SIMPLIFICATION AS A SERVICE

- Vasu Sharma

ABOUT ME

- Frustrated Python Data Scientist

ABOUT THIS TALK

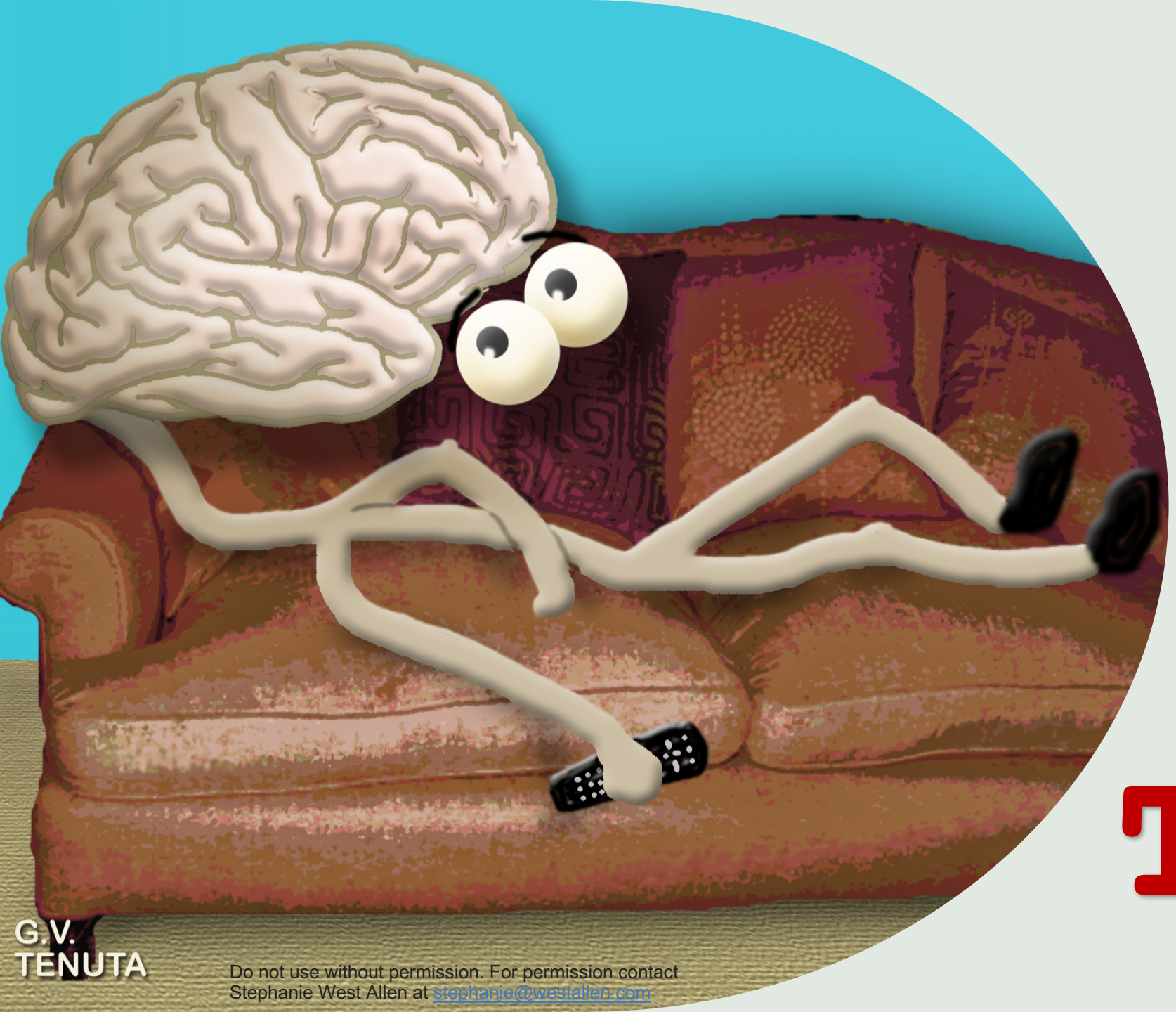
- Mostly a rant
- Part “common-sense” advice
- Suitable audience(s):

1. Software developers
2. Data Scientist(s)/Engineer(s)
3. Anyone who reads/writes code that‘s not meant for just personal consumption

...

?
"Daddy, what did you bring that book that I don't want to be read to out of, up for?"
The Quilt Story

?
"Daddy, what did you bring that book that I don't want to be read to out of, up for?"
The Quilt Story



DON'T MAKE ME THINK!

G.V. TENUTA

Do not use without permission. For permission contact Stephanie West Allen at stephanie@westallen.com

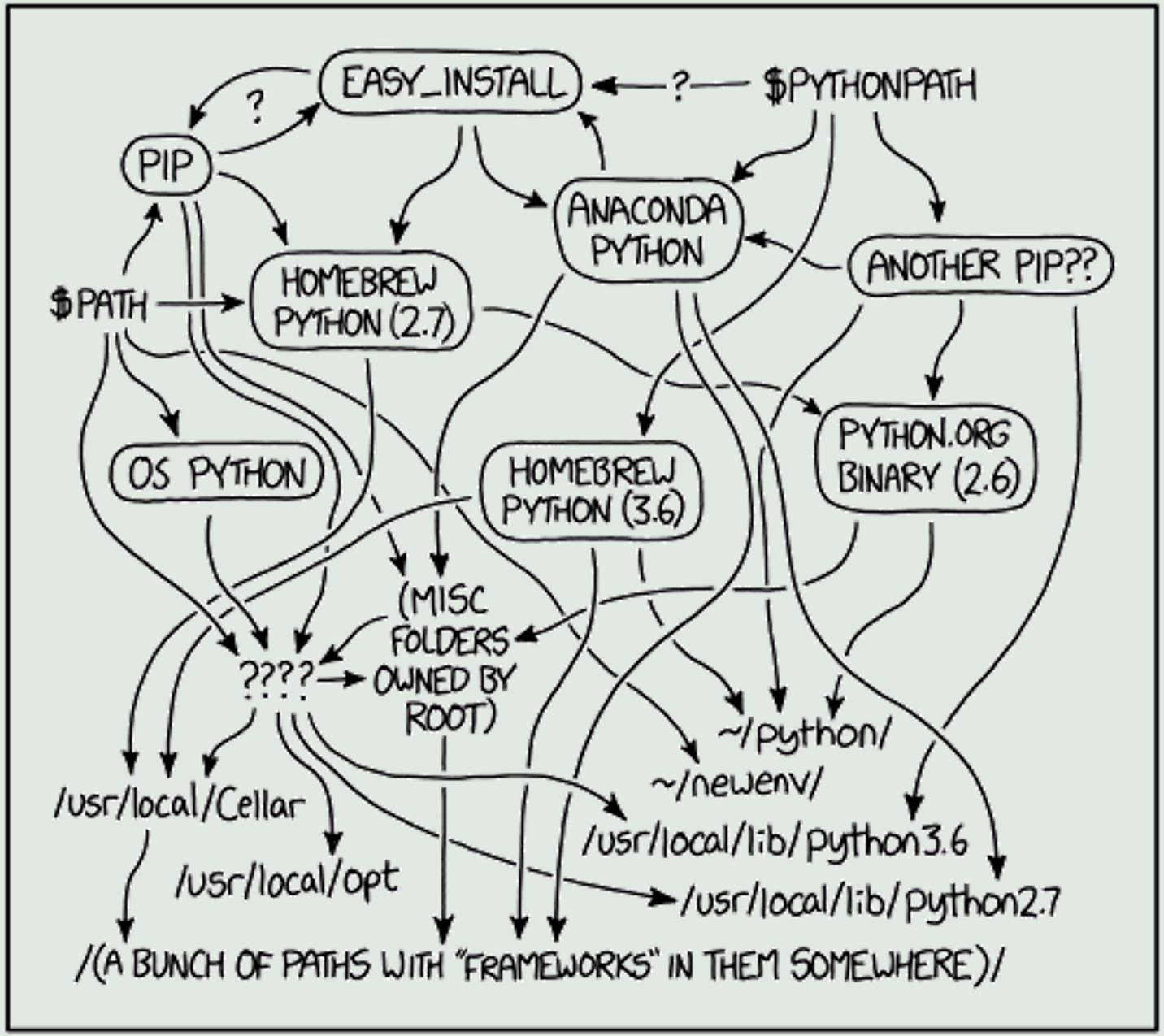
> Python lets you do a lot without demanding a lot of you. It lets you think.
>
> - Travis Oliphant

Entry barriers

*There should be one-
- and preferably
only one -- obvious
way to do it.
  - The Zen of Python*

“

*There should be one-
- and preferably
only one -- obvious
way to do it.
  - The Zen of Python*



MY PYTHON ENVIRONMENT HAS BECOME SO DEGRADED THAT MY LAPTOP HAS BEEN DECLARED A SUPERFUND SITE.

requests 2.7.0

```
pip install requests==2.7.0
```

Newer version available (2.26.0)

Released: May 3, 2015

Python HTTP for Humans.

Navigation

- Project description
- Release history
- Download files

Project links

Project description

pypi v2.26.0 downloads 135M/month

Requests is an Apache2 Licensed HTTP library, written in Python, for human beings.

Most existing Python modules for sending HTTP requests are extremely verbose and cumbersome. Python's builtin urllib2 module provides most of the HTTP capabilities you should need, but the api is thoroughly broken. It requires an enormous amount of work (even method overrides) to perform the simplest of tasks.

Things shouldn't be this way. Not in Python.

Python HTTP for Humans.

Most existing Python modules for sending HTTP requests are extremely verbose and cumbersome. Python's builtin urllib2 module provides most of the HTTP capabilities you should need, but the api is thoroughly broken. It requires an enormous amount of work (even method overrides) to perform the simplest of tasks.

Things shouldn't be this way. Not in Python.

GPT-NeoX

An implementation of model parallel GPT-3-like models on GPUs, based on the DeepSpeed library. Designed to be able to train models in the hundreds of billions of parameters or larger. This repository is under development and may change rapidly without warning.

Requirements

Everything you need to get started running the code can be installed via pip:

```
$ pip install -r requirements.txt
```

Local Environment

First make sure you are in an environment with Python 3.8 or later and `torch>=1.8` installed. Then run `pip install -r requirements/requirements.txt`. You may need to change the version of `cupy-cudaxxx` to match your machine's cuda version.

https://pythonrepo.com/repo/EleutherAI-gpt-neox
https://github.com/EleutherAI/gpt-neox#quick-start

GPT-NeoX

An implementation of model parallel GPT-3-like models on GPUs, based on the DeepSpeed library. Designed to be able to train models in the hundreds of billions of parameters or larger. This repository is under development and may change rapidly without warning.

Requirements

Everything you need to get started running the code can be installed via pip:

<install error demo>

```
$ pip install -r requirements.txt
```

Local Environment

First make sure you are in an environment with Python 3.8 or later and `torch>=1.8` installed. Then run `pip install -r requirements/requirements.txt`. You may need to change the version of `cupy-cudaxxx` to match your machine's cuda version.

https://pythonrepo.com/repo/EleutherAI-gpt-neox
https://github.com/EleutherAI/gpt-neox#quick-start

Requirements

Everything you need to get started running the code can be installed via pip:

```
$ pip install -r requirements.txt
```

except an NVIDIA CUDA GPU!!

https://docs.cupy.dev/en/stable/install.html

Y **Hacker News** new | past | comments | ask | show | jobs | submit login

▲ breatheoften on April 20, 2017 | parent | favorite | on: New IPython release drops Python 2.7 support

A comment from a beginner python user -- but I need to rant ...

I've been doing a bit of python the last few weeks for some image processing/computer vision tasks (using opencv and numpy).

I have to say, all together it's a pretty miserable developer experience.

Python is incredibly slow - forcing pretty much all computation no matter how trivial into contortions via numpy incantations so that the inner loops can run in native extensions -- these Incantations have a lot of implicit not well documented magic. Miss some details in the behavior of the magic and suddenly you have a major 10x slowdown -- but good luck finding where. I would kill for an easy to use tool like xcodes Time Profiler ...

API usage errors (even those where invariants are checked at runtime) are ridiculously under informative -- opencv for example does quite a bit of runtime sanity checking on the shape and type of arguments to its methods -- but somehow even simple details as to which parameter is the cause of the error don't get reported in the stack trace severely increasing the cognitive load required to identify the mismatch -- not fun when multiple arguments to an API are the result of a chain of numpy magic data munging. This may be an opencv complaint more than python (aside: opencv is pretty terrible.)

I'm not sure what I'm doing wrong with python but I find the majority of my code to be sort of menial data munging -- and I haven't figured out good patterns to organize this munging in any sensical way --with a static language d.r.y patterns to centralize such plumbing operations have the awesome effect of moving invariants into reasonable places -- in python without any ability to organjze guarantees, as the code base evolves I find myself needing to repeatedly check data shapes/types -- there doesn't seem to be an obviously useful way to organize verification of data types as the necessary invariants become apparent. These issues are compounded by the fact that refactoring is an enormous pain in the ass!

https://news.ycombinator.com/item?id=14158269

Informative Errors

```
1 # 01_pep626.py
2
3 a = {'foo': 'blah', 'bar': 'blah blah’
4 print("You see what I'm talking!?!")
```

```
1 # 01_pep626.py
2
3 a = {'foo': 'blah', 'bar': 'blah blah’
4 print("You see what I'm talking!?!")
```

```
$> python3.8 01_pep626.py
  File "01_pep626.py", line 4
    print("You see what I'm talking!?!")
    ^
SyntaxError: invalid syntax
```

```
1 # 01_pep626.py
2
3 a = {'foo': 'blah', 'bar': 'blah blah’
4 print("You see what I'm talking!?!")
```

```
$> python3.8 01_pep626.py
  File "01_pep626.py", line 4
    print("You see what I'm talking!?!")
    ^
SyntaxError: invalid syntax
```

```
1 # 01_pep626.py
2
3 a = {'foo': 'blah', 'bar': 'blah blah’
4 print("You see what I'm talking!?!")
```

```
$> python3.10 01_pep626.py
  File "/Users/vasu_mac/src/github_repos/Default/pydata_demo/01_pep626.py", line 3
    a = {'foo': 'blah', 'bar': 'blah blah'
        ^
SyntaxError: '{' was never closed
```

Fail Clearly

- Show me up front what is wrong!!

- Show me up front what is wrong!!

```
mac-0111:~ ueiw$ pytest

Your tests failed... too bad!
You can find out why here: /tmp/eb6c0068f4a2fce0992048d31f1204ebaad31237a1/pytest-25/eb6c0fea162c.log
Good luck!

mac-0111:~ ueiw$
```

Fail Clearly and Fail Early

<Fail Early Demo #1>

- **Fail early** – validate the input as soon as you get it

Format [edit]

In its canonical textual representation, the 16 octets of a UUID are represented as 32 hexadecimal (base-16) digits, displayed in five groups separated by hyphens, in the form 8-4-4-4-12 for a total of 36 characters (32 hexadecimal characters and 4 hyphens). For example:

```
123e4567-e89b-12d3-a456-426614174000
```

```
xxxxxxxx-xxxx-Mxxx-Nxxx-xxxxxxxxxxxx
```

- https://en.wikipedia.org/wiki/Universally_unique_identifier

<Fail Early Demo #2>

- **Fail early** – validate the input as soon as you get it

```
>>> from uuid import UUID
>>> UUID('abc')
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "/Users/vasu_mac/anaconda/lib/python3.6/uuid.py", line 140, in __init__
    raise ValueError('badly formed hexadecimal UUID string')
ValueError: badly formed hexadecimal UUID string
```

```
def _validate_uuid(uuid: str) -> bool:
    return re.match(r"[a-zA-Z0-9+\-_]+$", uuid) is not None
```

- https://github.com/JDASoftwareGroup/kartothek/blob/6bc7e868435e98cbda0b695900f29d1ff7d49110/kartothek/core/dataset.py#L39

- If you have a CLI based tool, “do you mean <this argument> also help”, like Git

Don’t silently ignore stuff

<silent ignore demo>

Sensible Defaults

<pandas columns demo>

Just to be clear..
DON’T do that stuff

Accessible Documentation

Making it easier for your users,
makes it easier for you

- Tutorials (learning oriented)
  - Immediate sense of achievement
- How-to Guides (problem-oriented)
  - Steps to solve a common problem
- Reference (Information oriented)
  - Include caveats

- Automated documentation with version control — sphinx, etc.
- Test your documentation — doctest
- Code comments (duh)
- Easy to navigate

Thank you!