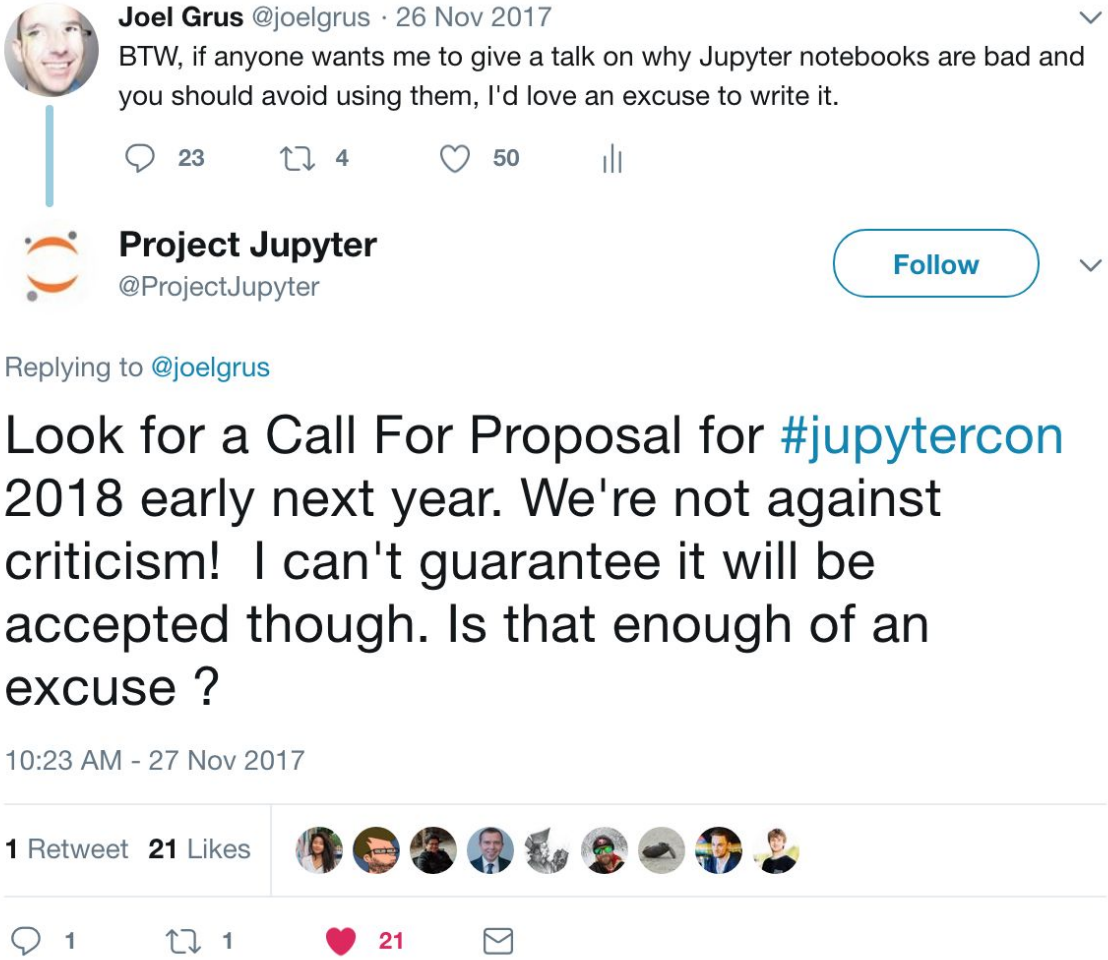

I DON'T LIKE NOTEBOOKS
Joel Grus (@joelgrus)
#JupyterCon 2018
(audience booing)


There are dozens of us!

HOW WE ENDED UP HERE



Joel Grus @joelgrus · 26 Nov 2017

BTW, if anyone wants me to give a talk on why Jupyter notebooks are bad and you should avoid using them, I'd love an excuse to write it.

23 4 50

Project Jupyter
@ProjectJupyter

Follow

Replying to @joelgrus

Look for a Call For Proposal for #jupytercon 2018 early next year. We're not against criticism! I can't guarantee it will be accepted though. Is that enough of an excuse ?

10:23 AM - 27 Nov 2017

1 Retweet **21** Likes

1 1 21

THE IDEA WAS RECEIVED PRETTY WARMLY

Jeremy Howard @jeremyphoward · 27 Nov 2017

Most people I know at the top of Kaggle use notebooks

2 2

Joel Grus @joelgrus · 27 Nov 2017

fantastic, that means there's a lot of really smart people who are just the target audience for my talk 😇

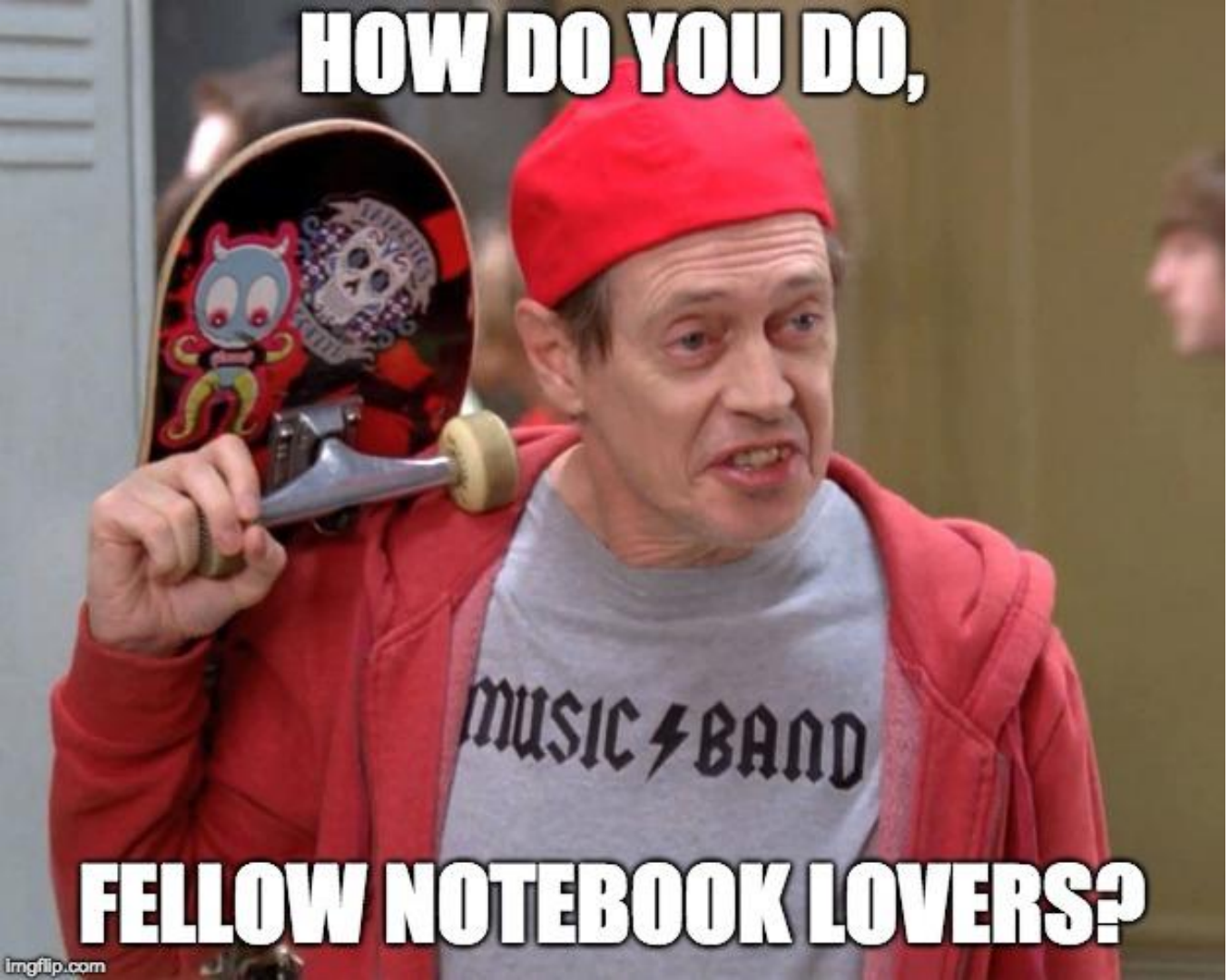
1

Jeremy Howard @jeremyphoward · 27 Nov 2017

I think it means @vboykis is right.

Convincing people not to use a tool which has repeatedly made people more effective seems like an unhelpful way to spend time, frankly

1 6



ABOUT ME

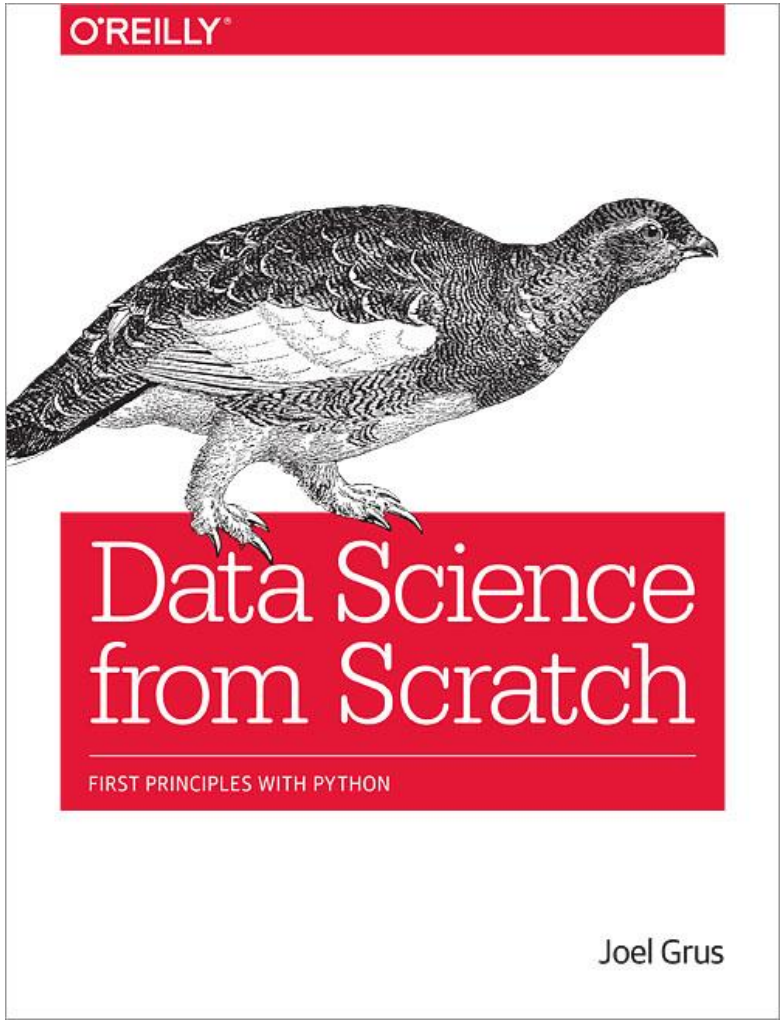
ALLEN INSTITUTE *for* ARTIFICIAL INTELLIGENCE

AllenNLP

An open-source NLP research library, built on PyTorch

RESEARCH ENGINEER @ AI2

KNOW A FEW THINGS ABOUT PYTHON



KNOW A FEW THINGS ABOUT DATA SCIENCE

KNOW A FEW THINGS ABOUT SOFTWARE ENGINEERING

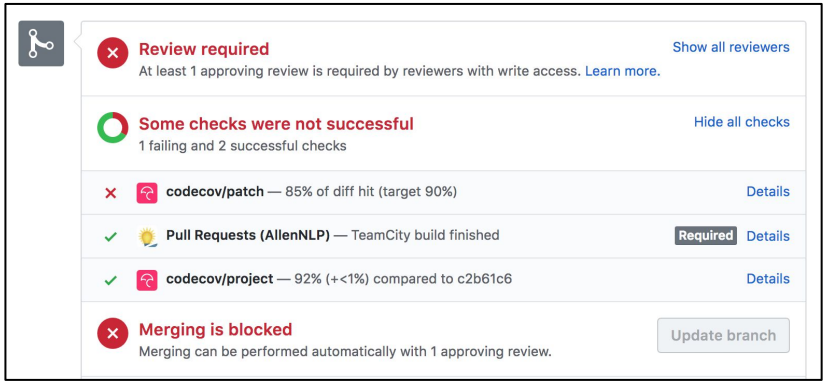
allenai / allennlp

Unwatch 119 | Star 2,107 | Fork 320

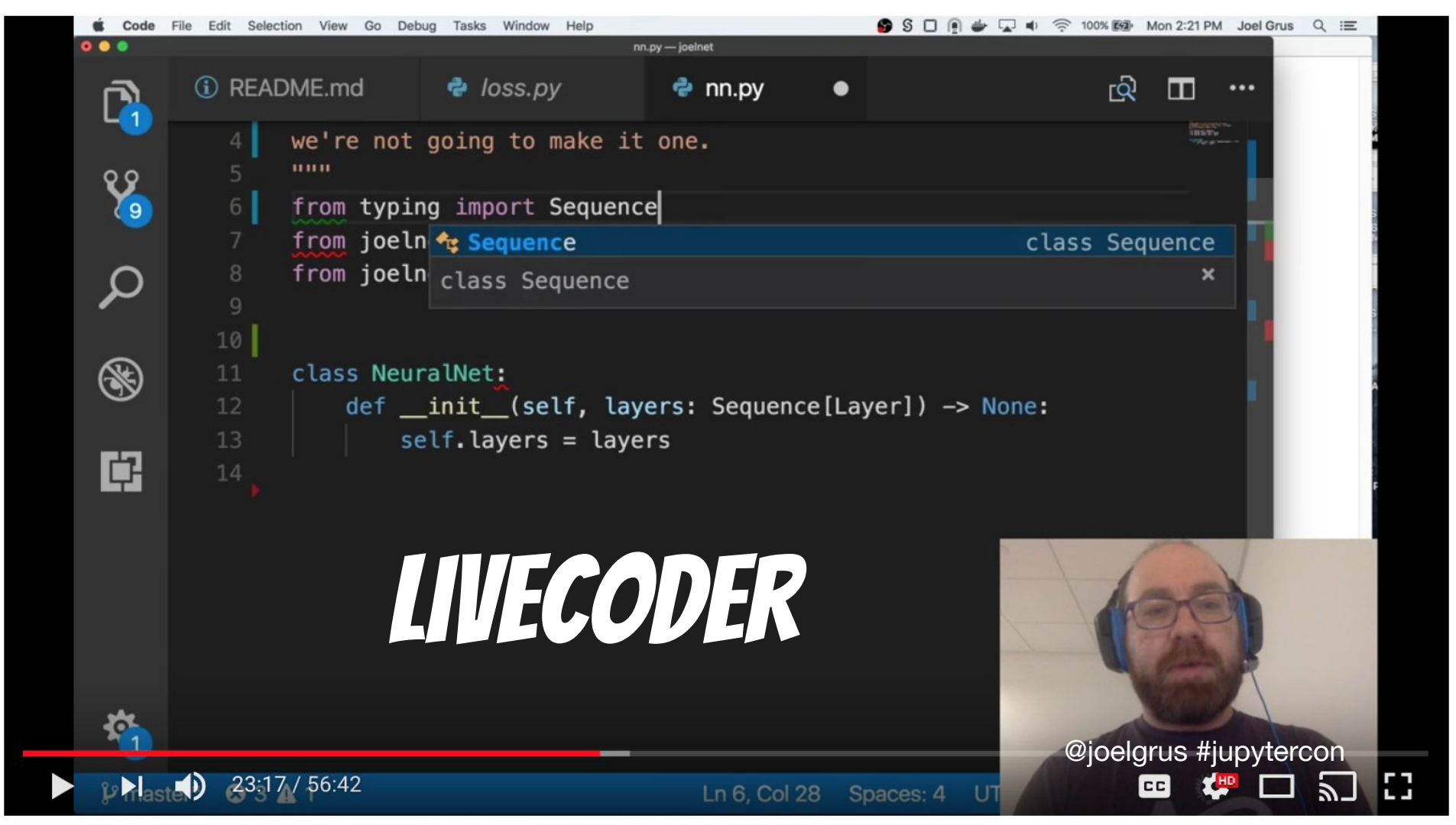
Code | Issues 64 | Pull requests 24 | Projects 2 | Insights | Settings

An open-source NLP research library, built on PyTorch. http://www.allennlp.org/ Edit

pytorch nlp natural-language-processing deep-learning data-science python Manage topics




```
[21:23:30] + Step 1/8: Docker Pull (for caching) (Command Line)  (18s)
[21:23:49] + Step 2/8: Docker Build (Docker Build)  (7s)
[21:23:56] + Step 3/8: Docker Push SHA (Command Line)
[21:23:56] + Step 4/8: Unit Tests (pytest) (Command Line)  (7m:05s)
[21:31:02] + Step 5/8: Linter (pylint) (Command Line)  (2m:23s)
[21:33:26] + Step 6/8: Type Checker (mypy) (Command Line)  (20s)
[21:33:46] + Step 7/8: Build Docs (Command Line)  (1m:02s)
[21:34:49] + Step 8/8: Check Docs (Command Line)  (5s)
```


Code File Edit Selection View Go Debug Tasks Window Help
100% Mon 2:21 PM Joel Grus
nn.py — joelnet
README.md
loss.py
nn.py
we're not going to make it one.
"""
from typing import Sequence
from joeln
from joeln
Sequence
class Sequence
class Sequence
class NeuralNet:
def __init__(self, layers: Sequence[Layer]) -> None:
self.layers = layers
LIVECODER
@joelgrus #jupytercon
23:17 / 56:42
Ln 6, Col 28
Spaces: 4

BLOGGER

Fizz Buzz in Tensorflow

interviewer: Welcome, can I get you coffee or anything? Do you need a break?

me: No, I've probably had too much coffee already!

interviewer: Great, great. And are you OK with writing code on the whiteboard?

me: It's the only way I code!

me: And, of course, the prediction will just be the largest output:

```
predict_op = tf.argmax(py_x, 1)
```

interviewer: Before you get *too far* astray, the problem you're *supposed to be* solving is to generate fizz buzz for the numbers from 1 to 100.

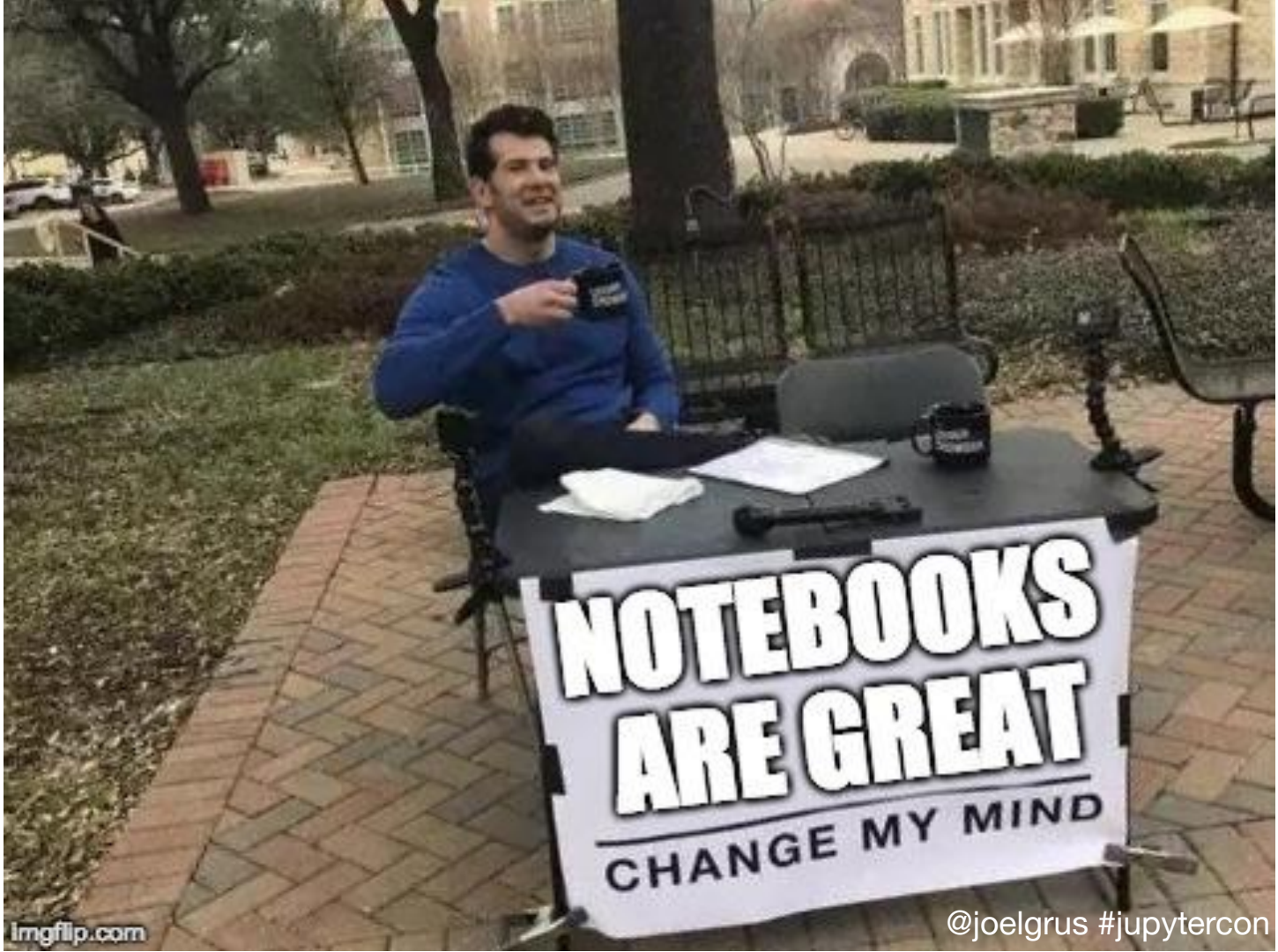
me: Oh, great point, the `predict_op` function will output a number from 0 to 3, but we want a "fizz buzz" output:

```
def fizz_buzz(i, prediction):
    return [str(i), "fizz", "buzz", "fizzbuzz"][prediction]
```

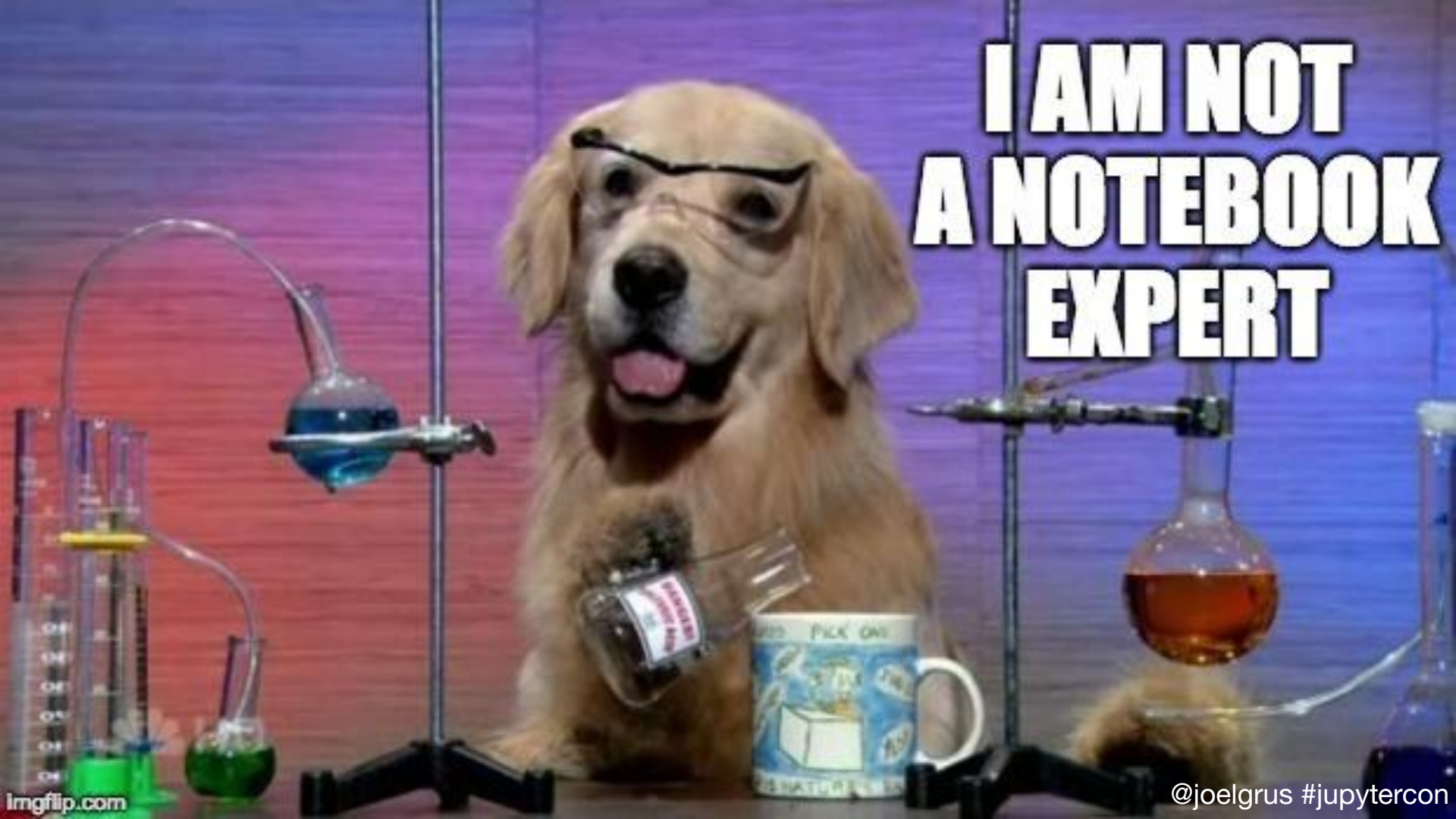
interviewer: ...

PODCAST
CO-HOST

WHAT I ASSUME ABOUT YOU



A COUPLE OF CAVEATS


I AM NOT
A NOTEBOOK
EXPERT
imgflip.com
@joelgrus #jupytercon

File Edit View Run Kernel Tabs Settings Help

Files Running Commands Cell Tools Tabs

Lorenz.ipynb

Code Python 3

We explore the Lorenz system of differential equations:

$$\dot{x} = \sigma(y - x)$$
$$\dot{y} = \rho x - y - xz$$
$$\dot{z} = -\beta z + xy$$

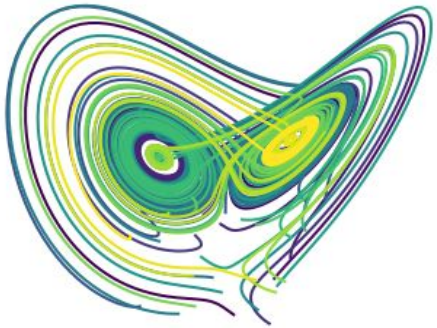
Let's change (σ, β, ρ) with ipywidgets and examine the trajectories.

In [2]:
```
from lorenz import solve_lorenz
w=interactive(solve_lorenz,sigma=(0.0,50.0),rho=(0.0,50.0))
w
```

sigma 10.00

beta 2.67

rho 28.00



lorenz.py

```
def solve_lorenz(sigma=10.0, beta=8./3, rho=28.0):
    """Plot a solution to the Lorenz differential equations."""

    max_time = 4.0
    N = 30

    fig = plt.figure()
    ax = fig.add_axes([0, 0, 1, 1], projection='3d')
    ax.axis('off')

    # prepare the axes limits
    ax.set_xlim((-25, 25))
    ax.set_ylim((-35, 35))
    ax.set_zlim((5, 55))

    def lorenz_deriv(x_y_z, t0, sigma=sigma, beta=beta, rho=rho):
        """Compute the time-derivative of a Lorenz system."""
        x, y, z = x_y_z
        return [sigma * (y - x), x * (rho - z) - y, x * y - beta * z]

    # Choose random starting points, uniformly distributed from -15 to 15
    np.random.seed(1)
    x0 = -15 + 30 * np.random.random((N, 3))

    # Solve for the trajectories
    t = np.linspace(0, max_time, int(250*max_time))
    x_t = np.asarray([integrate.odeint(lorenz_deriv, x0i, t)
                      for x0i in x0])

    # choose a different color for each trajectory
    colors = plt.cm.viridis(np.linspace(0, 1, N))

    for i in range(N):
        x, y, z = x_t[i,:,:].T
        lines = ax.plot(x, y, z, '-', c=colors[i])
        plt.setp(lines, linewidth=2)
    angle = 104
    ax.view_init(30, angle)
```

THINGS I DO LIKE ABOUT NOTEBOOKS

Mixed Markdown and Code

I love well-documented code!

Accordingly, I love the idea of mixing markdown and code.

For example, one underappreciated Python library is `itertools`:

In [1]:
```
import itertools

binary_numbers = [''.join(digits)
                  for digits in itertools.product(*[['0', '1'] for _ in range(10)])]
binary_numbers[:8]
```

Out[1]:
```
['0000000000',
 '0000000001',
 '0000000010',
 '0000000011',
 '0000000100',
 '0000000101',
 '0000000110',
 '0000000111']
```

In [ ]:

jupyter loss Last Checkpoint: 9 hours ago (unsaved changes)

File Edit View Insert Cell Kernel Help Python 3

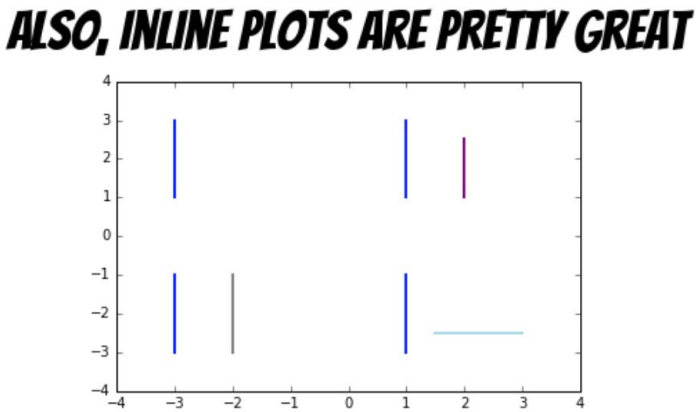
Code CellToolbar

In [1]:
```
%matplotlib inline
```

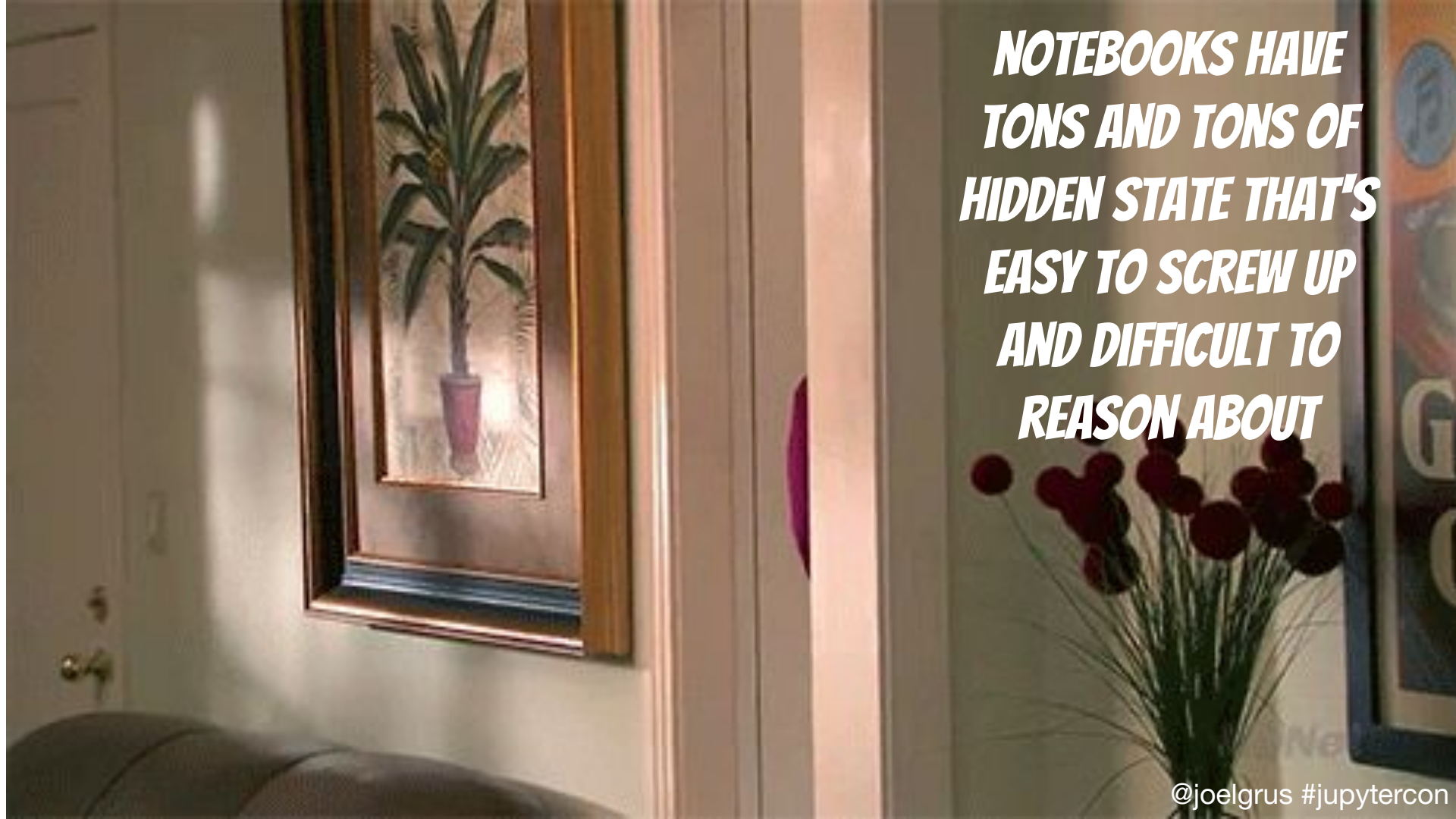
In [2]:
```
from matplotlib import pyplot as plt
import matplotlib.font_manager as fm
prop = fm.FontProperties(fname='/Users/joelg/Library/Fonts/skyfonts-google/Bangers regular.ttf')
```

In [3]:
```
plt.axis([-4, 4, -4, 4])
plt.plot([-3, -3], [1, 3], 'blue', linewidth=2)
plt.plot([1, 1], [1, 3], 'blue', linewidth=2)
plt.plot([2, 2], [1, 2.5], 'purple', linewidth=2)
plt.plot([-3, -3], [-3, -1], 'blue', linewidth=2)
plt.plot([-2, -2], [-3, -1], 'gray', linewidth=2)
plt.plot([1, 1], [-3, -1], 'blue', linewidth=2)
plt.plot([1.5, 3], [-2.5, -2.5], 'lightblue', linewidth=2)
plt.title("Also, Inline Plots are Pretty Great", y=1.08, size=40, fontproperties=prop)
```

Out[3]: <matplotlib.text.Text at 0x10eb23080>



THINGS I DON'T LIKE ABOUT NOTEBOOKS


NOTEBOOKS HAVE TONS AND TONS OF HIDDEN STATE THAT'S EASY TO SCREW UP AND DIFFICULT TO REASON ABOUT
@joelgrus #jupytercon

SIMPLE EXAMPLE

```
(allennlp) OS-XImage:~ joelg$ ipython
Python 3.6.2 |Continuum Analytics, Inc.| (default, Jul 20 2017, 13:14:59)
Type 'copyright', 'credits' or 'license' for more information
IPython 6.4.0 -- An enhanced Interactive Python. Type '?' for help.
REPL completion using Jedi 0.12.0

In [1]: def f(x): return x + 2

In [2]: y = f(2)

In [3]: y == 4
Out[3]: True

In [4]: print(y)
4

In [5]:
```

jupyter madness

File Edit View Insert Cell Kernel Help | Python 3

Code CellToolbar

```
In [3]: def f(x): return x + 2

In [2]: y = f(2)

In [4]: y == 4
Out[4]: False

In [5]: print(y)
        5
```

jupyter madness

File Edit View Insert Cell Kernel Help | Python 3

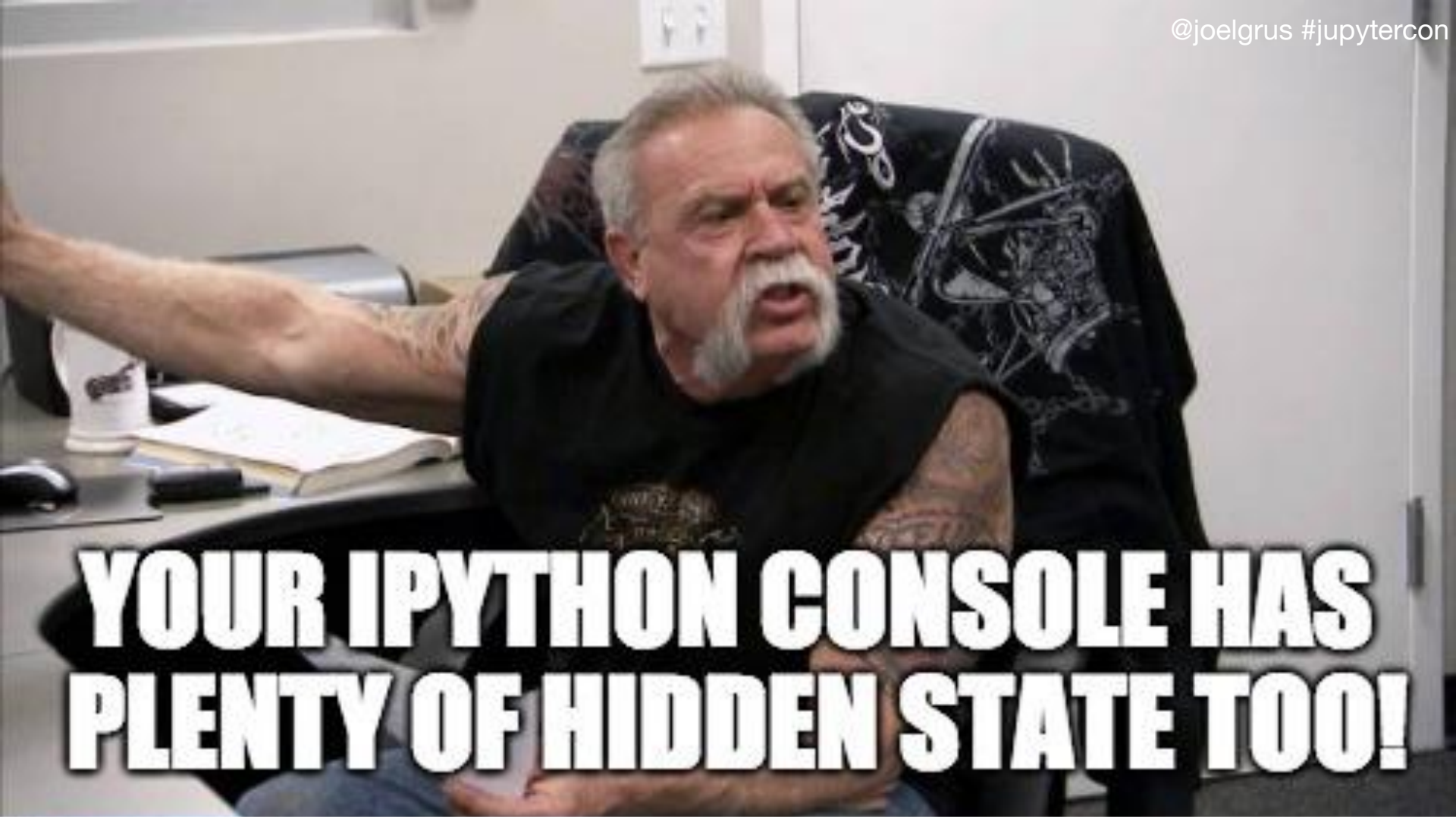
Code | CellToolbar

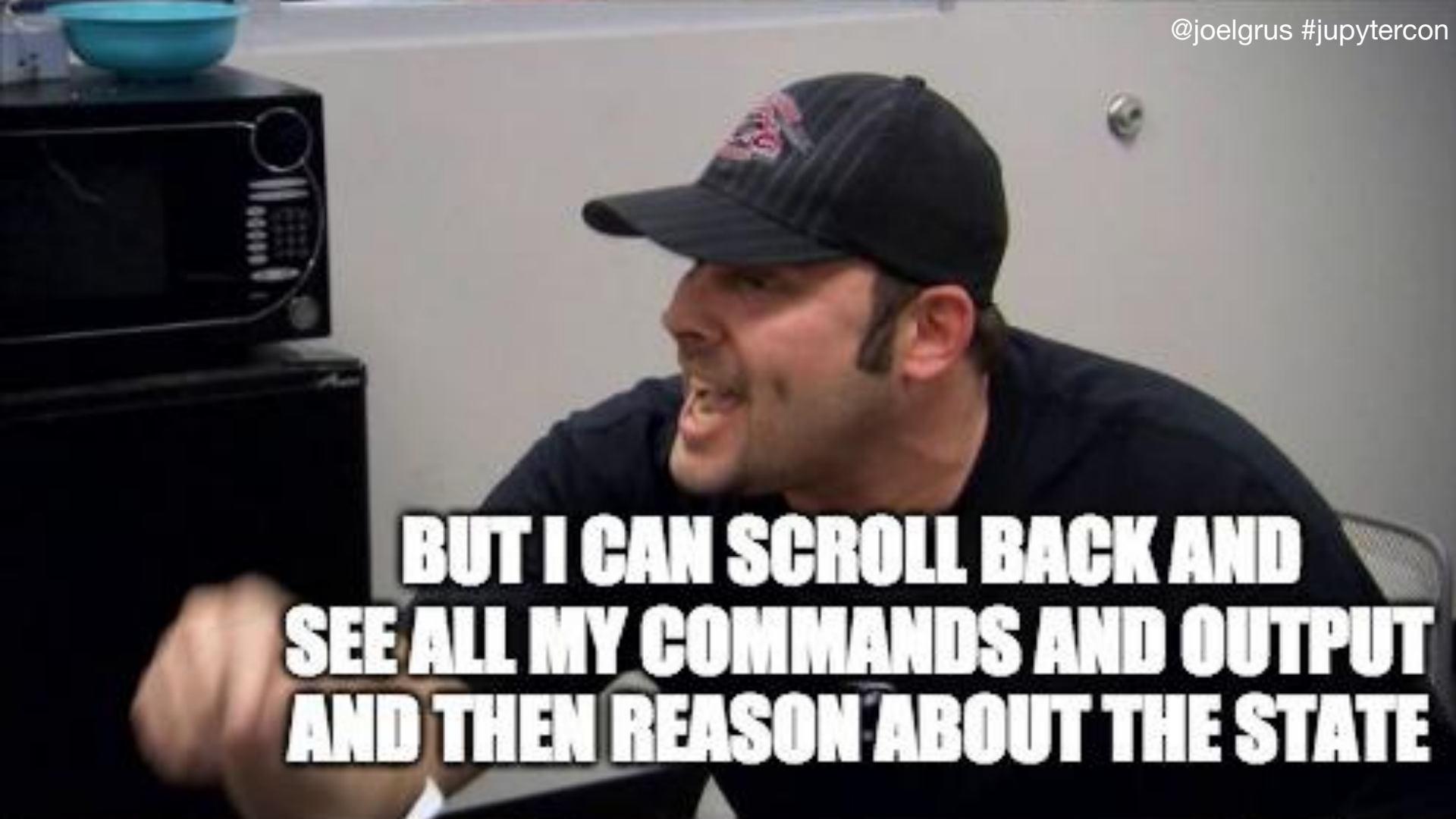
```
In [1]: def f(x): return x + 2

In [3]: y = f(2)

In [4]: y == 4
Out[4]: False

In [5]: print(y)
        5
```

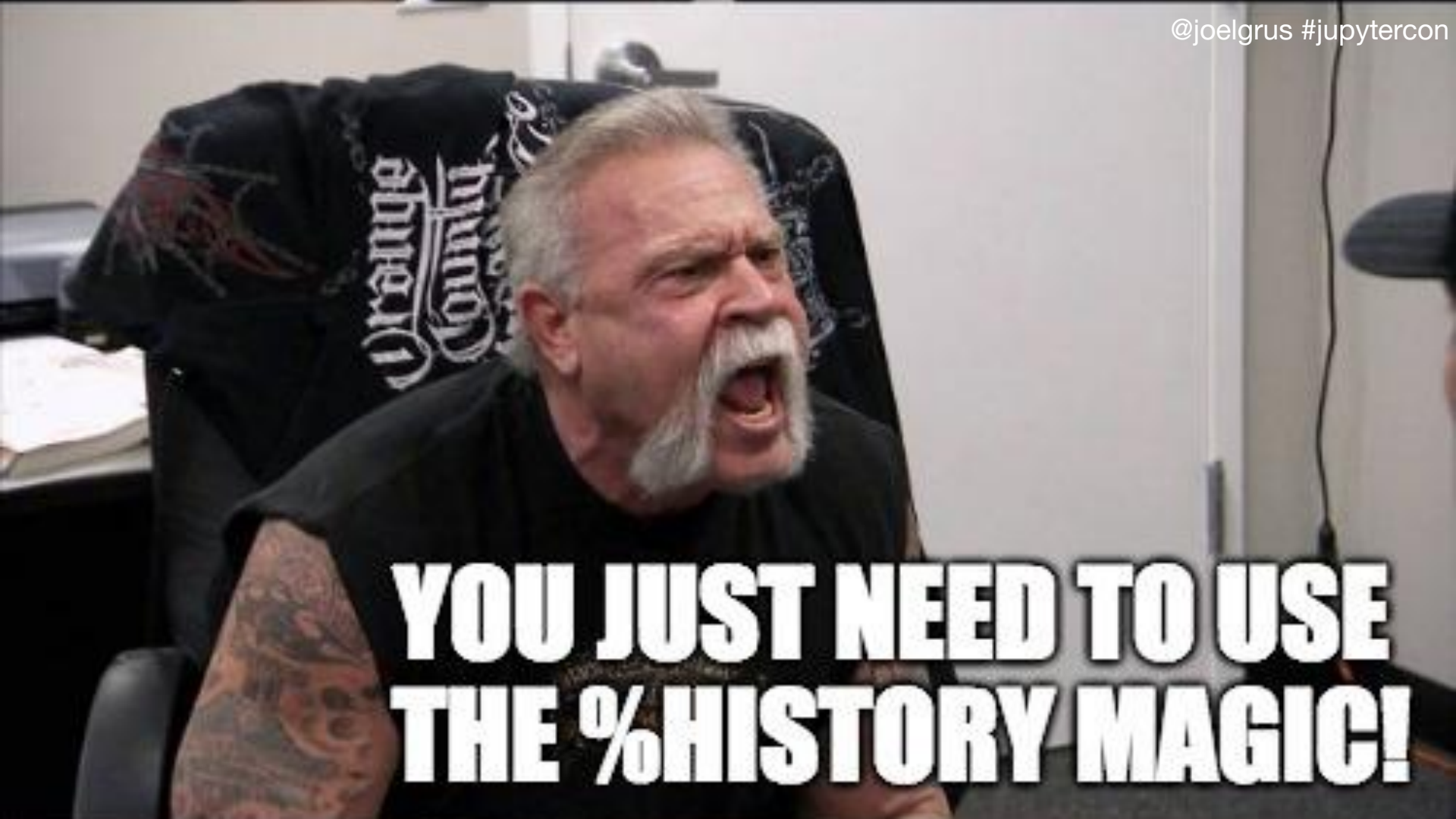

NOTEBOOKS ARE GREAT FOR ITERATIVE DEVELOPMENT
NOTEBOOKS ARE DANGEROUS UNLESS YOU RUN EACH CELL EXACTLY ONCE, IN ORDER

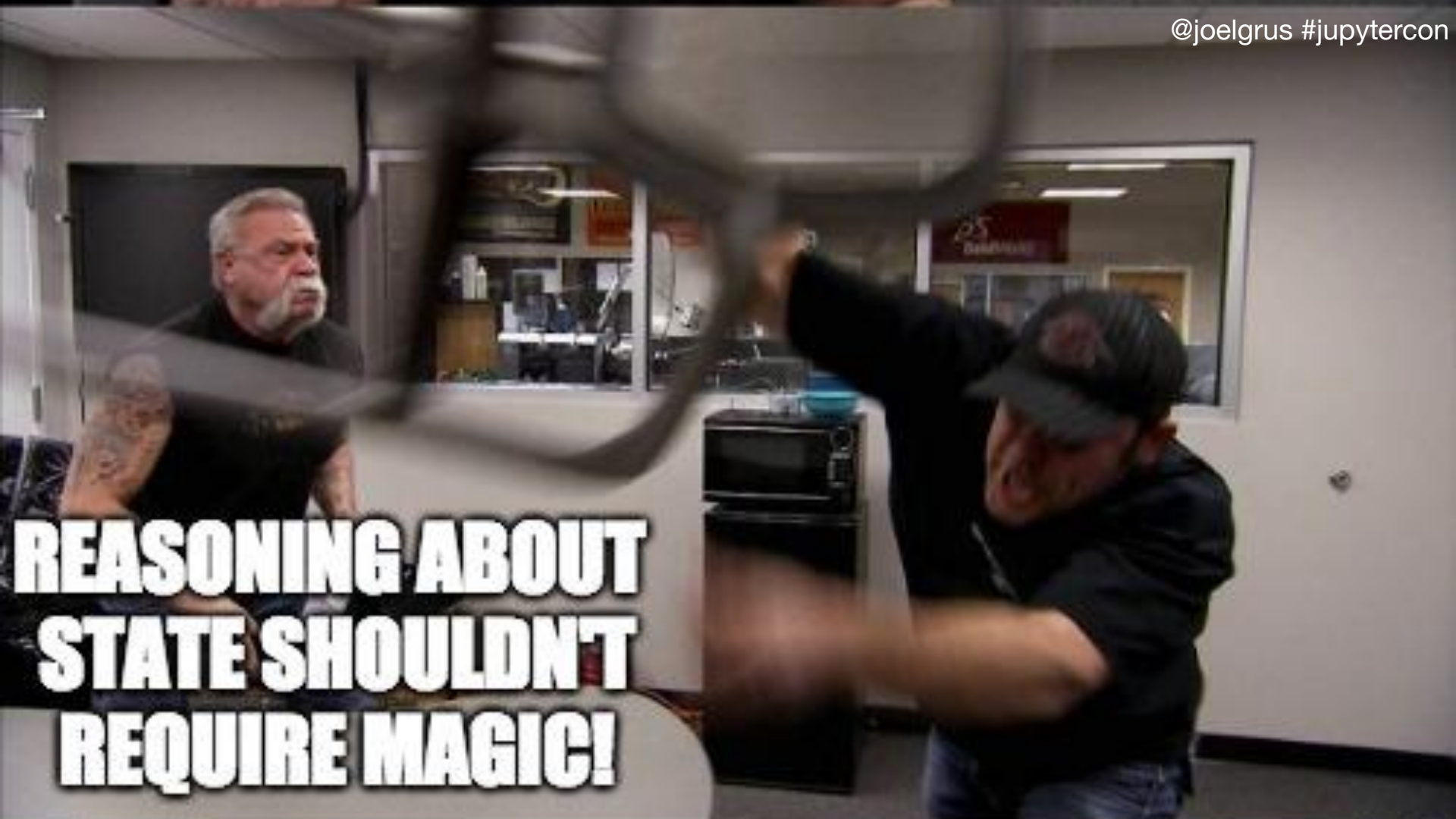

JAKE-CLARK.TUMBLR

imgflip.com


YOUR IPYTHON CONSOLE HAS
PLENTY OF HIDDEN STATE TOO!


BUT I CAN SCROLL BACK AND
SEE ALL MY COMMANDS AND OUTPUT
AND THEN REASON ABOUT THE STATE


YOU JUST NEED TO USE
THE %HISTORY MAGIC!


REASONING ABOUT
STATE SHOULDN'T
REQUIRE MAGIC!


THOSE WHO DON'T BELIEVE
IN MAGIC WILL NEVER FIND IT!
imgflip.com

ONE MORE

jupyter madness

File Edit View Insert Cell Kernel Help Python 3

Code CellToolbar

```
In [1]: def f(x): return x + 2

In [2]: y = f(2)

In [3]: y == 4
Out[3]: False

In [4]: print(y)
        5
```

jupyter show that cell has been edited

All Images News Videos Shopping More Settings Tools

About 114,000 results (0.61 seconds)

Notebook Basics — Jupyter Notebook 5.5.0 documentation
jupyter-notebook.readthedocs.io/en/stable/examples/.../Notebook%20Basics.html
Edit mode is indicated by a green **cell** border and a prompt **showing** in the editor ... The modal user interface of the **Jupyter** Notebook **has been** optimized for ...

Edit meta-data to suppress code input and cell numbers for reveal.js ...
https://github.com/jupyterlab/jupyterlab/issues/4100
Mar 3, 2018 - from **IPython.display** import YouTubeVideo # a talk about **IPython** at .... will require a jupyterlab implementation after that PR **has been** merged.

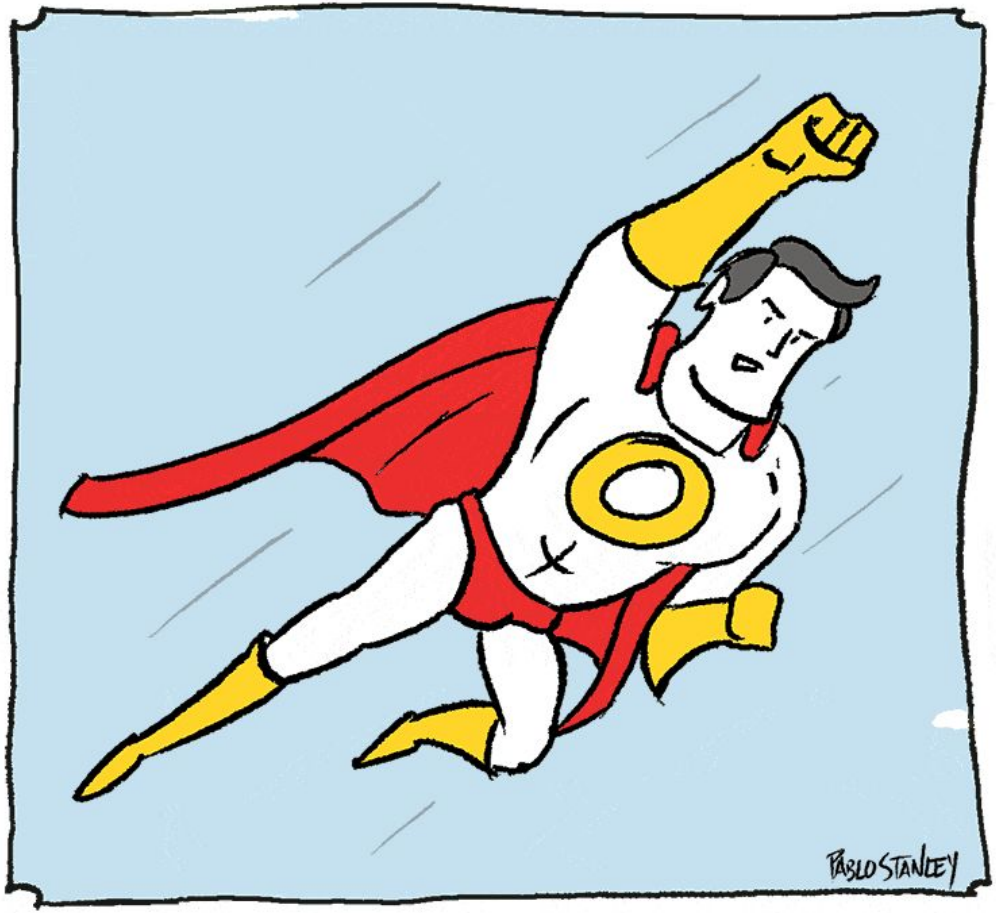
Interrupt planned cell execution · Issue #2340 · jupyter/notebook · GitHub
https://github.com/jupyter/notebook/issues/2340
Mar 27, 2017 - Some **cell** can take a very long time to execute, and it is just then th... ... I'd **suggest** just running these **cells** individually vs. "Run all" or "Run all below". This allows you to **edit** the second **cell** while the first **cell** is running. .... Now even the queue **has** not **been** run at all, there is no way I can fix the bug and put ...

Keyboard shortcuts for ipython notebook 3.1.0 / jupyter · GitHub
https://gist.github.com/kidpixo/f4318f8c8143adee5b40
The **IPython** Notebook has two different keyboard input modes. **Edit** mode allows you to type code/text into a **cell** and is indicated by a green **cell** border. ... H : **show** keyboard shortcut help dialog ... This comment **has been** minimized. **Show** ...

NOTEBOOKS ARE DIFFICULT FOR BEGINNERS


Start off by creating a notebook.
4GIFs.com
@joelgrus #jupytercon

(Hidden state complications are not obvious)

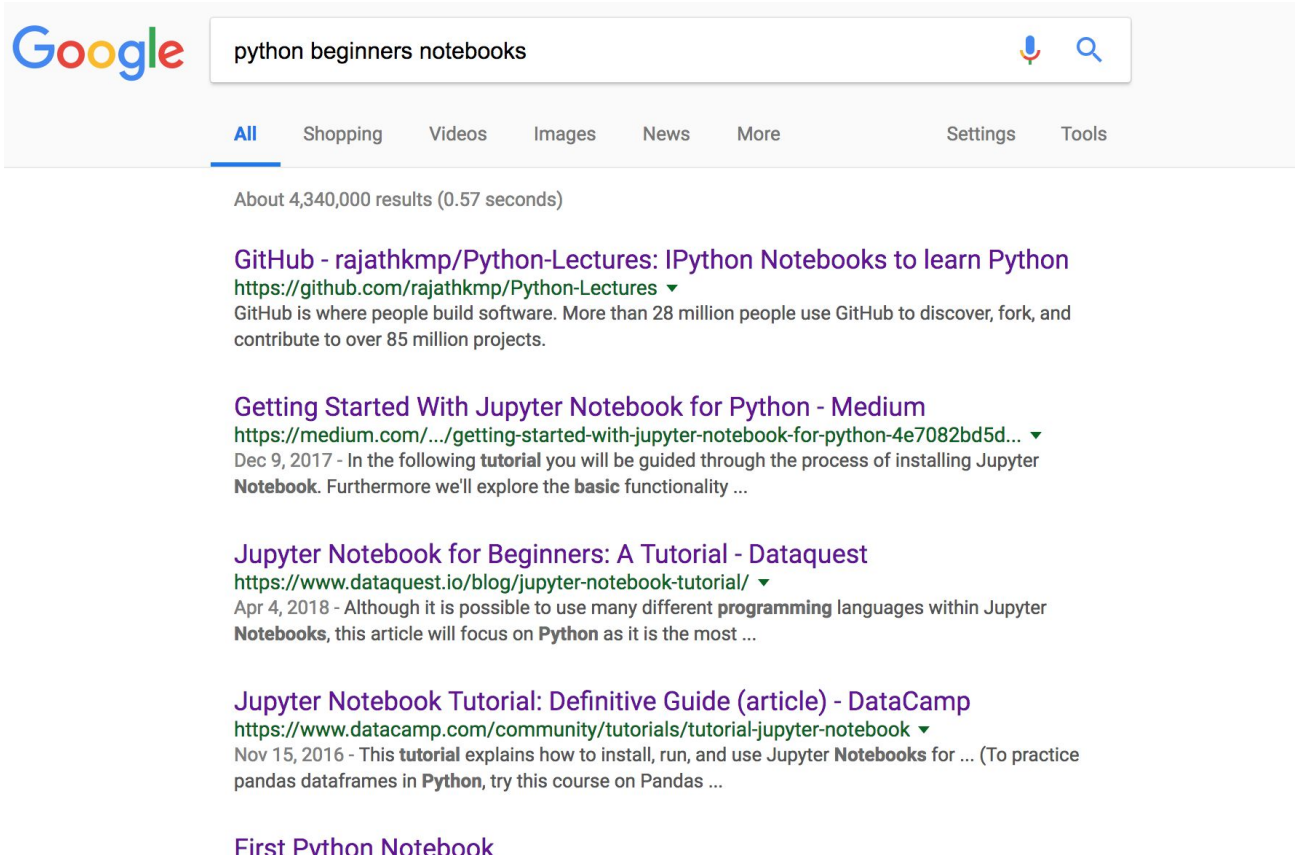


THIS IS MOST PEOPLE'S EXPERIENCE OF HOW CODE WORKS

```
(allennlp) OS-XImage:~ joelg$ ipython
Python 3.6.2 |Continuum Analytics, Inc.| (default, Jul 20 2017, 13:14:59)
Type 'copyright', 'credits' or 'license' for more information
IPython 6.4.0 -- An enhanced Interactive Python. Type '?' for help.
REPL completion using Jedi 0.12.0

In [1]: def f(x): return x + 2

In [2]: y = f(2)

In [3]: y == 4
Out[3]: True

In [4]: print(y)
4

In [5]:
```

the ability to run code snippets in arbitrary order is *weird* and *unintuitive*

BEGINNER TUTORIALS ARE CAVALIER ABOUT (OR SILENT ON) HIDDEN STATE



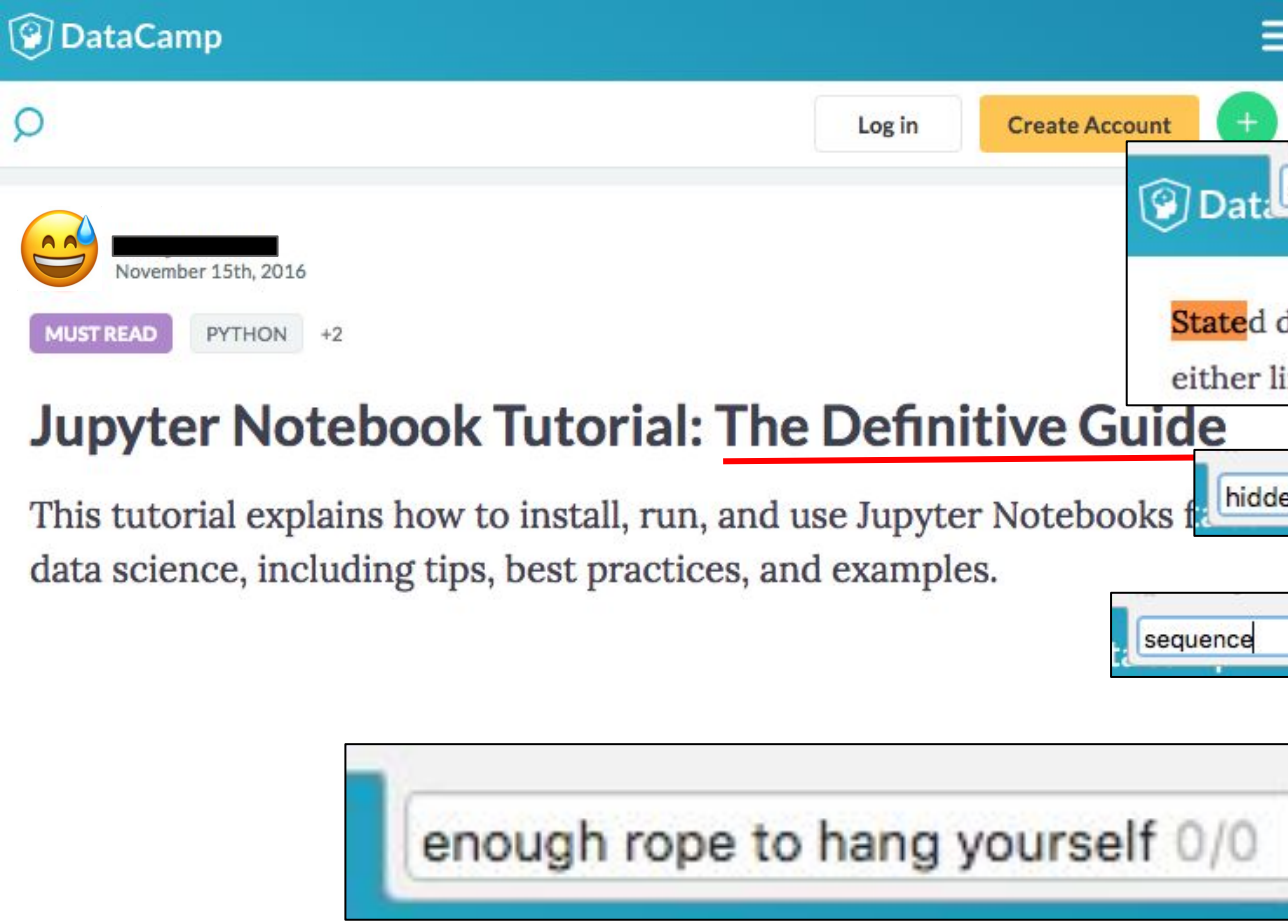
```
return x * x
```

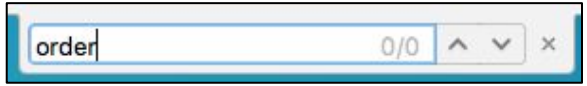
Once we've executed the cell above, we can reference `np` and `square` in any other cell.

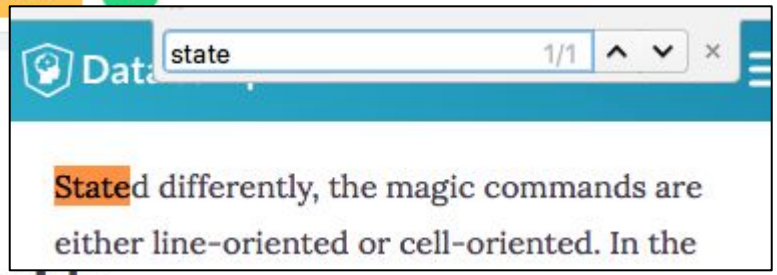
Most of the time, the flow in your notebook will be top-to-bottom, but it's common to go back to make changes. In this case, the order of execution stated to the left of each cell, such as `In [6]`, will let you know whether any of your cells have stale output. And

```
1 squared is 1
```

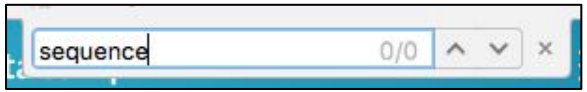
This will work regardless of the order of the cells in your notebook. You can try it yourself, let's print out our variables again.


DataCamp
Log in
Create Account
November 15th, 2016
MUST READ
PYTHON
+2
Jupyter Notebook Tutorial: The Definitive Guide
This tutorial explains how to install, run, and use Jupyter Notebooks f
data science, including tips, best practices, and examples.


order
0/0


Data
state
1/1
Stated differently, the magic commands are
either line-oriented or cell-oriented. In the


hidden
0/0


sequence
0/0

enough rope to hang yourself 0/0

Analytics Vidhya
Learn everything about analytics

Sapient

BIG BREAK IN
BIG DATA
TALENT HUNT FOR DATA ENG
GET A CHANCE TO WIN AN IF
& MACBOOK AIR

LEARN ENGAGE COMPETE GET HIRED TRAININGS STUDENT DATAFEST DATAHACK

Home › Data Science › Comprehensive Beginner's Guide to Jupyter Notebooks for Data Science & Machine Learning

DATA SCIENCE PYTHON

Comprehensive Beginner's Guide to Jupyter Notebooks for Data S Learning

[redacted] MAY 24, 2018

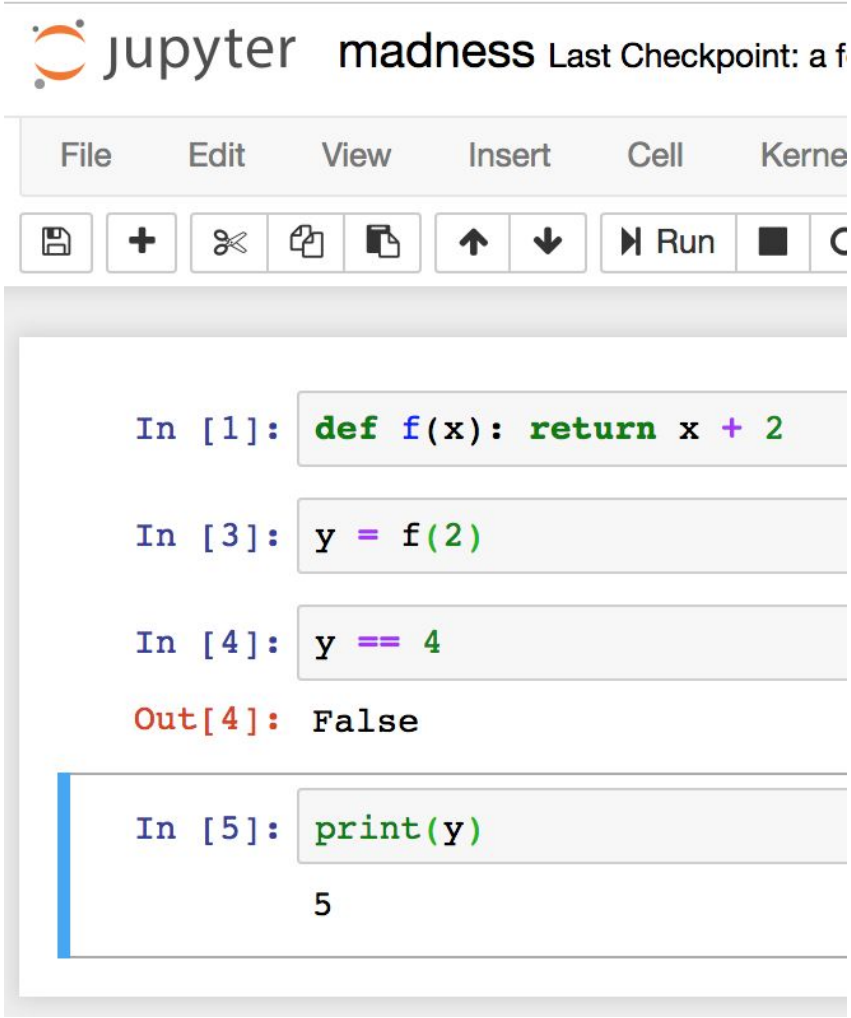
know how you lived without them!

order 1/2

A Jupyter Notebook offers two different keyboard input modes – Command and Edit. Command mode binds the keyboard to notebook level commands and is indicated by a grey cell border with a blue left margin. Edit mode allows you to type text (or code) into the active cell and is indicated by a green cell border.

Jump between command and edit mode using Esc and Enter, respectively. Try it out right now!

@joelgrus #jupytercon



as notebook experts, the problem here is obvious to you

(also this is a *really* simple example)

for beginners, with dozens of cells and more complex code, this is utterly confusing

(I know this because they come to me with their problems)

Joel Grus @joelgrus · 26 Nov 2017

BTW, if anyone wants me to give a talk on why Jupyter notebooks are bad and you should avoid using them, I'd love an excuse to write it.

23 4 50

IN FACT, MY ORIGINAL ANGRY TWEET WAS PROMPTED BY THE [LARGE NUMBER]TH OCCASION OF ME HELPING A NEWCOMER TO PYTHON WHO USED A NOTEBOOK BECAUSE THEY HEARD THAT WAS *THE THING TO DO* AND WHO FOUND "PYTHON" BEWILDERING 100% ON ACCOUNT OF MESSED UP HIDDEN STATE IN THEIR NOTEBOOK DUE TO NORMAL BEGINNER SLOPPINESS AND OUT-OF-ORDER EXECUTION

Lots of beginners use notebooks; clearly it can't be that difficult.

It's not that it's insurmountably difficult, it's that the out-of-order execution makes learning Python *more confusing than it needs to be*.

NOTEBOOKS ENCOURAGE BAD HABITS


UNTITLED25.IPYNB
UNTITLED24.IPYNB
imgflip.com

[redacted]

@[redacted]

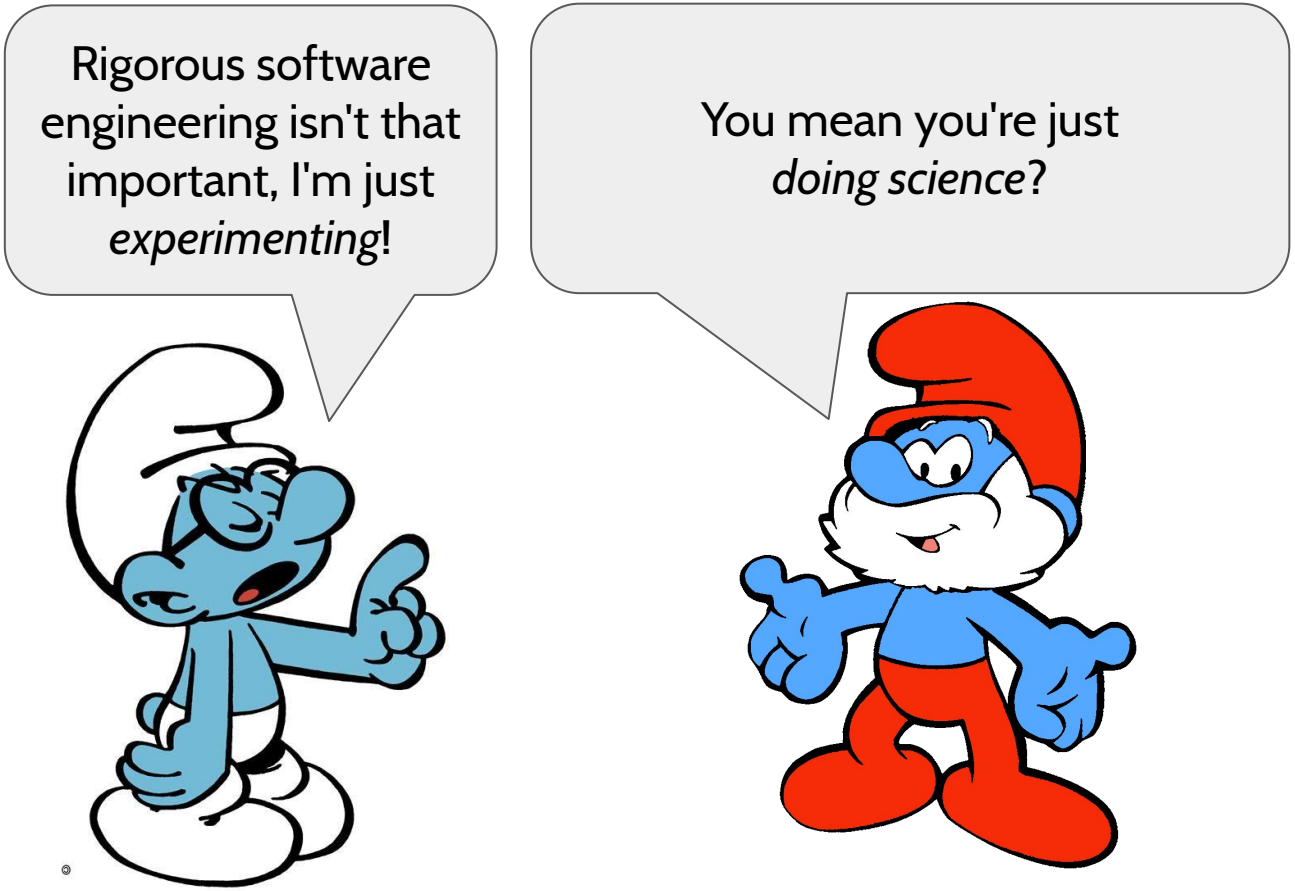
Follow

Data science code doesn't need to follow the rules of good software engineering, because data science is not about creating software but about experimenting with building prototypes of models. 👈 Great tip from @jeremyphoward

4:18 AM - 2 May 2018

4 Retweets **26** Likes

apparently this was a slight misrepresentation of the "tip", although I didn't learn that until after I started a Twitter war over it


Rigorous software engineering isn't that important, I'm just *experimenting*!
You mean you're just *doing science*?

I just want to see if my model *works* before I put it into production.

Don't you need to write *correct code* to make sure it works?

ANTI-PATTERN: THE NEED FOR SPEED

[redacted] @[redacted] · May 2

Replying to @joelgrus @[redacted] @jeremyphoward

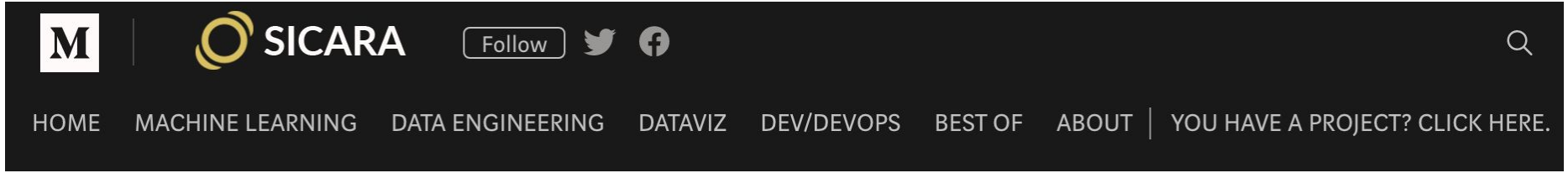
Why? Granted terrible code isn't acceptable. But if you're prototyping shouldn't speed be prioritized over beauty? Provided that the code is working properly and efficiently

2 2

Joel Grus @joelgrus · May 2

good software engineering is the safest/easiest way to be (somewhat) confident that your code works correctly and efficiently

2 14



Applause from Paco Nathan and 45 others

[redacted] Follow

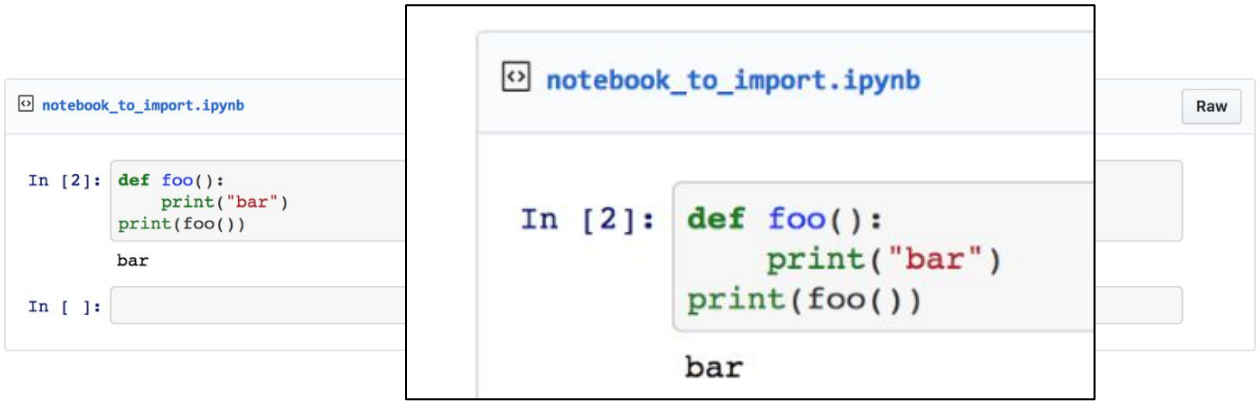
Sep 25, 2017 · 4 min read

Present Your Data Science Results in a Jupyter Notebook, the Right Way

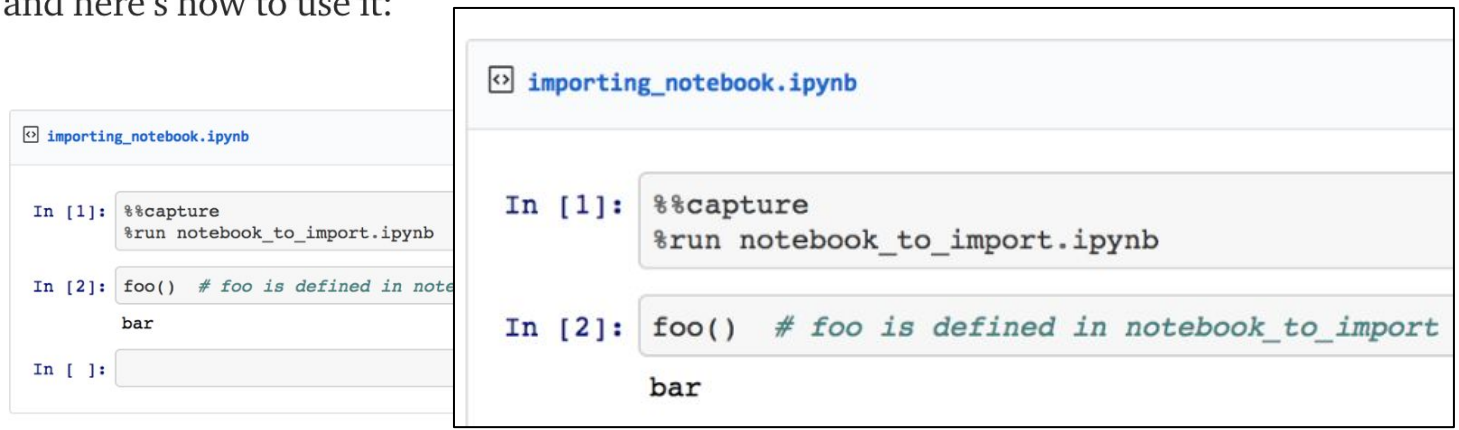
How to import one Jupyter notebook into another

The greedy solution

Here's some code I wrote, but I don't want my client or decision maker to see:



and here's how to use it:



"Here's some code I wrote, but I don't want my client or decision maker to see"

The greedy solution

Here's some code I wrote, but I don't want my

notebook_to_import.ipynb

```
In [2]: def foo():
            print("bar")
        print(foo())
```

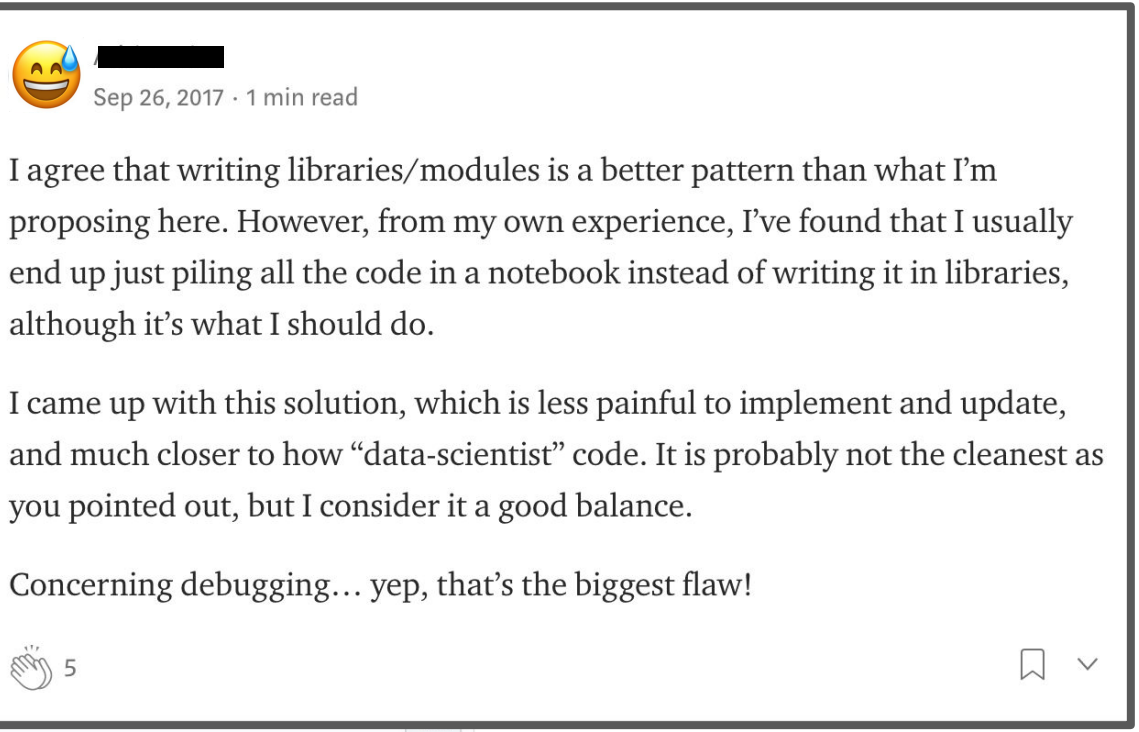
```
bar
```

```
In [ ]:
```

and here's how to use it:

importing_notebook.ipynb

```
In [1]: %%capture
        %run notebook_to_import.ipynb
```



the Right Way


NOTEBOOKS
DISCOURAGE
GOOD HABITS
@joelgrus #jupytercon

joelgrus / joelnet

Code | Issues 0 | Pull reque

Branch: master ▾ joelnet / joelnet /

joelgrus second recording

..

__init__.py

data.py

layers.py

loss.py

nn.py

optim.py

tensor.py

train.py

```python
from joelnet.tensor import Tensor


class Layer:
    def __init__(self) -> None:
        self.params: Dict[str, Tensor] = {}
        self.grads: Dict[str, Tensor] = {}

    def forward(self, inputs: Tensor) -> Tensor:
        """
        Produce the outputs corresponding to these inputs
        """
        raise NotImplementedError

    def backward(self, grad: Tensor) -> Tensor:
        """
        Backpropagate this gradient through the layer
        """
        raise NotImplementedError


class Linear(Layer):
    """
    computes output = inputs @ w + b
    """
    def __init__(self, input_size: int, output_size: int) -> None:
        # inputs will be (batch_size, input_size)
        # outputs will be (batch_size, output_size)
        super().__init__()
        self.params["w"] = np.random.randn(input_size, output_size)
        self.params["b"] = np.random.randn(output_size)
```

MODULARITY

```python
class CrfTaggerTest(ModelTestCase):
    def setUp(self):
        super().setUp()
        self.set_up_model(self.FIXTURES_ROOT / 'crf_tagger' / 'experiment.json',
                          self.FIXTURES_ROOT / 'data' / 'conll2003.txt')

    def test_simple_tagger_can_train_save_and_load(self):
        self.ensure_model_can_train_save_and_load(self.param_file)

    @flaky
    def test_batch_predictions_are_consistent(self):
        self.ensure_batch_predictions_are_consistent()
```

```python
def test_forward_pass_runs_correctly(self):
    training_tensors = self.dataset.as_tensor_dict()
    output_dict = self.model(**training_tensors)
    tags = output_dict['tags']
    assert len(tags) == 2
    assert len(tags[0]) == 7
    assert len(tags[1]) == 7
    for example_tags in tags:
        for tag_id in example_tags:
            tag = self.model.vocab.get_token_from_index(tag_id, namespace="labels")
            assert tag in {'O', 'I-ORG', 'I-PER', 'I-LOC'}

def test_mismatching_dimensions_throws_configuration_error(self):
    params = Params.from_file(self.param_file)
    # Make the encoder wrong - it should be 2 to match
    # the embedding dimension from the text_field_embedder.
    params["model"]["encoder"]["input_size"] = 10
    with pytest.raises(ConfigurationError):
        Model.from_params(vocab=self.vocab, params=params.pop("model"))
```

TESTABILITY

type_checking.py

```python
from typing import Generic, TypeVar, Any

T = TypeVar('T')

class TypedLruCache(Generic[T]):
    def __init__(self, max_size: int = 256) -> None:
        raise NotImplementedError

    def get(self, key: Any, default: T) -> T:
        raise NotImplementedError

    def put(self, key: Any, value: T) -> None:
        raise NotImplementedError

my_cache = TypedLruCache[str](max_size=10)
my_cache.put("first_name", 10)
```

```
(allennlp) OS-XImage:junk joelg$ mypy type_checking.py
type_checking.py:16: error: Argument 2 to "put" of "TypedLruCache" has incompatible type "int"; expected "str"
```

AllenNLP

```
[21:23:30] + Step 1/8: Docker Pull (for caching) (Command Line)  (18s)
[21:23:49] + Step 2/8: Docker Build (Docker Build)  (7s)
[21:23:56] + Step 3/8: Docker Push SHA (Command Line)
[21:23:56] + Step 4/8: Unit Tests (pytest) (Command Line)  (7m:05s)
[21:31:02] + Step 5/8: Linter (pylint) (Command Line)  (2m:23s)
[21:33:26] + Step 6/8: Type Checker (mypy) (Command Line)  (20s)
[21:33:46] + Step 7/8: Build Docs (Command Line)  (1m:02s)
[21:34:49] + Step 8/8: Check Docs (Command Line)  (5s)
```

Using AllenNLP as a Library (Part 1) - Datasets and Models

In this tutorial, we take you step-by-step through the process you need to go through to get up and running with your own models on your own data in your own repository, using AllenNLP as an imported library. There is naturally some overlap with things we covered in previous tutorials, but, hey, some repetition is good, right?

We'll use the open research corpus provided by the academic search engine Semantic Scholar, with a heuristically-edited "venue" field. You can follow the link to see the full specification of this data, but it's provided as a JSON-lines file, where each JSON blob has at least these fields:

```
{
  "title": "A review of Web searching studies and a framework for future research",
  "paperAbstract": "Research on Web searching is at an incipient stage. ...",
  "venue": "{AI|ML|ACL}"
}
```

Because we like writing tests (and you should too!), we'll write a test for our `DatasetReader` on some sample data before even writing any code. We downloaded the data and took a sample of 10 papers and made a little test fixture out of them. We'll use this fixture to make sure we can read the data as we expect.

NOTEBOOKS ARE WAY LESS HELPFUL THAN MY TEXT EDITOR

SOME THINGS ARE EASIER DEMONSTRATED


```
xs = [1, 2, 3]
xs.
```

- append
- clear
- copy
- count
- extend
- index
- insert
- pop
- remove
- reverse
- sort
- __add__

```
In [1]: xs = [1, 2, 3]
        xs.
```

```
In [1]: xs = [1, 2, 3]
In [2]: xs.
In [ ]:
```

- xs.append
- xs.clear
- xs.copy
- xs.count
- xs.extend
- xs.index
- xs.insert
- xs.pop
- xs.remove
- xs.reverse

Just run the cell first. And then just press TAB.


```
xs = [1, 2
xs.insert()
```

```
insert(index, object)

param index

L.insert(index, object) -- insert object before index
```

```
In [1]: xs = [1, 2, 3]

In [2]: xs.insert()
```

Just replace the parentheses with a question mark. And then just execute the cell

This also "uses up" a computation, so either I have to leave my ignorance there or have "out of order" cells

```
In [1]: xs = [1, 2, 3]

In [2]: xs.insert?
```

```
Docstring: L.insert(index, object) -- insert object before index
Type:      builtin_function_or_method
```


```
vanpool_riders = 10

cost = 100 * van
```

vanpool_riders

```
In [1]: vanpool_riders = 10

In [ ]: cost = van
```

van_halen_ice_cream_
vanessa_williams_sav
vangelis_chariots_of
vanilla_ice_ninja_ra
vanpool_riders

You may not want to use a filename as your variable name, but some people might!


```
from typing import List

def process(xs: List[int]) -> None:
    xs.
```

```
append
clear
copy
count
extend
index
insert
pop
remove
reverse
sort
__add__
```

```
In [ ]: from typing import List

        def process(xs: List[int]) -> None:
            xs.
```

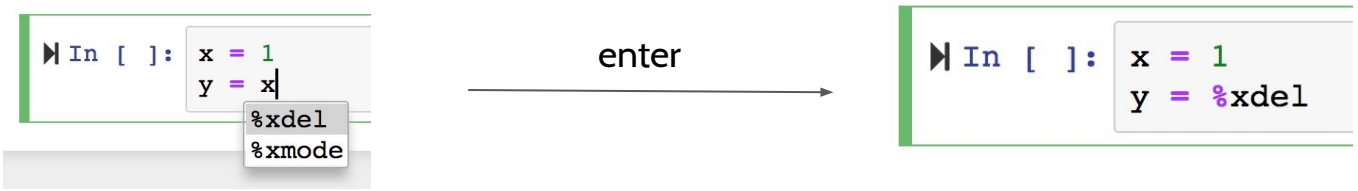
...


```
with open('file.txt') as f:
    f.w
        writable
        write
        writelin
        _checkWr
```

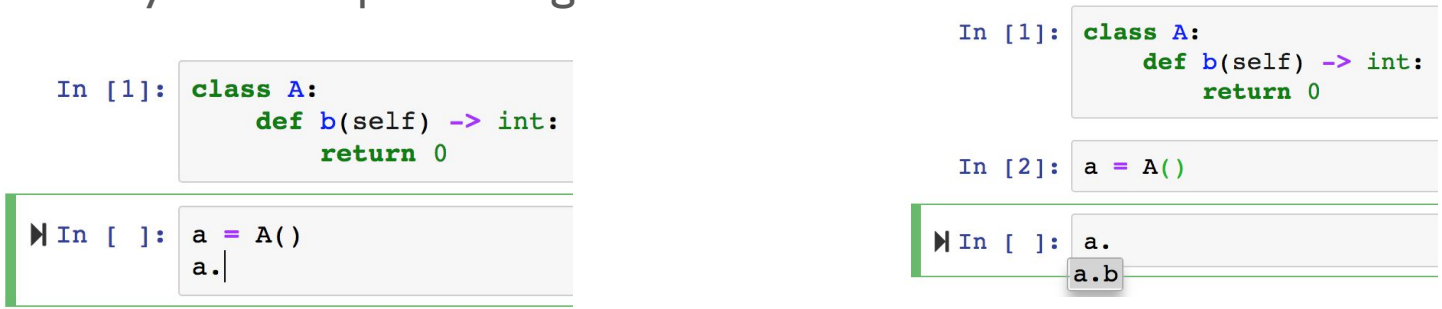
```
In [ ]: with open('file.txt') as f:
            f.
```

I FOUND AN EXTENSION CALLED "HINTERLAND" THAT DOES SOME OF THESE, BUT...

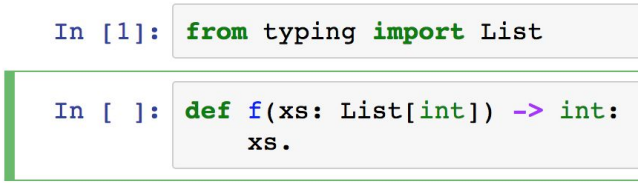
- questionable autocomplete behavior



- still can only autocomplete things that have been executed

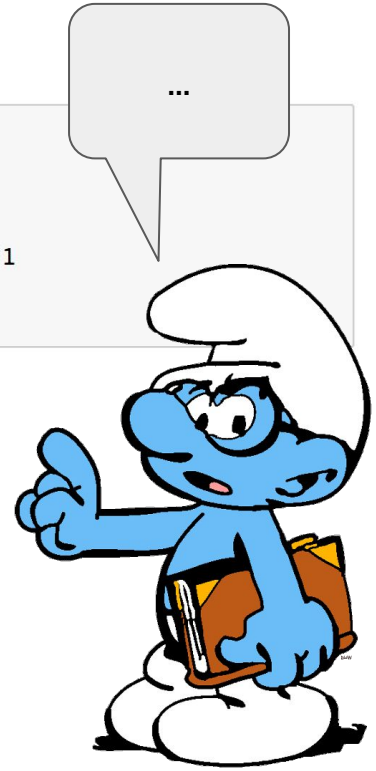


- which means still can't handle type hints or `with` blocks




```
def do_stuff() -> str:
    stuff1 = "something"
    stuff2 = "something else"

    concatenated = stuff1 + stuff1

    return concatenated
```

```
In [1]: def do_stuff() -> str:
            stuff1 = "something"
            stuff2 = "something else"

            concatenated = stuff1 + stuff1

            return concatenated
```




```
def
    [pylint] W0612:Unused variable 'stuff2'
    stuff2: str
    stuff2 = "something else"
```

Verifying PEP8 in iPython notebook c

21

Is there an easy way to check that iPython noteboc
PEP8?

python ipython ipython-notebook pep8

share edit flag

edited Oct 6

3

add a comment

5

Make sure you've the module `pycodestyle` to be able to check your code against the style guide. Then enable the magic function by using the `pycodestyle_magic` module:

```
pip install pycodestyle
pip install pycodestyle_magic
```

- first load the magic in a Jupyter Notebook cell:

```
%load_ext pycodestyle_magic
```

- and then use the function in your cell to check compliance with `pycodestyle`

```
%%pycodestyle
```

12

Install the pep8 extension for ipython notebook using the following command :

```
%install_ext https://raw.githubusercontent.com/SiggyF/notebooks/master/pep8_magic.py
```

Refer the official docs for more info.

After that use the `%%pep8` Cell magic function to check your particular cell for pep8 styling.

Note that this has to be put inside every cell for which pep8 checking needs to be enforced.

Refer this example.

share edit flag

answered Oct 5 '14 at 13:27

Raghav RV
2,558 ● 13 ● 25

6 install_ext has been deprecated and removed. Extensions now need to be published on PyPI and installed with pip. – Matt Feb 7 '17 at 19:43

pep8 is deprecated and extensions now needs to be published on PyPi, see answer stackoverflow.com/a/47204361/2459096 or srcoll down – Mattijn Nov 9 '17 at 14:29

add a comment

5

In case this helps anyone, I'm using:

```
conttest "jupyter nbconvert notebook.ipynb --stdout --to script | flake8 - --ignore=W391"
```

- `conttest` reruns when saving changes to the notebook
- `flake8 -` tells flake8 to take input from stdin
- `--ignore=W391` - this is because the output of `jupyter nbconvert` seems to always have a "blank line at end of file", so I don't want flake8 to complain about that.

I'm having a problem with markdown cells (whose line lengths may legitimately be quite long, though): ignore markdown cells in `jupyter nbconvert` with `--to script`.

share edit flag

edited May 23 '17 at 12:18

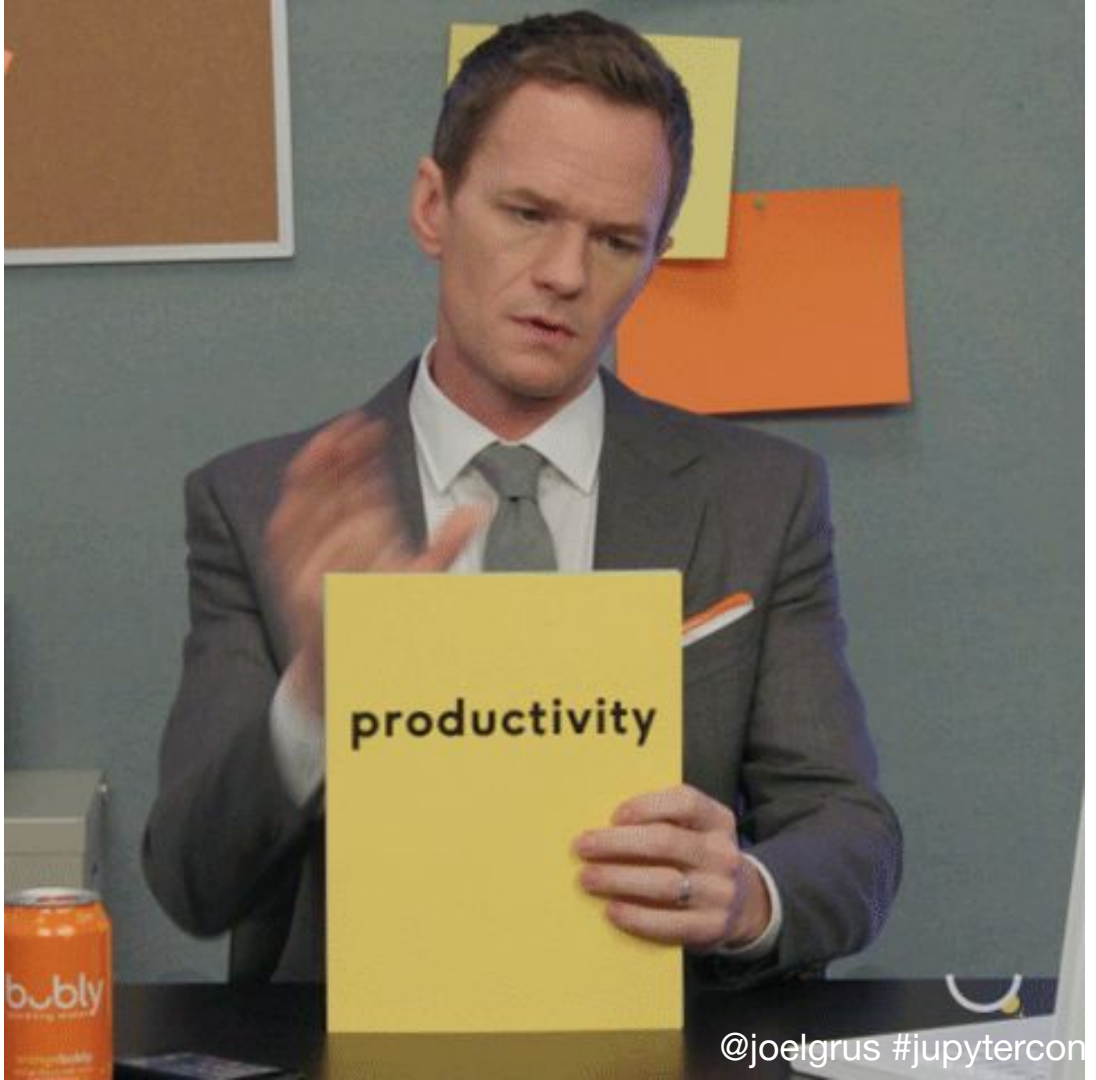
Community ♦
1 ● 1

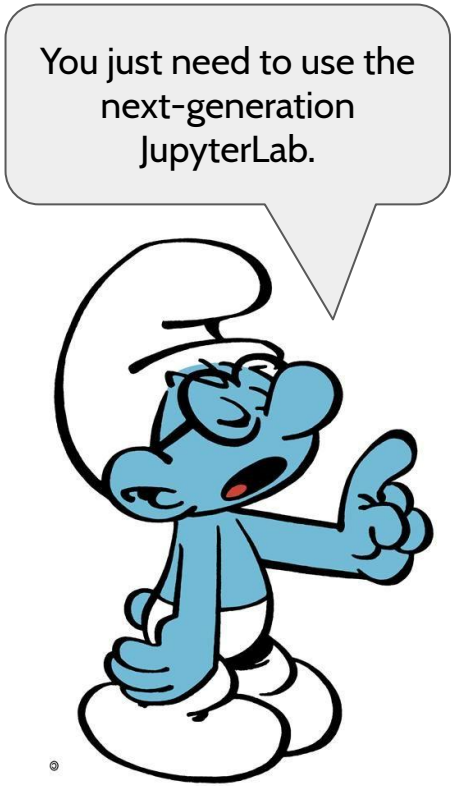
answered May 24 '16 at 20:39

DavidC
674 ● 5 ● 15

add a comment

THOSE ARE THE TOOLS THAT HELP ME BE PRODUCTIVE AND WRITE GOOD CODE





scratch.py

```
1  x = [1, 2, 3]
2  x.ins
```

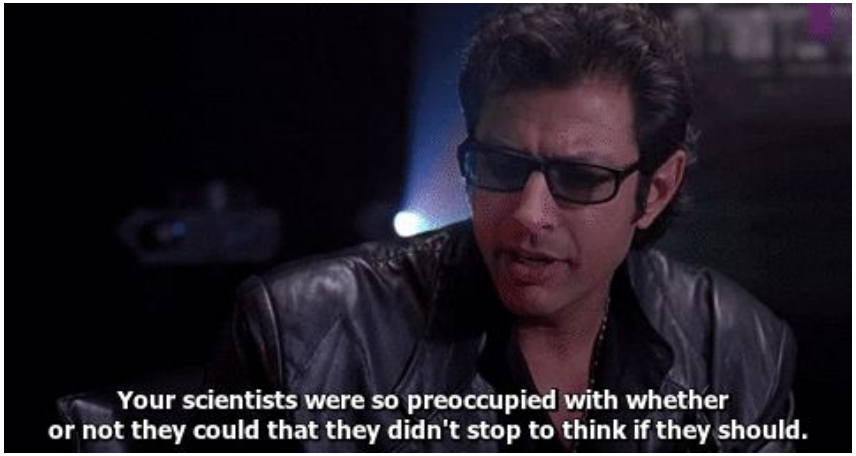
NOTEBOOKS ENCOURAGE BAD PROCESSES

Randy Olson
@randal_olson

Follow

TIL Jupyter Notebooks can now share kernels within Jupyter Lab. Makes it easy to share state between notebooks. #Python #SciPy2018

github.com/jupyterlab/sci ...



Notebook A.ipy ●

Code Python 3

Notebook A

In [1]: a=3

In [ ]:

Notebook B.ipy ●

Code Python 3

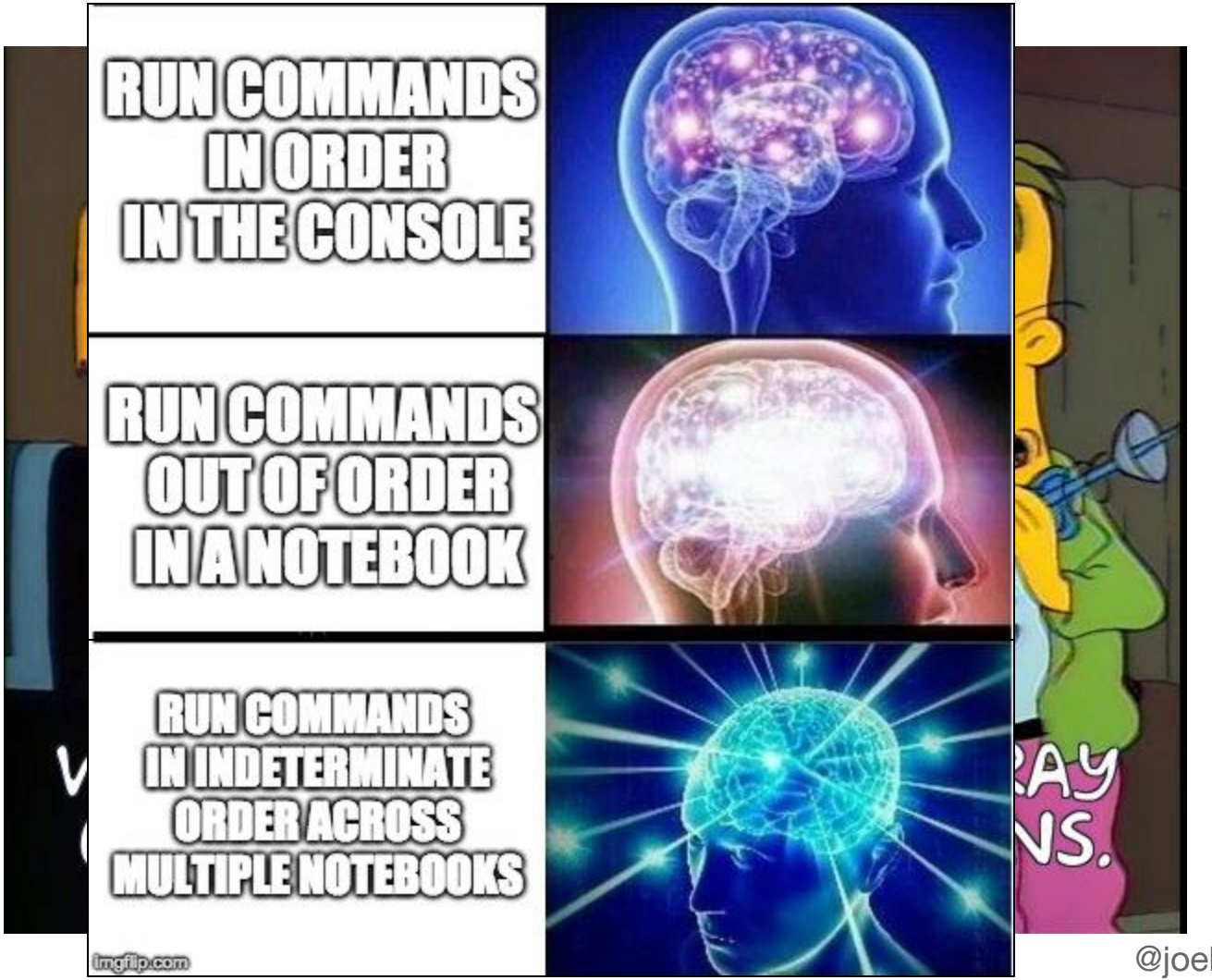
Notebook B

In [2]: a

Out[2]: 3

In [ ]:

@joelgrus #jupytercon


RUN COMMANDS IN ORDER IN THE CONSOLE
RUN COMMANDS OUT OF ORDER IN A NOTEBOOK
RUN COMMANDS IN INDETERMINATE ORDER ACROSS MULTIPLE NOTEBOOKS
imgflip.com

b[redacted] 11:35 AM

@joelgrus [Jupyter Notebook is Best Notebook] so when I have a huge dataset in pyspark I'm used to saving time by using the shared state in my notebook

How do I handle this outside of a notebook?

joelgrus [Jupyter Notebook is Best Notebook] 11:37 AM

you can't save the processed data ?

that seems like a better solution than having it hang around it in-memory in a python process

👆 1

NOTEBOOKS HINDER REPRODUCIBLE + EXTENSIBLE SCIENCE


NO, I'M SERIOUS
The Fly ©2017FOX

NOTEBOOKS ARE A RECIPE FOR POORLY FACTORED CODE

François Chollet
@fchollet

Following

Buggy code is bad science. Poorly tuned benchmarks are bad science. Poorly factored code is bad science (hinders reproducibility, increases chances of a mistake). If your field is all about empirical validation, then your code *is* a large part of your scientific output.

12:26 AM - 15 Jul 2018

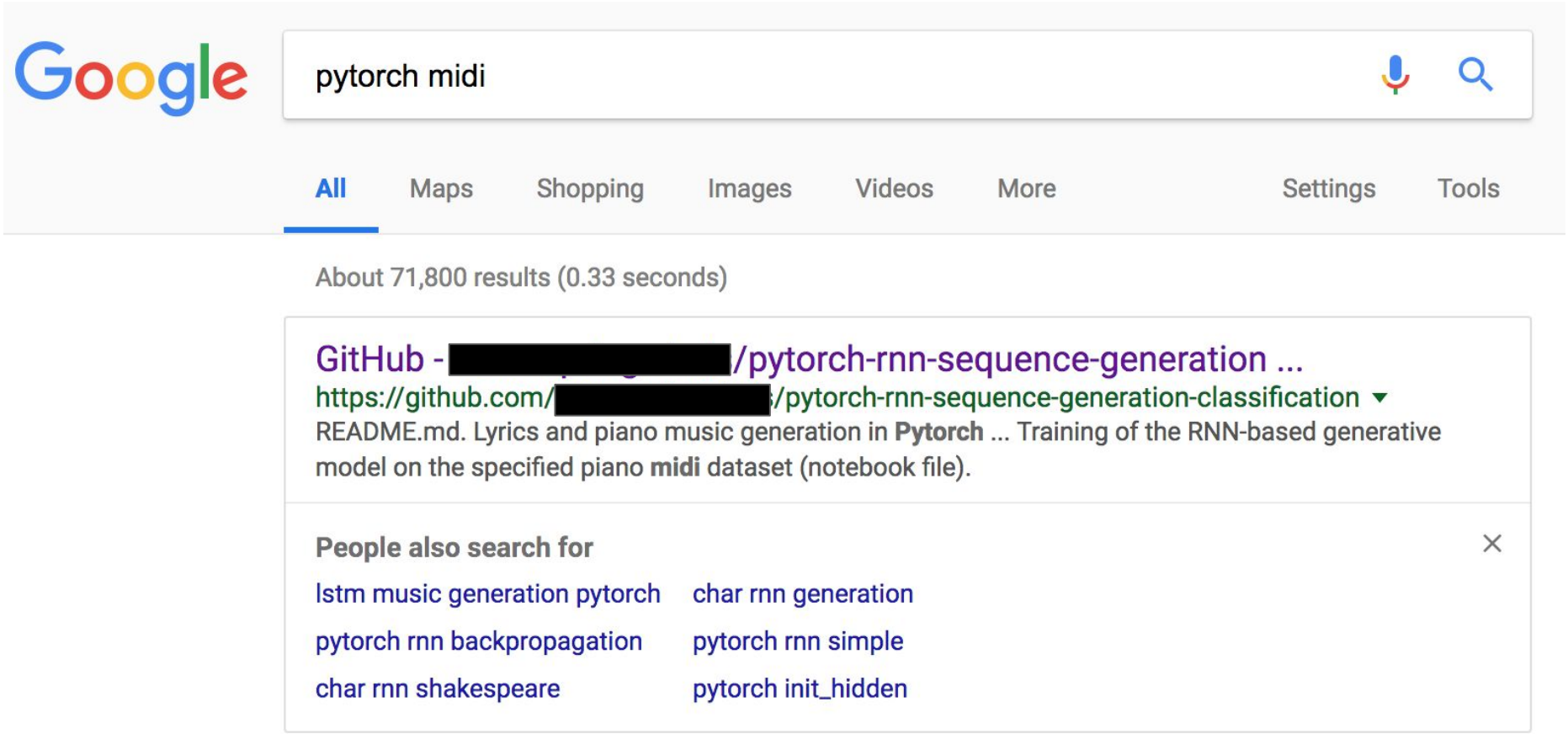
12 Retweets **66** Likes

3 12 66

IDEA:
TRAIN A NEURAL NETWORK TO GENERATE MUSIC

this is the true story of an idea that my co-workers and I were kicking around at lunch a couple of months ago, there was more to the idea but this was at the core

"STANDING ON THE SHOULDERS OF GIANTS, ETC., ETC."



[redacted] / pytorch-rnn-sequence-generation-classification

Watch 5 | Star 32 | Fork 8

Code | Issues 1 | Pull requests 0 | Projects 0 | Wiki | Insights

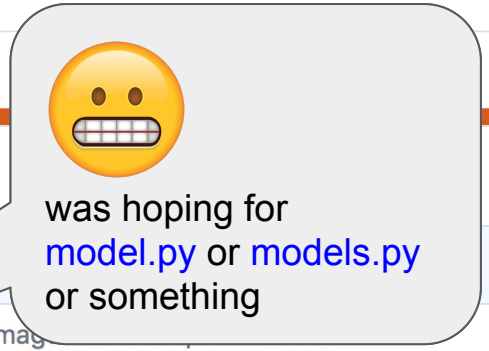
Lyrics and piano music generation in Pytorch http://[redacted]

42 commits | 0 releases | 1 contributor

Branch: master | New pull request | Create new file | Upload files | Find file | Clone or download

[redacted] typo correcte — Latest commit d511eb9 on Jan 27

imgs	moved imag	5 months ago
notebooks	.sf2 details were added	5 months ago
.gitignore	updated gitignore with respect to the folder structure change	5 months ago
README.md	typo corrected	5 months ago
blog_post.ipynb	added sources of the figures that were used	5 months ago
blog_post.md	converted jupyter to markdown -- just testing	5 months ago



NOPE

Piano polyphonic midi song generation

We are providing jupyter notebooks for training and sampling from generative RNN-based models trained on a piano midi songs dataset. Separate notebooks are avalable for:

1. Training of the RNN-based generative model on the specified piano midi dataset (notebook file).
2. Sampling from a trained RNN-based generative model (notebook file).

Branch: master ▾ | Find file | Copy path

pytorch-rnn-sequence-generation-classification / notebooks / music_generation_training_nottingham.ipynb

[redacted] added the link to download midi library — 11417b5 on Jan 27

1 contributor

1246 lines (1245 sloc) | 104 KB | Raw | Blame | History

```
In [1]: import pretty_midi
        import numpy as np
        import torch
        import torch.nn as nn
        from torch.autograd import Variable
        import torch.utils.data as data
        import os
        import random
```

pytorch 0.3? 0.4? 0.4.1?

I suppose I could guess based on the commit date

AN ASIDE

jupyter notebook managing requirements

All News Videos Shopping Images More Settings Tools

About 519,000 results (0.41 seconds)

python - Managing requirements in IPython Notebook? - Stack Overflow
https://stackoverflow.com/questions/.../managing-requirements-in-ipython-notebook
1 answer
Nov 3, 2015 - You can do this by importing numpy and then checking numpy.__version__ . Related question here.

Install **IPython notebook** on a remote server without root ... Apr 26, 2017
python - **jupyter notebook** running kernel in different env Jun 18, 2016
python - Where to put imports in an **iPython notebook**? Nov 3, 2015
More results from stackoverflow.com

Managing requirements in IPython Notebook?

0

I am a Python programmer, new to IPython Notebook. I have started a notebook that uses `numpy`.

If I was publishing the same code as a standalone Python script, I'd include a `requirements.txt` file with `numpy` and its version, so that other users can install the same version into a virtualenv when they run the script.

What is the equivalent of this for iPython Notebook? I can't find anything about managing requirements in the documentation, only about the dependencies required to install IPython itself.

My requirements are that I'd like to make sure that the notebook is using a particular version of `numpy`, and I want to make sure that I can publish the notebook with the same version specified.

python ipython-notebook

share edit flag

asked Nov 3 '15 at 4:41
Richard
17.1k ● 51 ● 175 ● 312

There's no automatic dependency management for notebooks. – cel Nov 3 '15 at 4:47

That's surprising! How do users safely reproduce results with them? – Richard Nov 3 '15 at 5:02

Good libraries are usually written to be backwards compatible. So ideally any version should do. Of course we are not living in an ideal world, so sometimes things break. Of course you can distribute a `requirements.txt` along with your notebook, but there's no automated way to include this information in your notebook. – cel Nov 3 '15 at 5:08

@cel thanks for the answer. Trust me, there are a lot of cases where any version won't do! I'll try running the notebook in a virtualenv and see if it picks up the requirements from the virtualenv. – Richard Nov 3 '15 at 5:40

```
        logits_flatten = binary_logits.view(-1, 2)

        return logits_flatten, hidden
```

In [4]:
```
rnn = RNN(input_size=88, hidden_size=512, num_classes=88)
rnn = rnn.cuda()

criterion = nn.CrossEntropyLoss().cuda()

criterion_val = nn.CrossEntropyLoss(size_average=False).cuda()

learning_rate = 0.001
optimizer = torch.optim.Adam(rnn.parameters(), lr=learning_rate)
```

In [6]:
```
%matplotlib notebook

import sys, os
sys.path.append("/home/daniil/repos/pytorch-segmentation-detection/")
sys.path.insert(0, '/home/daniil/repos/pytorch-segmentation-detection/vision/')
```

model definition

model instantiation (with hard-coded parameters)

more parameters

hard-coded paths

```
    return full_val_loss / (overall_sequence_length * 88)
```

In [8]:
```
clip = 1.0
epochs_number = 12000000
sample_history = []
best_val_loss = float("inf")

for epoch_number in xrange(epochs_number):

    for batch in trainset_loader:

        post_processed_batch_tuple = post_process_sequence_batch(batch)

        input_sequences_batch, output_sequences_batch, sequences_lengths = post_processed_batch_tu
ple

        output_sequences_batch_var = Variable( output_sequences_batch.contiguous().v
```

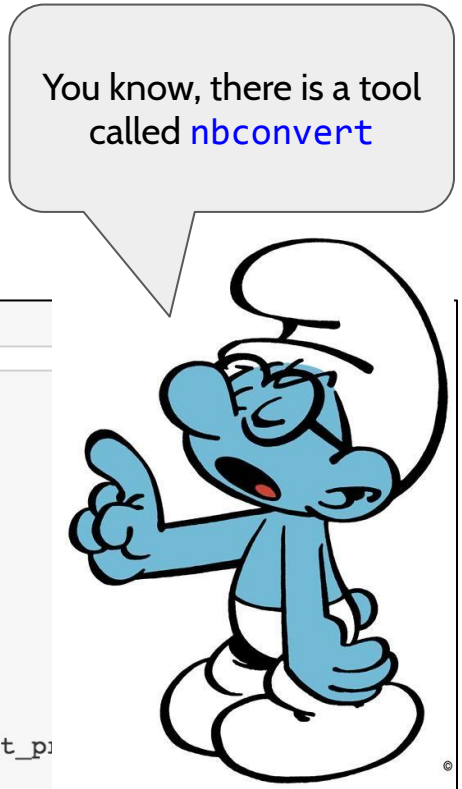
more parameters

training loop

```
        logits_flatten = binary_logits.view(-1, 2)

        return logits_flatten, hidden
```

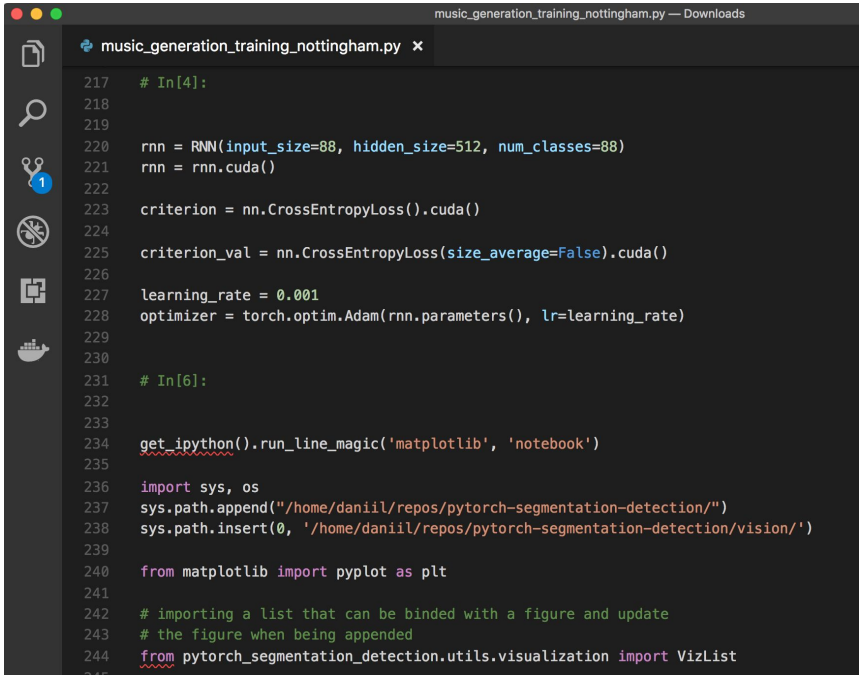
```
In [4]: rnn = RNN(input_size=88, hidden_size=512, num_classes=88)
```

- This code is pretty much **impossible** for me to use
- First, I have to look at the notebooks, manually install all the dependencies, and hope I got the right version
- Then either:
  - I can run it the *exact* same way on the *exact* same data (at the exact same paths!), or
  - I can copy the notebook and try to figure out how to modify it (forcing me to work in a notebook), or
  - I can do a lot of work to copy and paste the right parts into a module that I can import into my own code



(YES, I AM AWARE OF NBCONVERT)

```
$ jupyter nbconvert music_generation_training_nottingham.ipynb --to python
```



- still mixes "library" code and "execution" code
- still have to go through and perform surgery
- still no requirements.txt / unit tests
- still few comments :(

TO BE PERFECTLY FAIR

- this is someone's fun project
- **kudos to them for sharing what they did**
- **likely they thought they were being helpful by providing notebooks**
- what they did was make it semi-possible (but not easy) for me to repeat their exact same code (but only if I have my data at the exact same paths and the right versions of all their dependencies, ¯_(ツ)_/¯)
- while making it very hard for me to build new things on top of their code
- again, lots of non-notebook code has these same issues, it's just that notebooks implicitly *encourage* a workflow that has these issues

BUT IMAGINE...

BUT IMAGINE...

- a `requirements.txt` (possibly even a Docker image) indicating exactly what dependencies I need
- clean, parameterizable `load_data.py` and `model.py` with good documentation and unit tests (but I repeat myself)
- an easy way to programmatically specify model parameters, data paths, and so on
- a `run_experiment.py` script that replicates exactly the original result, and that can be tweaked in obvious ways to run novel variants

allenai / bilm-tf

Watch 21 | Star 290 | Fork 76

Code | Issues 3 | Pull requests 1 | Projects 0 | Wiki | Insights | Settings

Tensorflow implementation of contextualized word representations from bi-directional language models

Edit

Add topics

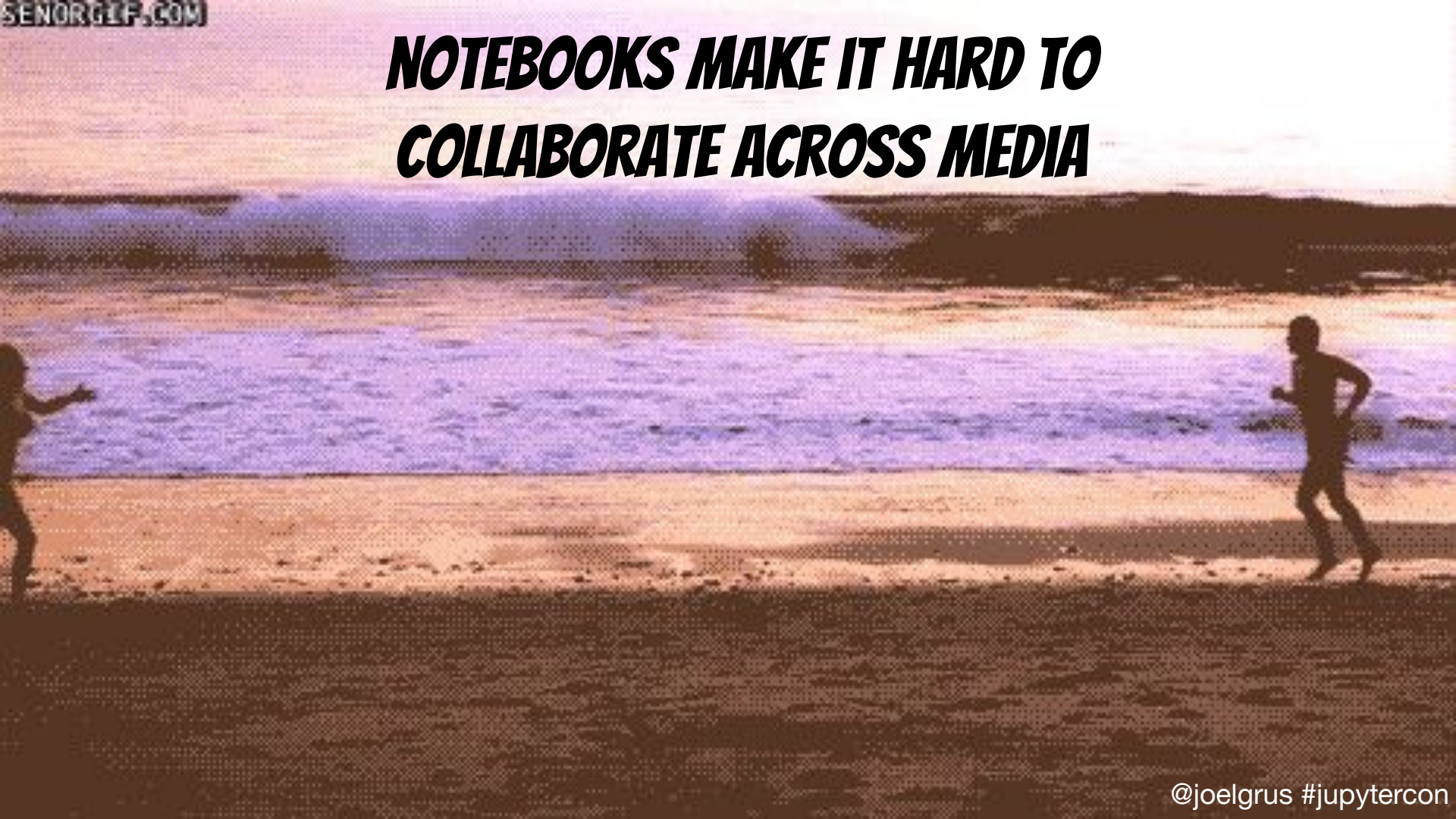
Installing

2. Train the biLM.

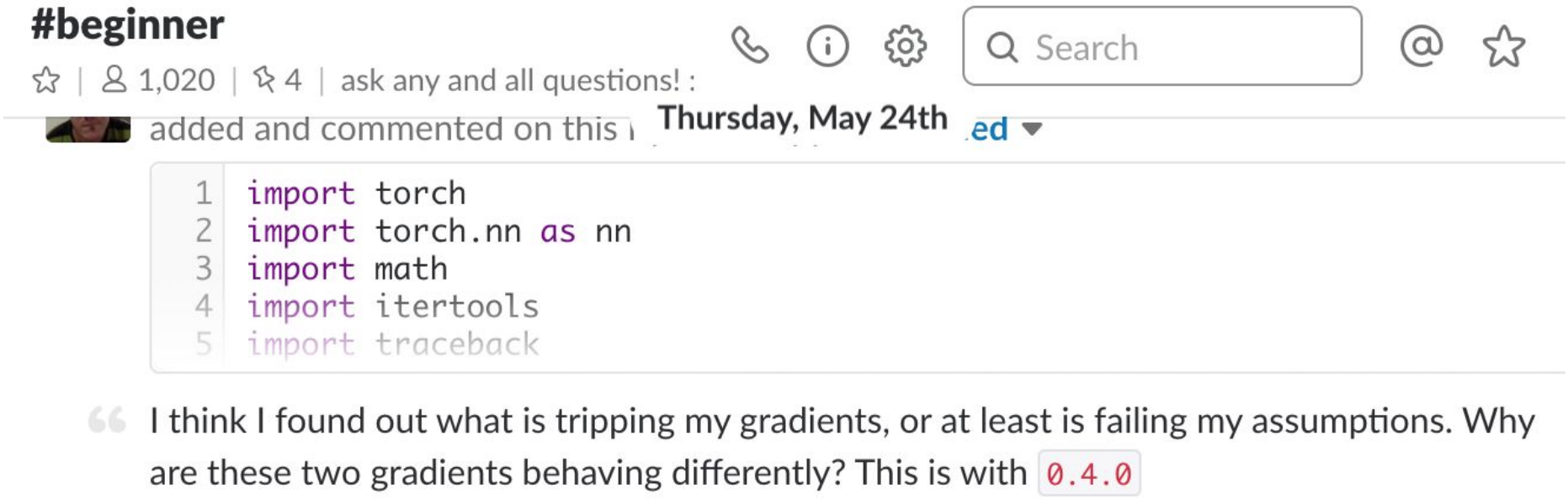
The hyperparameters used to train the ELMo model can be found in `bin/train_elmo.py`.

The ELMo model was trained on 3 GPUs. To train a new model with the same hyperparameters, first download the training data from the 1 Billion Word Benchmark. Then download the vocabulary file. Finally, run:

```
export CUDA_VISIBLE_DEVICES=0,1,2
python bin/train_elmo.py \
    --train_prefix='/path/to/1-billion-word-language-modeling-benchmark-r13output/training-monolingual.toke
    --vocab_file /path/to/vocab-2016-09-10.txt \
    --save_dir /output_path/to/checkpoint
```


SENORGIF.COM
NOTEBOOKS MAKE IT HARD TO COLLABORATE ACROSS MEDIA
@joelgrus #jupytercon

THE SCENE: PYTORCH SLACK, #BEGINNER CHANNEL



"IT NEVER HURTS TO HELP!"

> it looks like a bug to me, but I'm not sure if missing something important

Joel Grus 7:26 AM

`torch.tensor` copies data and breaks the computation graph

Joel Grus 7:26 AM

here's a simpler version

Joel Grus 7:26 AM

here's a simpler version

1 reply 1 month ago

Joel Grus 7:26 AM

```
In [1]: import torch

In [2]: x = torch.tensor([1, 2, 3], requires_grad=True)

In [3]: y = torch.tensor(x).sum()

In [4]: y.backward()
RuntimeError: element 0 of tensors does not require grad and does not have a grad_fn

In [5]: z = x.sum()

In [6]: z.backward()

In [7]: x.grad
Out[7]: tensor([ 1,  1,  1])
```

In [1]:
```
import torch
```

In [2]:
```
x = torch.tensor([1, 2, 3], requires_grad=True)
```

In [3]:
```
y = torch.tensor(x).sum()
```

In [4]:
```
y.backward()
```

```
---------------------------------------------------------------------------
RuntimeError                              Traceback (most recent call last)
<ipython-input-4-ab75bb780f4c> in <module>()
----> 1 y.backward()

~/anaconda3/envs/allennlp/lib/python3.6/site-packages/torch/tensor.py in backward(self, gradient, retain_graph, cre
ate_graph)
     91                 products. Defaults to ``False``.
     92         """
---> 93         torch.autograd.backward(self, gradient, retain_graph, create_graph)
     94
     95     def register_hook(self, hook):

~/anaconda3/envs/allennlp/lib/python3.6/site-packages/torch/autograd/__init__.py in backward(tensors, grad_tensors,
 retain_graph, create_graph, grad_variables)
     87     Variable._execution_engine.run_backward(
     88         tensors, grad_tensors, retain_graph, create_graph,
---> 89         allow_unreachable=True)  # allow_unreachable flag
     90
     91
```

In [5]:
```
z = x.sum()
```

In [6]:
```
z.backward()
```

In [7]:
```
x.grad
```

Out[7]:
```
tensor([ 1,  1,  1])
```

OK, I GUESS I COULD DO THAT IN A NOTEBOOK

BUT HOW TO COPY AND PASTE INTO SLACK?

SELECT ALL -> COPY -> PASTE

Joel Grus 11:11 AM

Jupyter Notebook
Untitled
Last Checkpoint: 12 minutes ago
(autosaved)
Current Kernel Logo
Python 3
File
Edit
View
Insert
Cell
Kernel
Widgets
Help

ch
import torch

```
y.backward()

---------------------------------------------------------------
RuntimeError                              Traceback (most rece
<ipython-input-4-ab75bb780f4c> in <module>()
----> 1 y.backward()

~/anaconda3/envs/allennlp/lib/python3.6/site-packages/torch/te
retain_graph, create_graph)
     91                 products. Defaults to ``False``.
     92         """
---> 93         torch.autograd.backward(self, gradient, retair
     94
     95     def register_hook(self, hook):

~/anaconda3/envs/allennlp/lib/python3.6/site-packages/torch/au
grad_tensors, retain_graph, create_graph, grad_variables)
     87     Variable._execution_engine.run_backward(
     88         tensors, grad_tensors, retain_graph, create_gr
---> 89         allow_unreachable=True)  # allow_unreachable f
     90
     91

RuntimeError: element 0 of tensors does not require grad and d


z = x.sum()


)
z.backward()

x.grad
tensor([ 1,  1,  1])
```

SELECT CELLS -> EDIT -> COPY CELLS -> PASTE

doesn't seem to do anything
outside of the notebook

that is, when you go to Slack,
there's nothing in the clipboard

SELECT CELLS -> COPY -> PASTE

Joel Grus 11:16 AM

```
import torch

x = torch.tensor([1, 2, 3], requires_grad=True)


y = torch.tensor(x).sum()


y.backward()


z = x.sum()


z.backward()

x.grad
```

BASICALLY, I HAVE NO IDEA HOW TO COPY AND PASTE CODE + OUTPUTS FROM A NOTEBOOK INTO SLACK

as far as I can tell, in JupyterLab even the "ugly" versions don't work!

THIS MAY SEEM FRIVOLOUS, BUT I SPEND A LOT OF TIME DEBUGGING PEOPLE'S PYTHON ISSUES VIA SLACK

pyhocon does not handle backslashes the same way json does #171

Edit | New issue

Open **joelgrus** opened this issue 19 days ago · 1 comment

joelgrus commented 19 days ago • edited ▾ Contributor

```
In [1]: import json, pyhocon

In [2]: single = r'{"a": "b\.c"}'

In [3]: double = r'{"a": "b\\.c"}'

In [4]: json.loads(single)
--------------------------------------------------------------------------------
JSONDecodeError...

In [5]: json.loads(double)
Out[5]: {'a': 'b\\.c'}

In [6]: pyhocon.ConfigFactory.parse_string(single)
Out[6]: ConfigTree([('a', 'b\\.c')])

In [7]: pyhocon.ConfigFactory.parse_string(double)
Out[7]: ConfigTree([('a', 'b\\\\.c')])

In [8]: json.dumps(json.loads(double))
Out[8]: '{"a": "b\\\\.c"}'

In [9]: json.dumps(json.loads(double)) == double
Out[9]: True

In [10]: json.dumps(pyhocon.ConfigFactory.parse_string(double))
Out[10]: '{"a": "b\\\\\\\\.c"}'
```

Assignees

No one assigned

Labels

None yet

Projects

None yet

Milestone

No milestone

Notifications

Unsubscribe

You're receiving notifications because you authored the thread.

2 participants

SAME THING FOR GITHUB ISSUES / CODE REVIEWS

One other nuisance that I discovered when working on this example!

KERNEL -> RESTART AND RUN ALL

```
In [1]: import torch

In [2]: x = torch.tensor([1, 2, 3], requires_grad=True)

In [3]: y = torch.tensor(x).sum()

In [4]: y.backward()
```

```
----> 1 y.backward()

~/anaconda3/envs/allennlp/lib/python3.6/site-packages/torch/tensor.py in backward(self, gradient, retain_graph, cre
ate_graph)
     91                 products. Defaults to ``False``.
     92         """
---> 93         torch.autograd.backward(self, gradient, retain_graph, create_graph)
     94
     95     def register_hook(self, hook):

~/anaconda3/envs/allennlp/lib/python3.6/site-packages/torch/autograd/__init__.py in backward(tensors, grad_tensors,
 retain_graph, create_graph, grad_variables)
     87     Variable._execution_engine.run_backward(
     88         tensors, grad_tensors, retain_graph, create_graph,
---> 89         allow_unreachable=True)  # allow_unreachable flag
     90
     91

RuntimeError: element 0 of tensors does not require grad and does not have a grad_fn
```

```
In [ ]: z = x.sum()

In [ ]: z.backward()

In [ ]: x.grad
```

error was expected

(and is part of what I want to demonstrate)

but halts execution

run all cells stop

Closed ruoyu0088 opene

ruoyu0088 commer

Currently "run all" co
by using notebook, s

I want to known are t

juhasch commented on Apr 6, 2016 Contributor

The `runtools` extension from https://github.com/ipython-contrib/IPython-notebook-extensions supports this.

The actual code is shown below with the relevant part being `cell.execute(false)'

```
var execute_all_cells_ignore_errors = function () {
    for (var i=0; i < IPython.notebook.ncells(); i++) {
        IPython.notebook.select(i);
        var cell = IPython.notebook.get_selected_cell();
        cell.execute(false);
    }
};
```

Installation

To install the `jupyter_contrib_nbextensions` notebook extensions, three steps are required. First, the Python pip package needs to be installed. Then, the notebook extensions themselves need to be copied to the Jupyter data directory. Finally, the installed notebook extensions can be enabled, either by using built-in Jupyter commands, or more conveniently by using the jupyter_nbextensions_configurator server extension, which is installed as a dependency of this repo.

run all cells stor

Closed ruoy

ruoyu0

Curren
by usir

I want

jubasch commented on Apr 6, 2016 Contributor

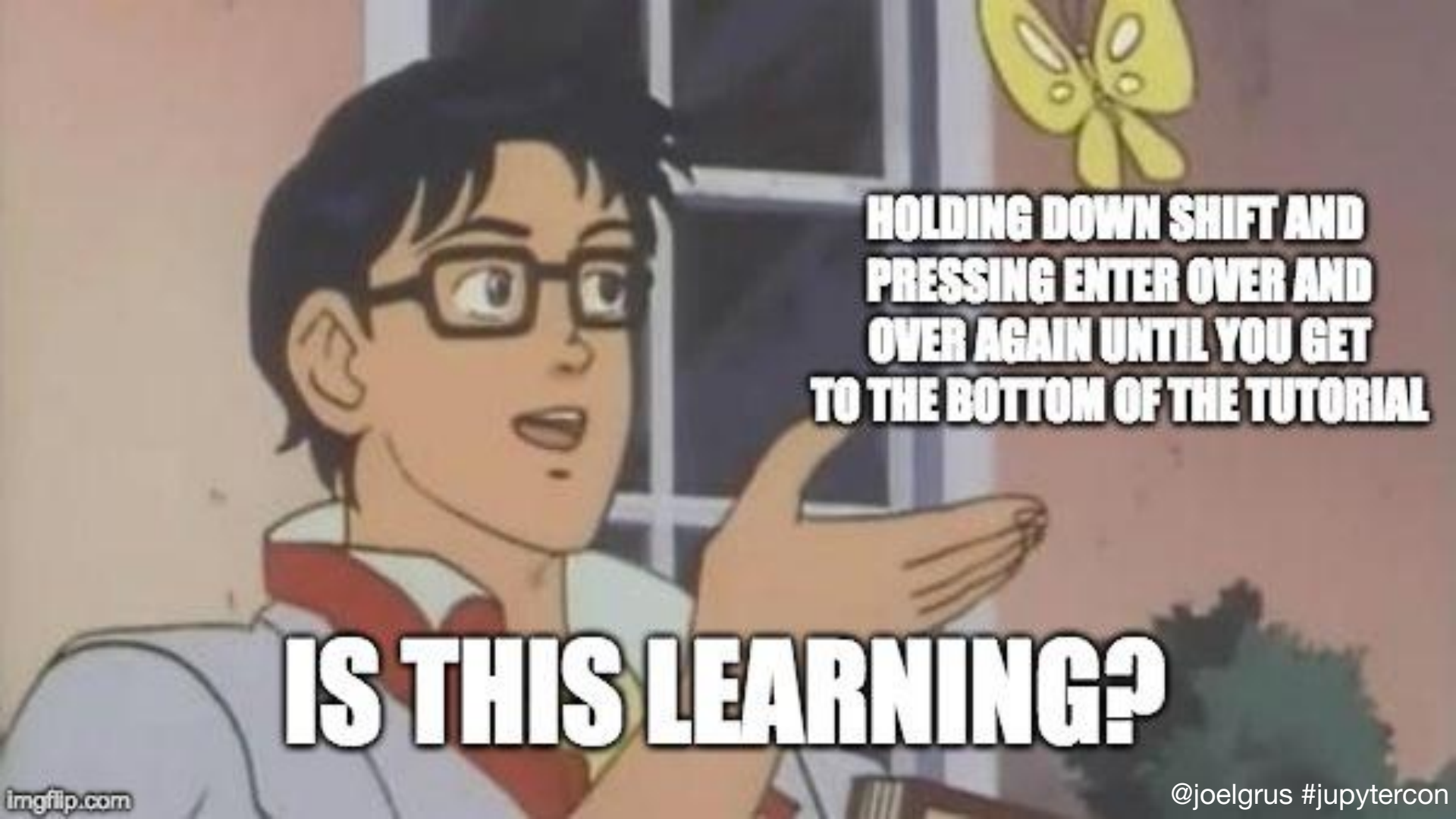
on-notebook-extensions

ute(false)'

Installat

To install the `jupyter_contrib_nbextensions` notebook extensions, three steps are required. First, the Python pip package needs to be installed. Then, the notebook extensions themselves need to be copied to the Jupyter data directory. Finally, the installed notebook extensions can be enabled, either by using built-in Jupyter commands, or more conveniently by using the jupyter_nbextensions_configurator server extension, which is installed as a dependency of this repo.

NOTEBOOKS MAKE IT EASY TO TEACH POORLY


HOLDING DOWN SHIFT AND
PRESSING ENTER OVER AND
OVER AGAIN UNTIL YOU GET
TO THE BOTTOM OF THE TUTORIAL
IS THIS LEARNING?
imgflip.com

GitHub, Inc. [US] | https://github.com/jupyter/jupyter/wiki/A-gallery-of-interesting-Jupyter-Notebooks#introductory-tutorials

Introductory Tutorials

- First things first, how to run code in the notebook. There is also a general collection of notebooks from IPython. Another useful one from this collection is an explanation of our rich display system.
- A great matplotlib tutorial, part of the fantastic Lectures on Scientific Computing with Python by J.R. Johansson.
- The code of the IPython mini-book by C. Rossant, introducing IPython, NumPy, SciPy, Pandas and matplotlib for interactive computing and data visualization.
- Python Tutorial by [redacted]

While

while some_condition:

```
    algorithm
```

In [8]:
```
i = 1
while i < 3:
    print(i ** 2)
    i = i+1
print('Bye')
```

```
1
4
Bye
```

Break

As the name says. It is used to break out of a loop when a condition becomes true when executing the loop.

In [9]:
```
for i in range(100):
    print i
    if i>=7:
        break
```

```
0
1
2
3
4
5
6
7
```

While

while some_condition:

    algorithm

In [8]:
```
i = 1
while i < 3:
    print(i ** 2)
    i = i+1
print('Bye')
```

```
1
4
Bye
```

Break

As the name says. It is used to

In [9]:
```
for i in range(100):
    print i
    if i>=7:
        break
```

```
0
1
2
3
4
5
6
7
```

Control Flow Statements

If

if some_condition:

    algorithm

In [1]:
```
x = 12
if x >10:
    print "Hello"
```

```
Hello
```

If-else

if some_condition:

    algorithm

else:

    algorithm

In [2]:
```
x = 12
if x > 10:
    print "hello"
else:
    print "world"
```

```
hello
```

jupyter / jupyter

Watch

Code | Issues 100 | Pull requests 1 | Projects 0 | Wiki | Insights

A gallery of interesting Jupyter Notebooks

Rugantio Costa edited this page 28 days ago · 66 revisions

This page is a curated collection of Jupyter/IPython notebooks that are notable. Feel free to add new content here, but please try to only include links to notebooks that include interesting visual or technical content; this should *not* simply be a dump of a Google search on every ipynb file out there.

This page is a curated collection of Jupyter/IPython notebooks that are notable.

Table of Contents

NOTEBOOKS MAKE IT HARD FOR ME TO TEACH WELL

this is from a real book

written by me

Data Science from Scratch

available wherever books are sold

2nd edition being written as we speak

OK, not *exactly* as we speak.

te this table with:

```
(
T NULL,
00),
T);
```

d that the `user_id` and `num_friends` must be integers (and tha
`user_id` isn't allowed to be `NULL`, which indicates a missing value and is sort of lik
name should be a string of length 200 or less. NotQuiteABas
ccount, but we'll behave as if it did.

almost completely case and indentation insensitive. The
zation and indentation style here is my preferred style. If
you start learning SQL, you will surely encounter other examples
ifferently.

s with INSERT

(user_id,

ing

u'll c

row,

d to be

Behind the scenes, we'll store each ro to values. A rea
database would never use such a space-wasting rep , but doing so wil
make NotQuiteABase much easier to work with:

```
class Table:
    def __init__(self, columns):
        self.columns = columns
        self.rows = []

    def __repr__(self):
        """pretty representation of the table: columns then rows"""
        return str(self.columns) + "\n" + "\n".join(map(str, self.rows))

    def insert(self, row_values):
        if len(row_values) != len(self.columns):
            raise TypeError("wrong number of elements")
        row_dict = dict(zip(self.columns, row_values))
        self.rows.append(row_dict)
```

For example, we could set up:


```
users = Table(["user_id", "name", "num_friends"])
users.insert([0, "Hero", 0])
users.insert([1, "Dunn", 2])
users.insert([2, "Sue", 3])
users.insert([3, "Chi", 3])
users.insert([4, "Thor", 3])
users.insert([5, "Clive", 2])
users.insert([6, "Hicks", 3])
users.insert([7, "Devin", 2])
users.insert([8, "Kate", 2])
users.insert([9, "Klein", 3])
users.insert([10, "Jen", 1])
```

If you now `print users`, you'll see:

```
['user_id', 'name', 'num_friends']
{'user_id': 0, 'name': 'Hero', 'num_friends': 0}
{'user_id': 1, 'name': 'Dunn', 'num_friends': 2}
{'user_id': 2, 'name': 'Sue', 'num_friends': 3}
...
```

UPDATE

Sometimes you need to update the data that's already in the database. For instance, if Dunn acquires another friend, you might need to do this:

```
UPDATE users
SET num_friends = 3
WHERE user_id = 1;
```

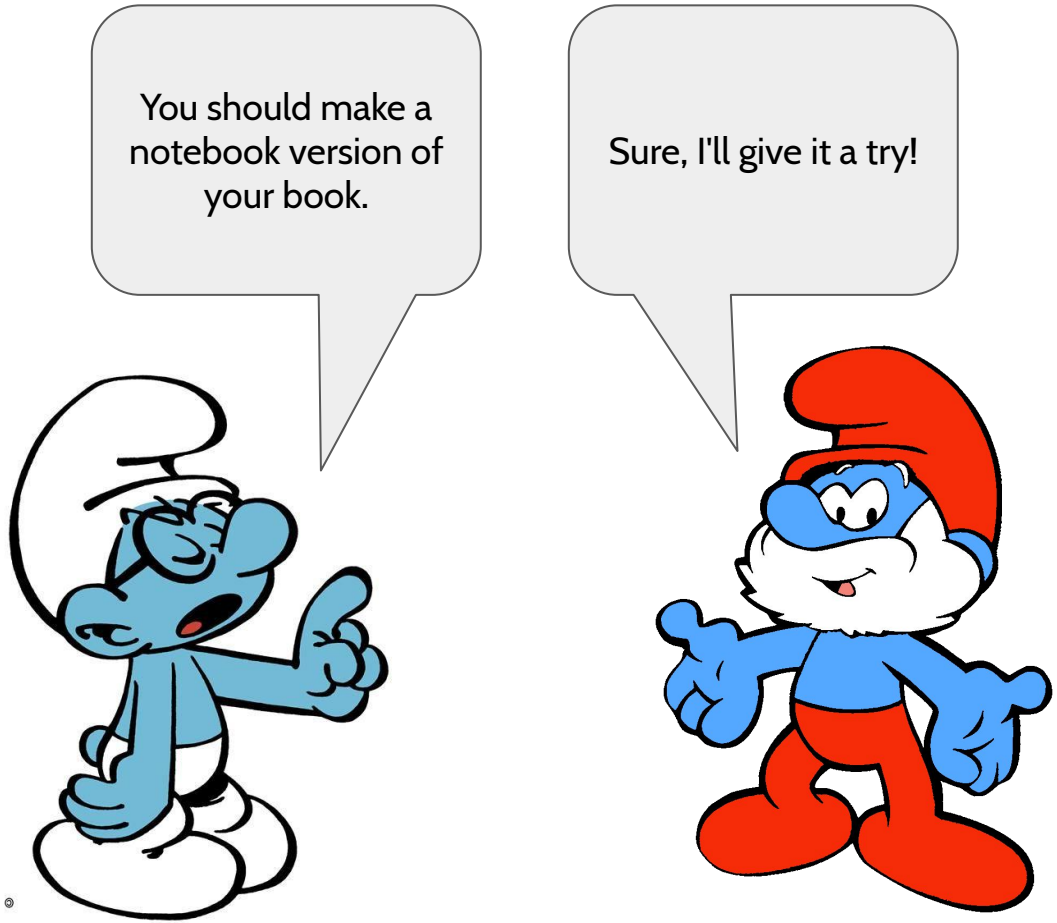
The key features are:

- What table to update
- Which rows to update
- Which fields to update
- What their new values should be

We'll add a similar `update` method to NotQuiteABase. Its first argument will be a `dict` whose keys are the columns to update and whose values are the new values for those fields. And its second argument is a `predicate` that returns `True` for rows that should be updated, `False` otherwise:

```
    def update(self, updates, predicate):
        for row in self.rows:
            if predicate(row):
                for column, new_value in updates.iteritems():
                    row[column] = new_value
```

after which we can simply do this:


You should make a notebook version of your book.
Sure, I'll give it a try!

this is from a real book

written by me

Data Science from Scratch

available wherever books are sold

2nd edition being written as we speak.

OK, not *exact*

```
users = Table(["user_id", "name", "num_friends"])
users.insert([0, "Hero", 0])
users.insert([1, "Dunn", 2])
users.insert([2, "Sue", 3])
users.insert([3, "Chi", 3])
users.insert([4, "Thor", 3])
users.insert([5, "Clive", 2])
users.insert([6, "Hicks", 3])
users.insert([7, "Devin", 2])
users.insert([8, "Kate", 2])
users.insert([9, "Klein", 3])
users.insert([10, "Jen", 1])
```

If you now `print users`, you'll see:

```
['user_id', 'name', 'num_friends']
{'user_id': 0, 'name': 'Hero', 'num_friends': 0}
{'user_id': 1, 'name': 'Dunn', 'num_friends': 2}
{'user_id': 2, 'name': 'Sue', 'num_friends': 3}
...
```

Behind the scenes, we'll store each row as a `dict` from column names to database would never use such a space-wasting representation, but make NotQuiteABase much easier to work with:

```
class Table:
    def __init__(self, columns):
        self.columns = columns
        self.rows = []

    def __repr__(self):
        """pretty representation of the table: columns then rows"""
        return str(self.columns) + "\n" + "\n".join(map(str, self.r

    def insert(self, row_values):
        if len(row_values) != len(self.columns):
            raise TypeError("wrong number of elements")
        row_dict = dict(zip(self.columns, row_values))
        self.rows.append(row_dict)
```

For example, we could set up:

We'll add a similar `update` method to NotQuiteABase. Its first argument will be a `dict` whose keys are the columns to update and whose values are the new values for those fields. And its second argument is a `predicate` that returns `True` for rows that should be updated, `False` otherwise:

```
def update(self, updates, predicate):
    for row in self.rows:
        if predicate(row):
            for column, new_value in updates.iteritems():
                row[column] = new_value
```

after which we can simply do this:

```
UPDATE users
SET num_friends = 3
WHERE user_id = 1;
```

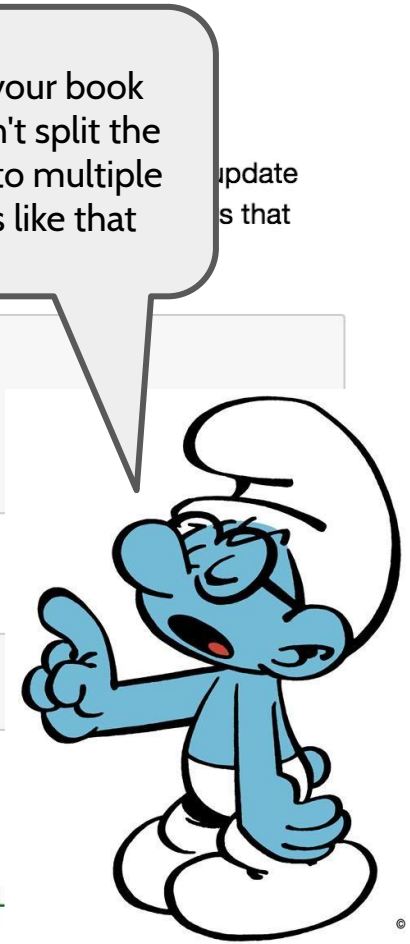
We'll add a similar update method to NotQuiteABase. Its first argument will be a `dict` w…pdate and whose values are the new values for those fields. And its second argument is a prec…s that should be updated, `False` otherwise:

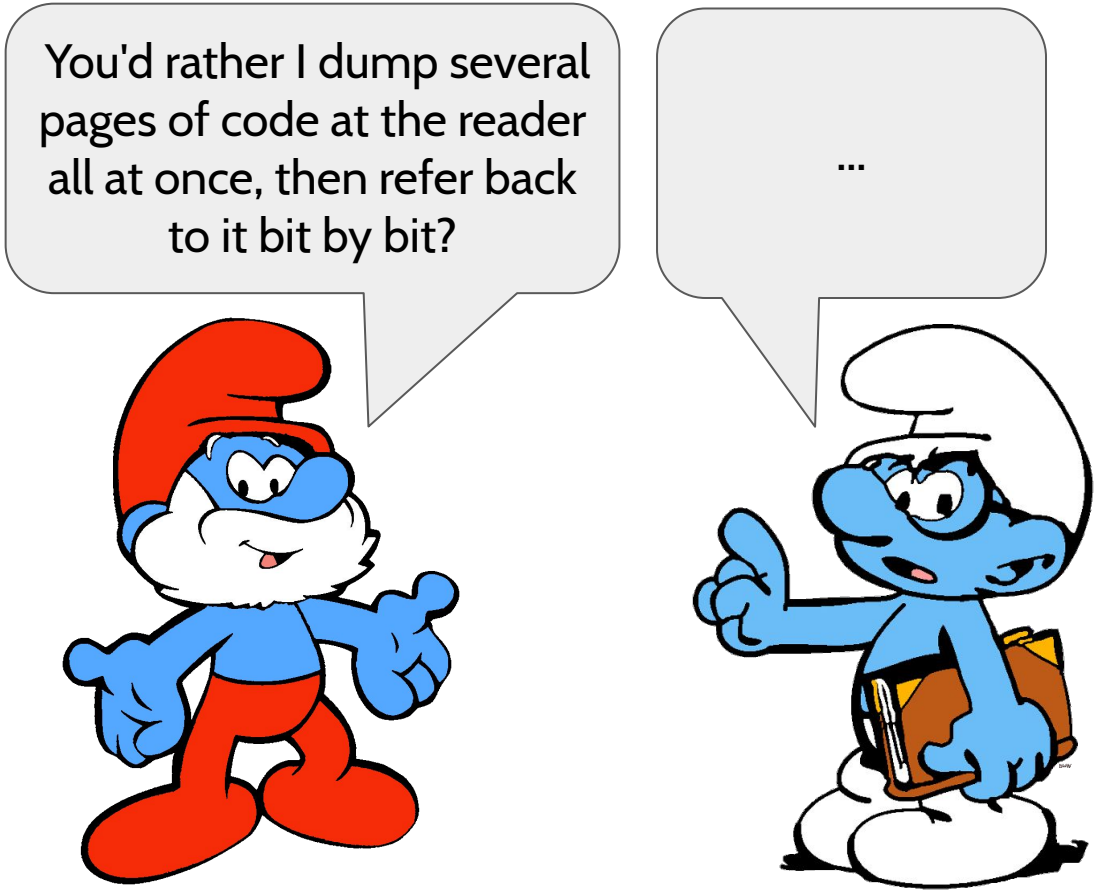
In [4]:
```
def update(self, updates, predicate):
    for row in self.rows:
        if predicate(row):
            for column, new_value in updates.iteritems():
                row[column] = new_value
```

after which we can do this:

In [5]:
```
users.update({'num_friends' : 3},              # set num_friends = 3
             lambda row: row['user_id'] == 1) # in rows where user_id == 1
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-5-1491612e18fc> in <module>()
----> 1 users.update({'num_friends' : 3},              # set num_friends = 3
      2              lambda row: row['user_id'] == 1) # in rows where user_i

AttributeError: 'Table' object has no attribute 'update'
```




You'd rather I dump several pages of code at the reader all at once, then refer back to it bit by bit?
...

jupyter / notebook

Watch 240

Code | Issues 920 | Pull requests 35 | Projects 9 | Insights

Define a Python class across multiple cells #1243

Closed **Carreau** opened this issue on Mar 22, 2016 · 17 comments

Carreau commented on Mar 22, 2016 — Member

This issue is both Notebook & IPython, but I guess could be solved here only with an extension.

Again, requested during JupyterDays:

For education purpose, it would be nice to be able to define a class across many cells., in particular 1 method/cell.

I think it is possible by peeking at next/previous coed cells, and if the all body of a cell is indented by at least 4 spaces, "join" the content of each cell before sending a single execution request.

we could "mark" joined executed cells with a specific prompt.

Ping **@ellisonbg**

👍 2

JDC 0.0.1 Alex Hagen Page

jdc - Jupyter Dynamic Classes

Alex Hagen

ExampleÂ¶

Below is an example on how to use `jdc`. First we have to import the class:

In [1]:
```
import jdc
```

Then, we have to make a dummy class:

In [2]:
```
class dog(object):
    def __init__(self, name, noise):
        self.name = name
```

%%add_to dog

```
def bark2(self, times=1):
    saying = ('%s ' % self.noise) * times
    print "%s says: %s" % (self.name, saying)
```

dsblank commented on Mar 1 Contributor

Just to point out that you can do this in regular Python, without resorting to any extra magic:

```
class MyClass(MyClass):
```

Cell 2:

```
class MyClass(MyClass):
    def method2(self):
        print("method2")
```

Cell 3:

```
instance = MyClass()
instance.method1()
instance.method2()
```

That is, you can define a class recursively, cell by cell. I don't think the final class is any different from one defined all in one cell. So you don't need an extra package to "solve" this problem.

👍 3

THIS WAS ALSO SUGGESTED

In [9]:
```
def update(self, updates, predicate):
    for row in self.rows:
        if predicate(row):
            for column, new_value in updates.iteritems():
                row[column] = new_value

Table.update = update
```

In [10]:
```
users.update({'num_friends' : 3},                # set num_friends = 3
             lambda row: row['user_id'] == 1) # in rows where user_id == 1
```

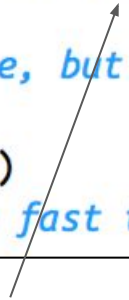
```
Table.update = update
```

EVERY ONE OF THESE REQUIRES PRESENTING UNNATURAL CODE TO MY READERS, SOLELY TO APPEASE THE NOTEBOOK GODS

SIMPLER EXAMPLES TOO

We'll use sets for two main reasons. The first is that `in` is a very fast operation on sets. If we have a large collection of items that we want to use for a membership test, a set is more appropriate than a list:

```
stopwords_list = ["a","an","at"] + hundreds_of_other_words + ["yet", "you"]

"zip" in stopwords_list     # False, but have to check every element

stopwords_set = set(stopwords_list)
"zip" in stopwords_set      # very fast to check
```



this is not a real variable, which means this snippet isn't executable

this example would not be improved by actually defining that variable

WHAT I DO INSTEAD

In addition, our CRF will accept an optional set of *constraints* that disallow "invalid" transitions (where "invalid" depends on what you're trying to model.) For example, our NER data has distinct tags that represent the beginning, middle, and end of each entity type. We'd like not to allow a "beginning of a person entity" tag to be followed by an "end of location entity tag".

As the CRF is just a component of our model, we'll implement it as a Module.

Implementing the CRF Module

To implement a PyTorch module, we just need to inherit from `torch.nn.Module` and override

```
    def forward(self, *input):
        ...
```

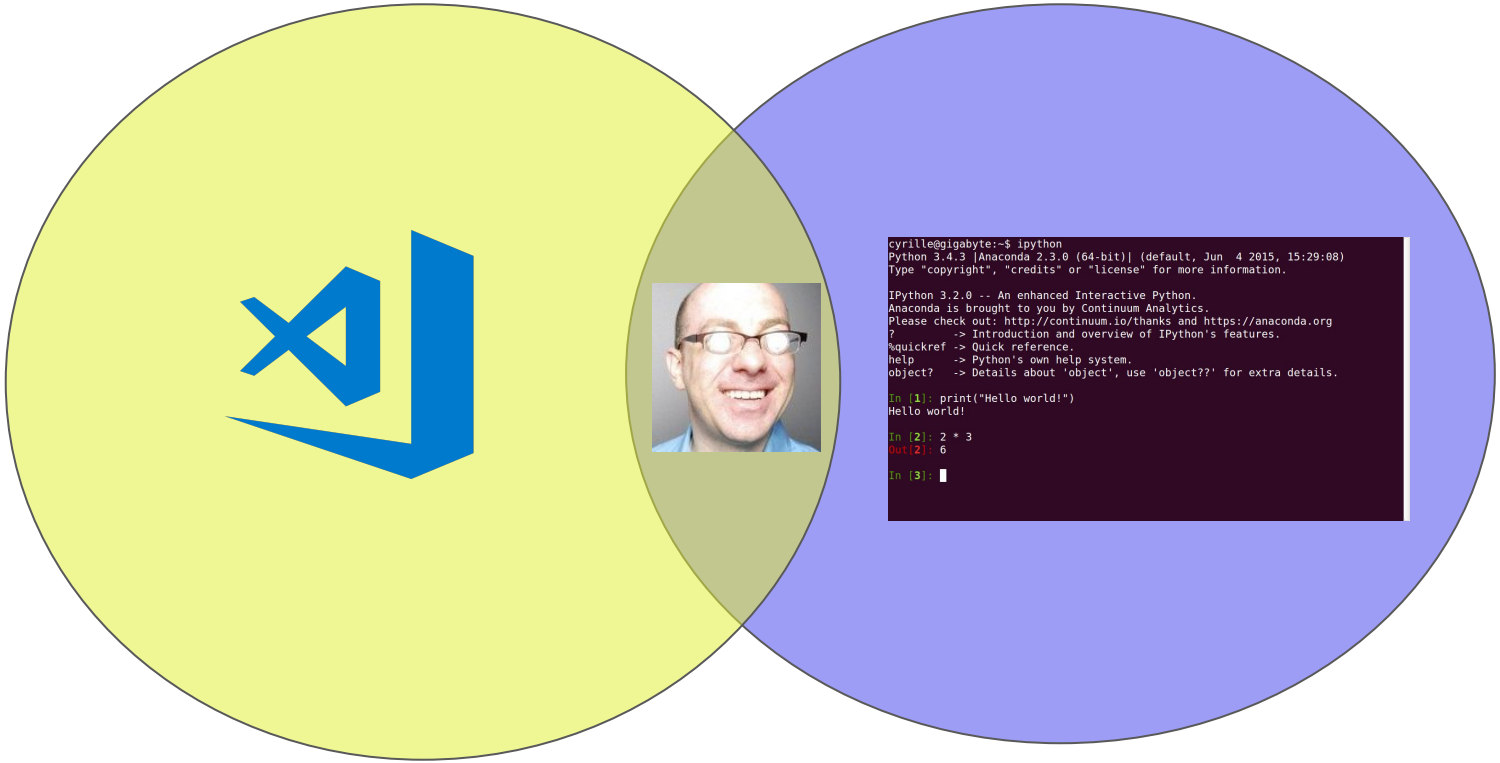
to compute the log-likelihood of the provided inputs.

To initialize this module, we just need the number of tags and optionally some constraints (represented as a set of allowed pairs `(from_tag_index, to_tag_index)`):

```
    def __init__(self,
                 num_tags: int,
                 constraints: List[Tuple[int, int]] = None) -> None:
        super().__init__()
        self.num_tags = num_tags
```

MARKDOWN TUTORIALS AND DOCS

VSCODE + IPYTHON CONSOLE



DEMO
(TIME PERMITTING)

HOW YOU COULD WIN ME OVER

Jake VanderPlas @jakevdp · 27 Nov 2017

Idea: Jupyter notebooks could have a "reproducibility mode" where:

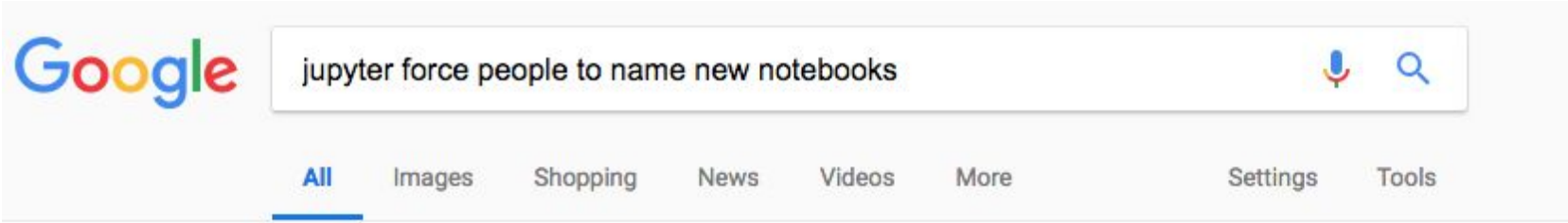
1) Code cells are read-only once executed
2) New code cells cannot be inserted above previously executed cells
3) No cell can be executed until all previous cells are executed

49 131 599

Joel Grus @joelgrus · 27 Nov 2017

Yeah, something like this would alleviate a lot of my complaints, BUT I suspect most people who love notebooks would hate this workflow

2 4


Google
jupyter force people to name new notebooks
All
Images
Shopping
News
Videos
More
Settings
Tools
About 42,300 results (0.40 seconds)
The Jupyter Notebook — Jupyter Notebook 5.5.0 documentation
jupyter-notebook.readthedocs.io/en/stable/notebook.html
You can create new notebooks from the dashboard with the New Notebook button, or open existing ones by clicking on their name. You can also drag and drop ...
Security in the Jupyter notebook server — Jupyter Notebook 5.5.0 ...
jupyter-notebook.readthedocs.io/en/stable/security.html
Since access to the Jupyter notebook server means access to running arbitrary ... New in version 5.0: jupyter notebook password command is added. ... the potential for malicious people to attempt to exploit the notebook for their nefarious ...
Missing: name | Must include: name
Regarding naming a specific name for jupyter notebook and different ...
https://github.com/jupyter/notebook/issues/3506
Apr 5, 2018 - Hello, In the terminal, if i type jupyter notebook, it will connect to a ... if i want to open a new notebook, then it will be automatically named as ...
Missing: force people


write_json.py - vsexample
write_json.py
1 import json
2
3 def write_json(filename, data):
4 with open(filename) as f:
5
IDE-STYLE AUTOCOMPLETE
0 0
Ln 5, Col 9
Spaces: 4
UTF-8
LF
Python

Real-time typechecking and linting

A Better Story Around Dependency Management


gifbin.com
FIRST-CLASS SUPPORT FOR REFACTORING CODE OUT OF NOTEBOOKS INTO MODULES
@joelgrus #jupytercon

THE REALITY IS

- you're not going to provide me all these things
- I'm not going to switch to notebooks
- but hopefully I've challenged you to think about
  - how your tools limit you
  - how to write better software
  - how to teach better
  - how to practice better science
  - how different workflows could make your life easier
  - some ways that you could make notebooks better

IN CONCLUSION

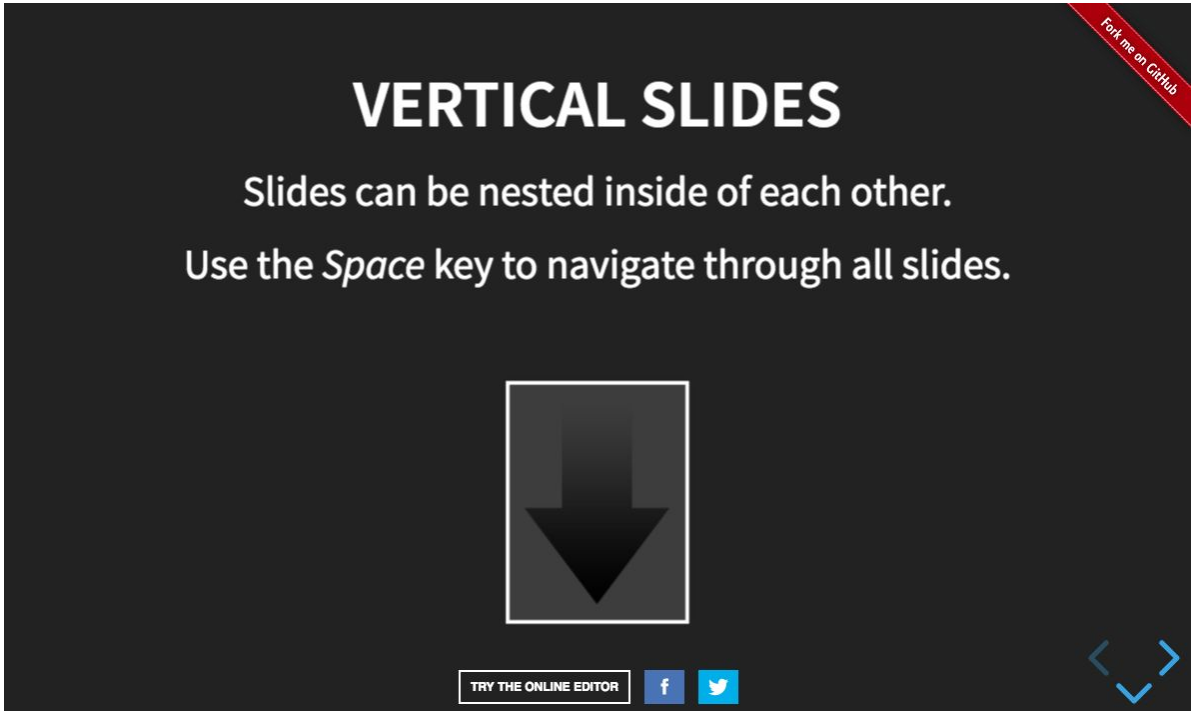
Jeremy Howard @jeremyphoward · 27 Nov 2017

I think it means @vboykis is right.

Convincing people not to use a tool which has repeatedly made people more effective seems like an unhelpful way to spend time, frankly

1 6

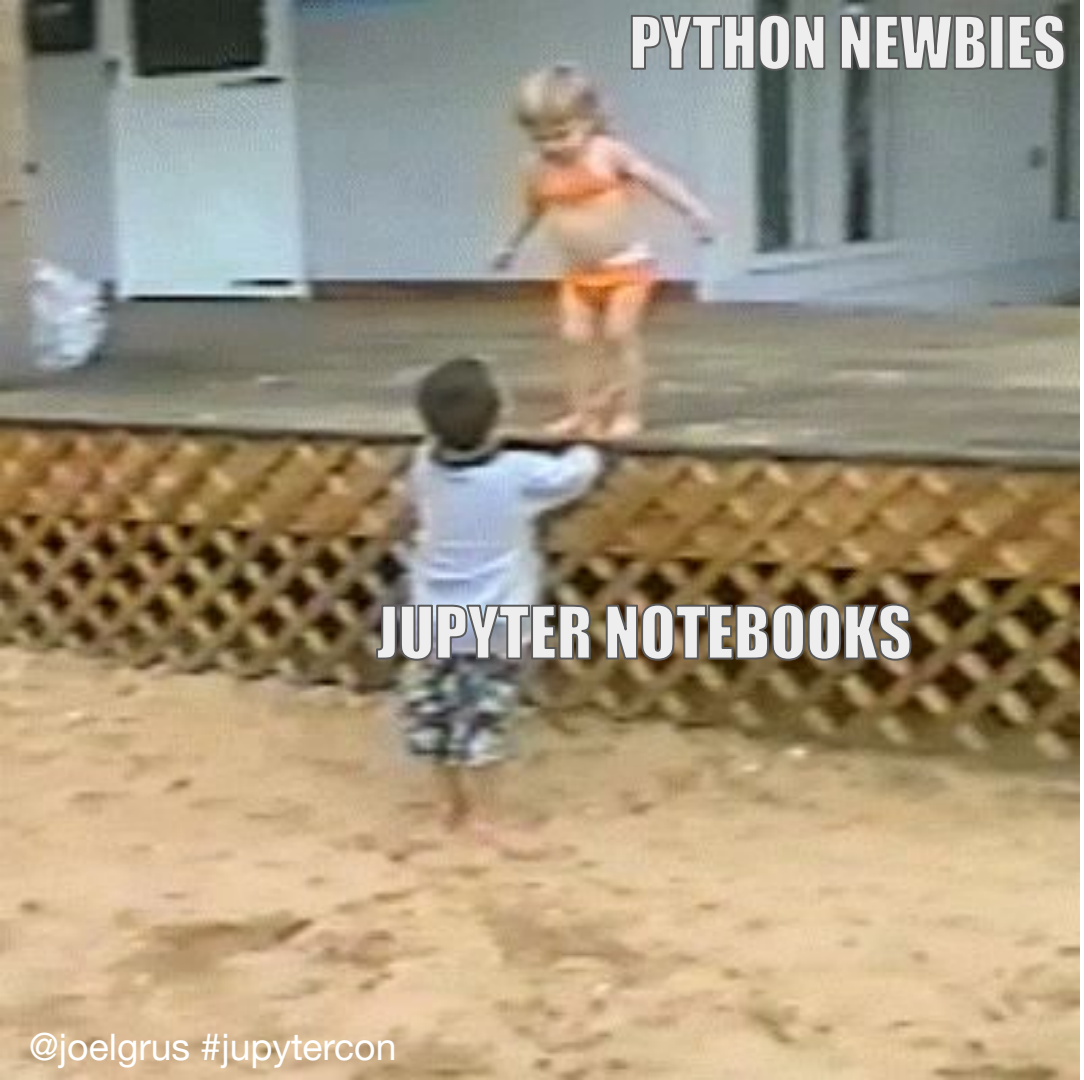
PLEASE COME TO MY NEXT TALK:
"I DON'T LIKE THAT SLIDESHOW PROGRAM WHERE SOMETIMES THE NEXT SLIDE IS RIGHT AND SOMETIMES THE NEXT SLIDE IS DOWN"



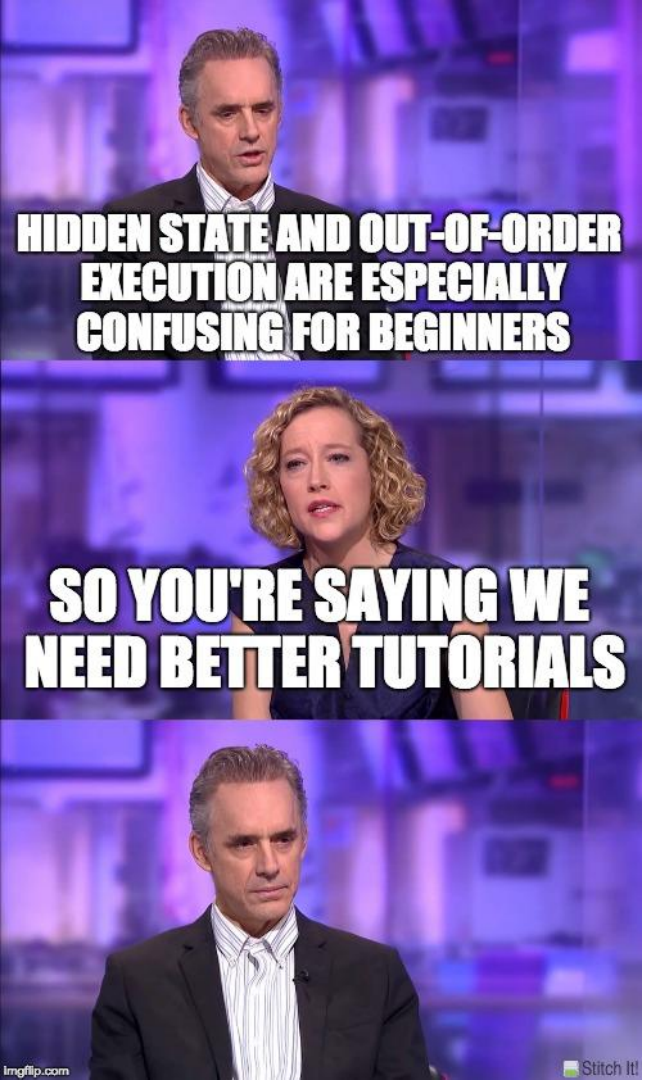
THANKS FOR COMING!

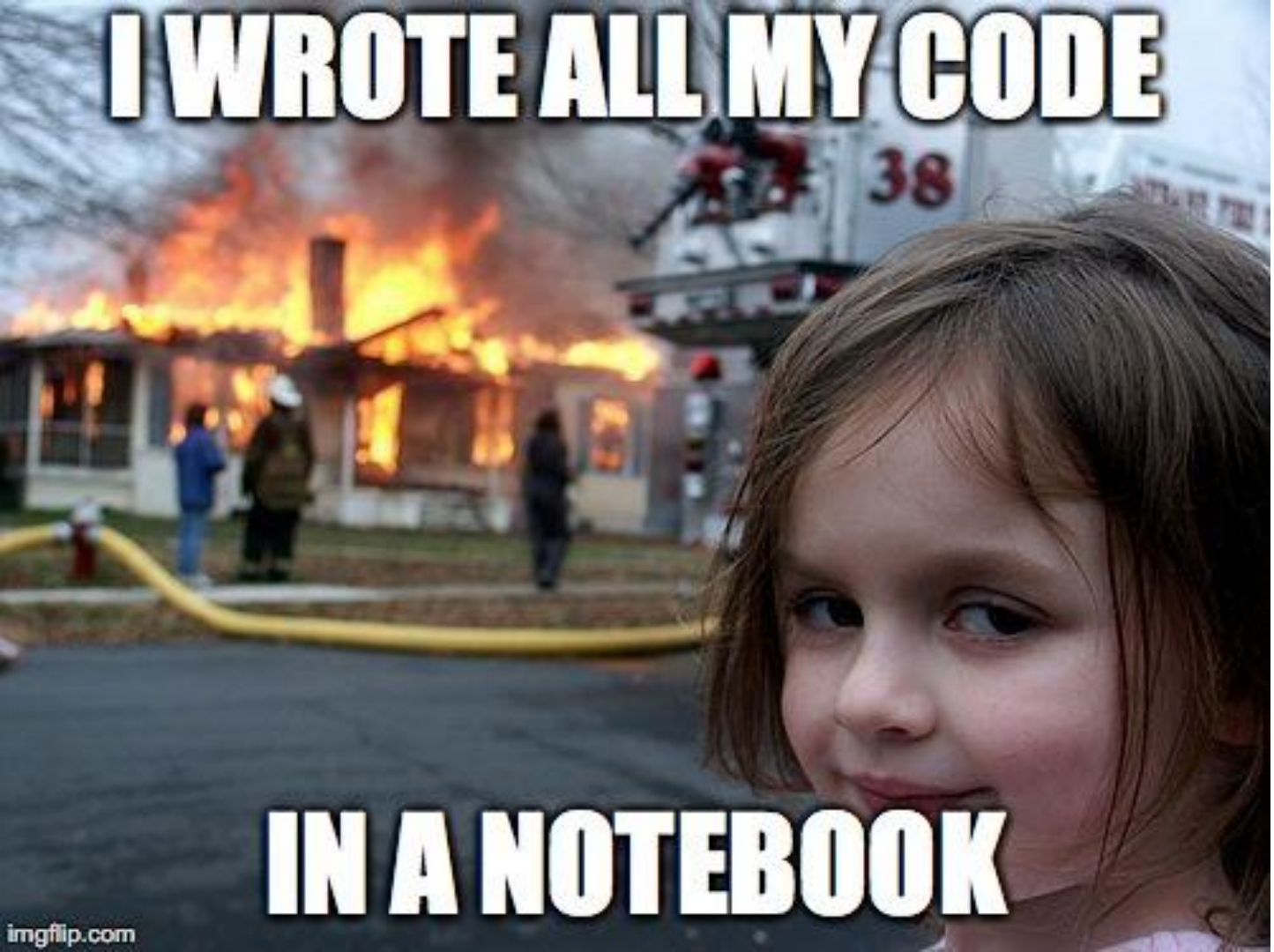
- will tweet out slides: @joelgrus
- check out my book *Data Science from Scratch*
- very infrequently updated blog: joelgrus.com
- podcast: adversariallearning.com
- livecoding videos: youtube.com/user/joelgrus
- tell me why I'm wrong: joelgrus **@** gmail
- and of course I have a SoundCloud 😂😂😂: soundcloud.com/joel-grus

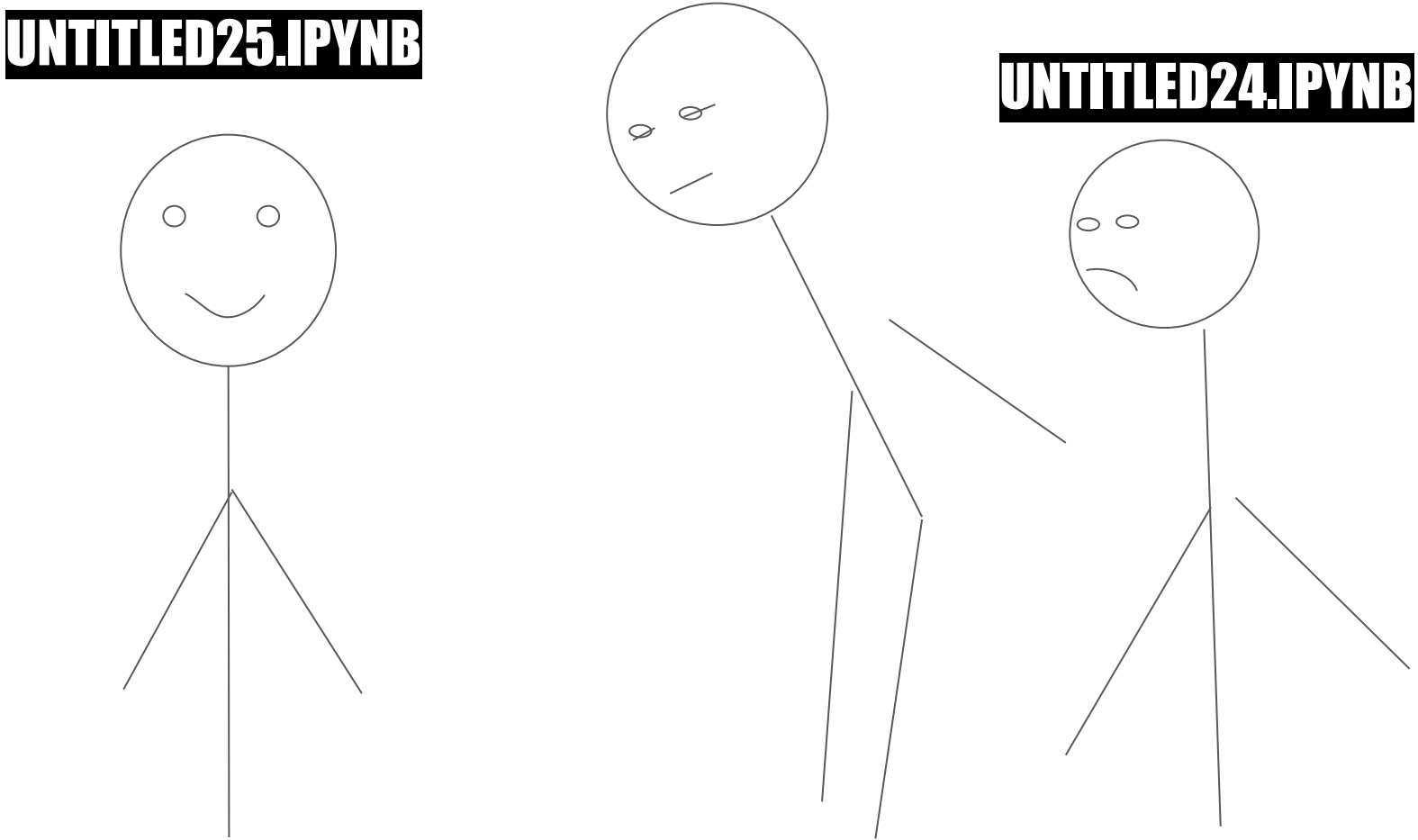
Appendix (Rejected Memes)



@joelgrus #jupytercon


HIDDEN STATE AND OUT-OF-ORDER EXECUTION ARE ESPECIALLY CONFUSING FOR BEGINNERS
SO YOU'RE SAYING WE NEED BETTER TUTORIALS
imgflip.com
Stitch It!


I WROTE ALL MY CODE
IN A NOTEBOOK
imgflip.com


UNTITLED25.IPYNB
UNTITLED24.IPYNB