

WiFi LIVE ONLINE TRAINING

Data Engineering for Data Scientists

Build resilient pipelines to support stronger models



MAX HUMBER



March 26, 2021
12:00pm – 2:00pm EDT

Agenda

1

Extend

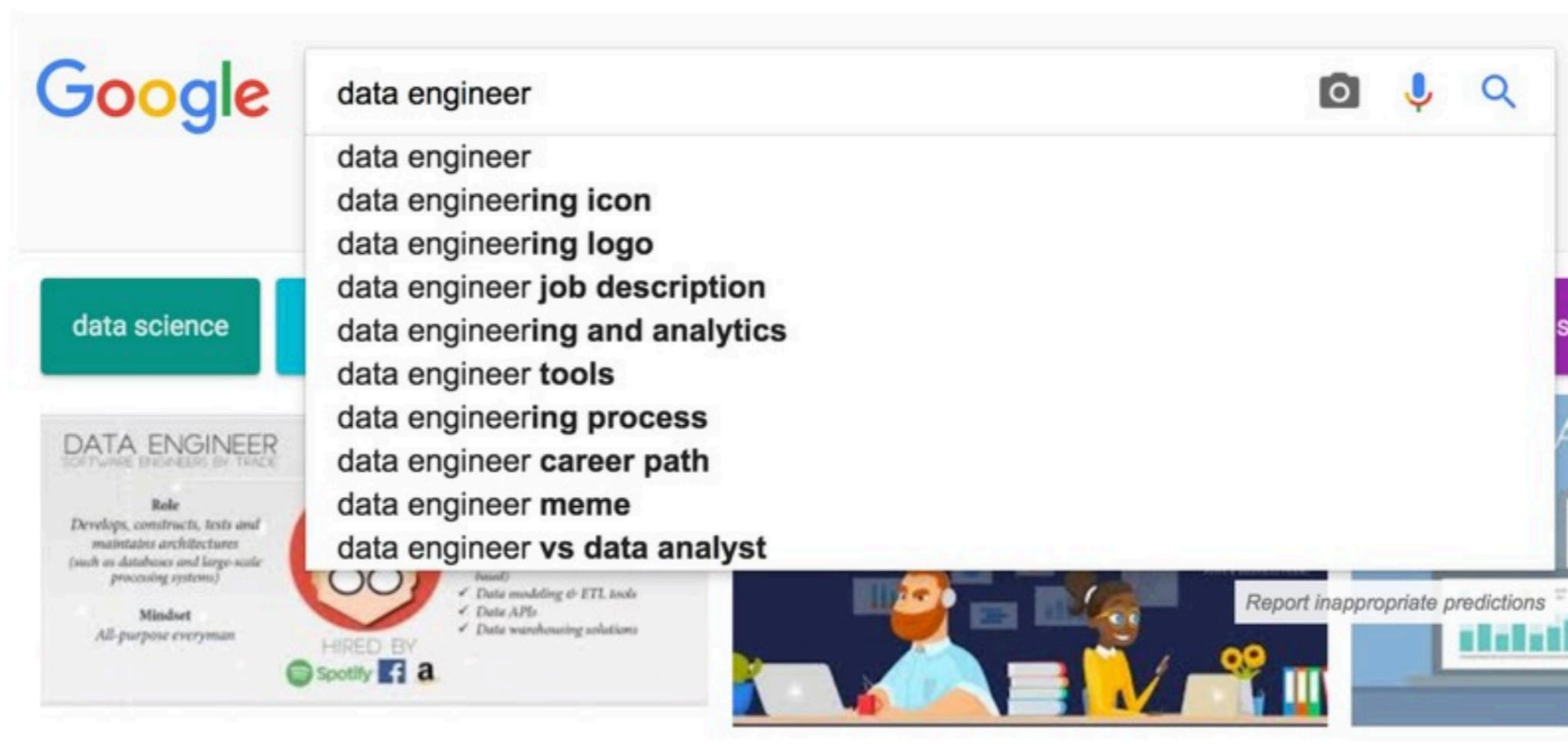
2

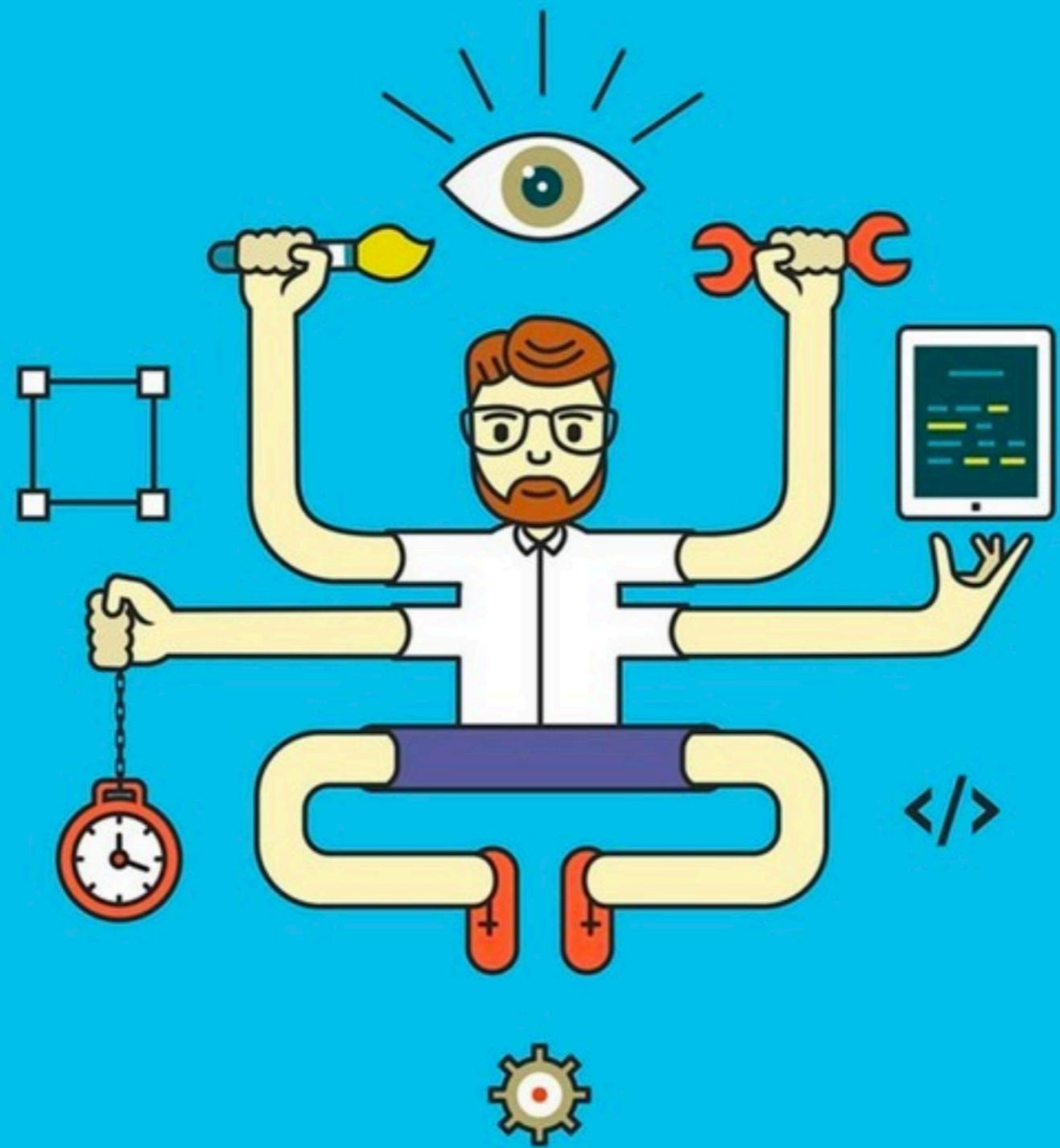
Save

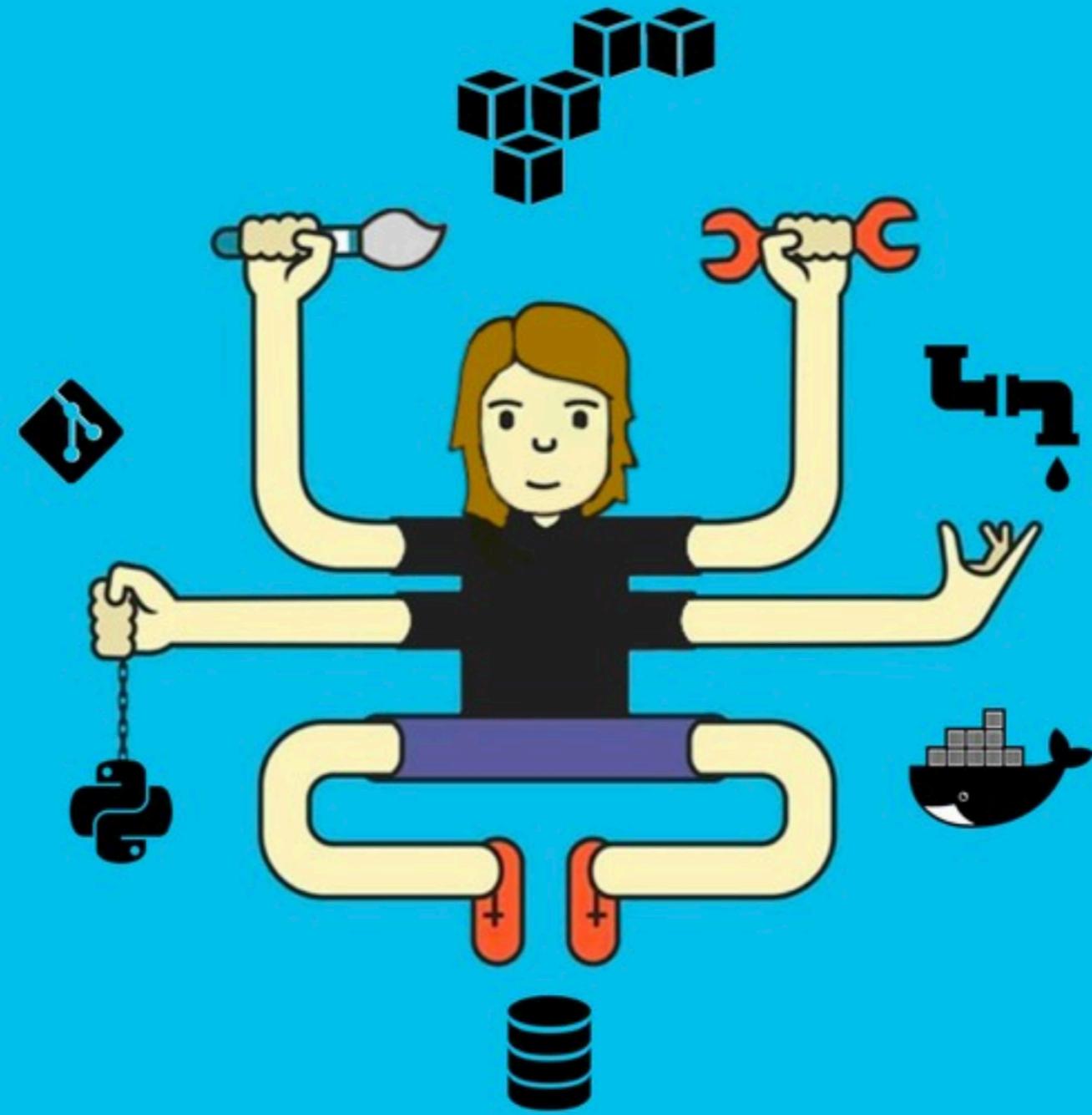
3

Schedule

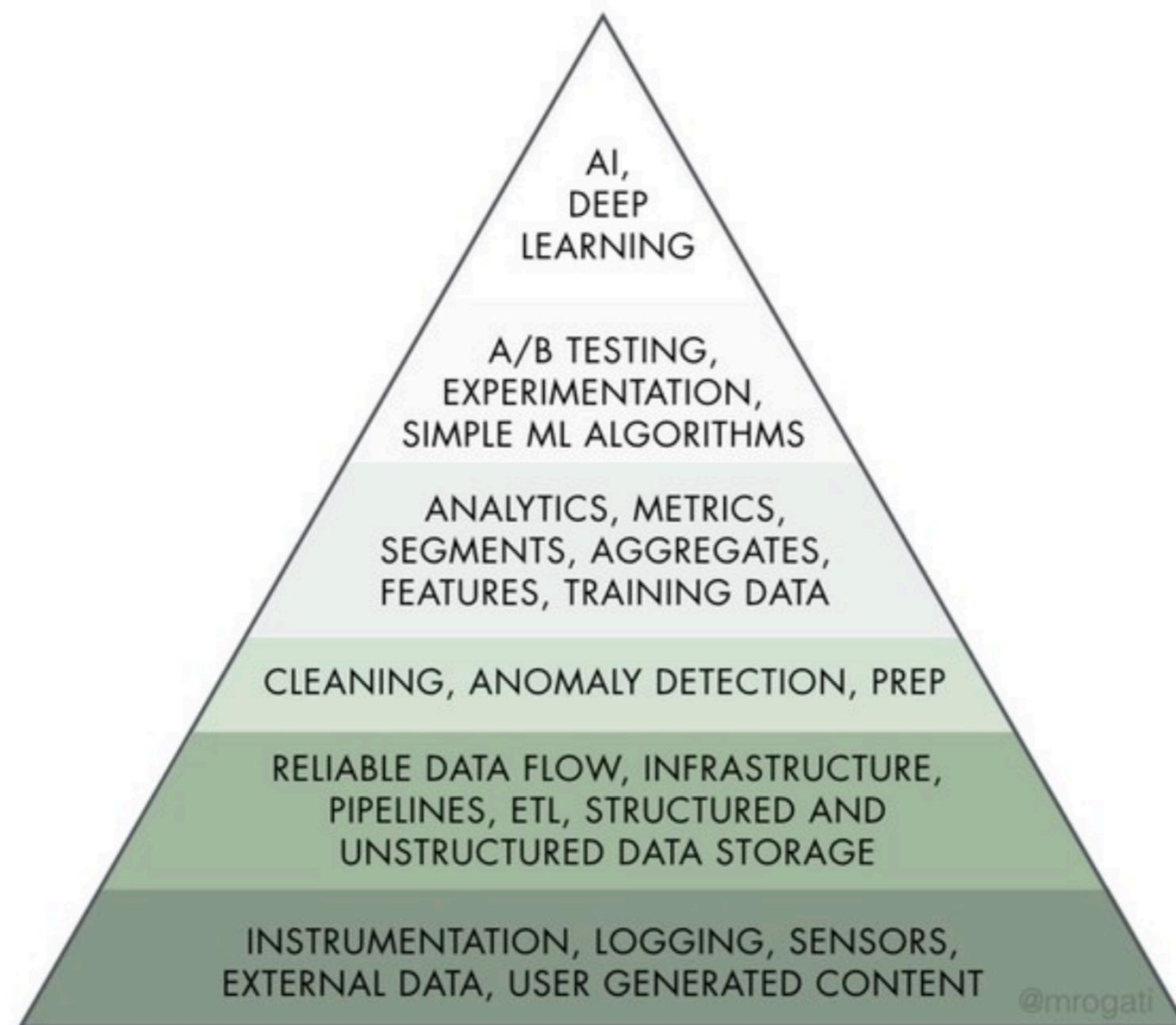
Data Engineering



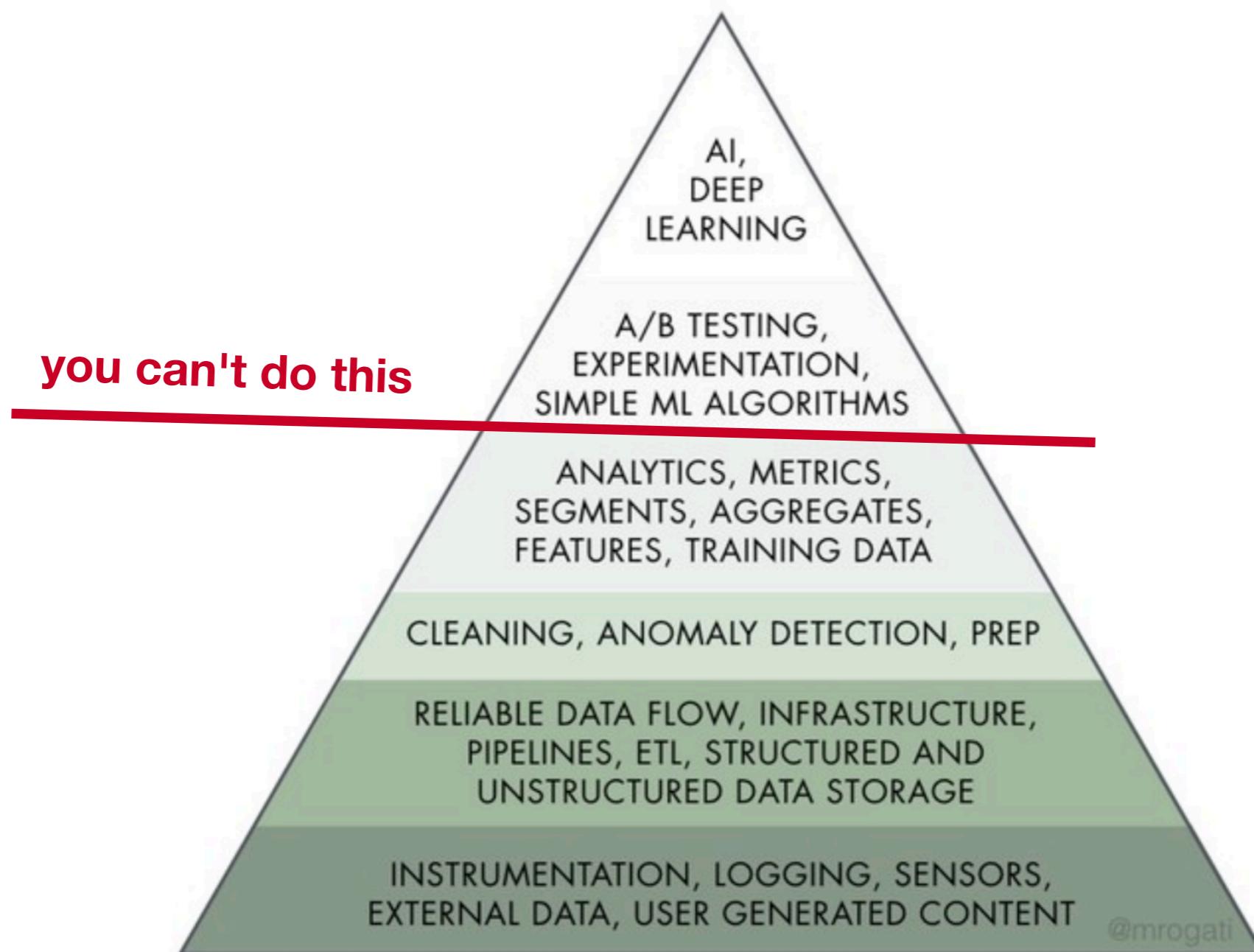




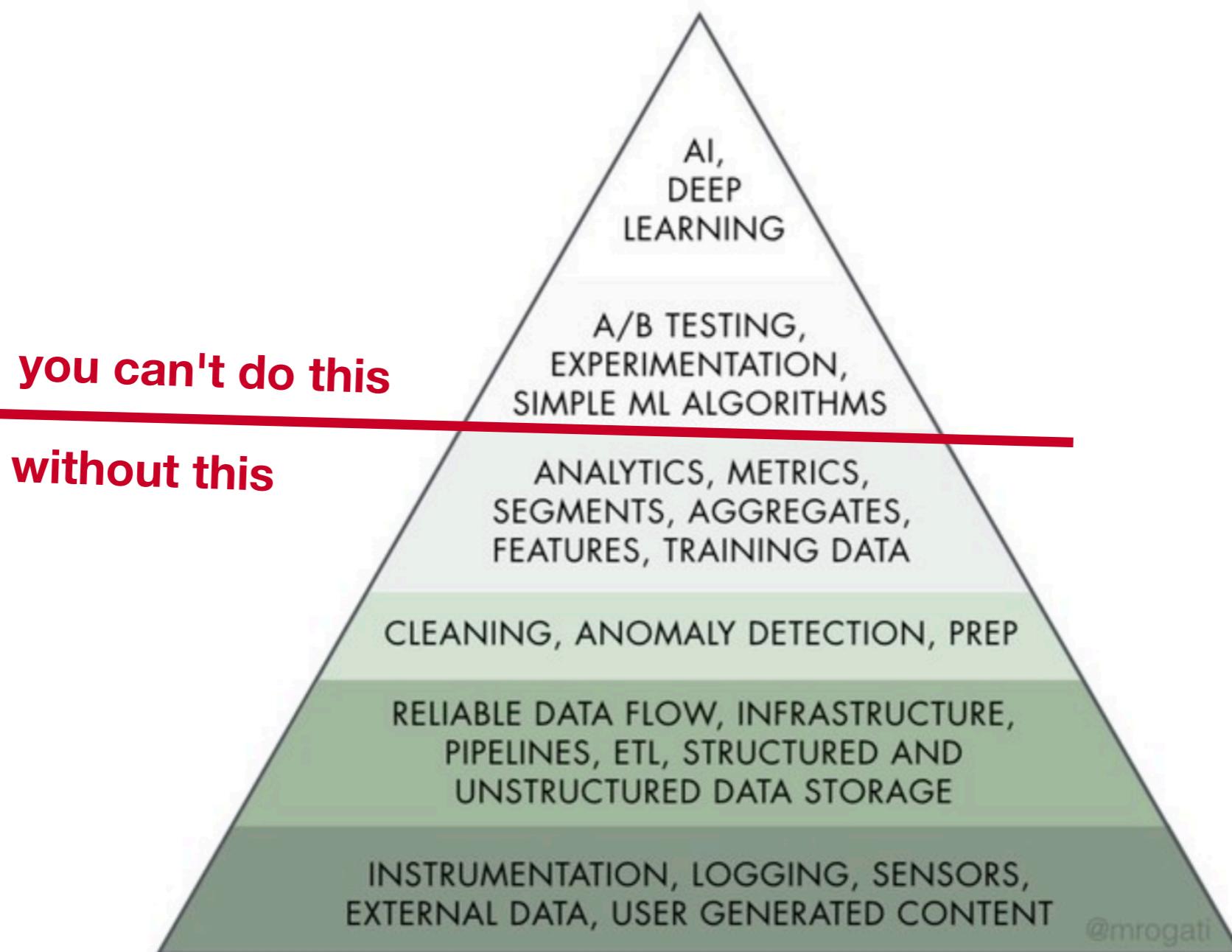
Data Hierarchy of Needs



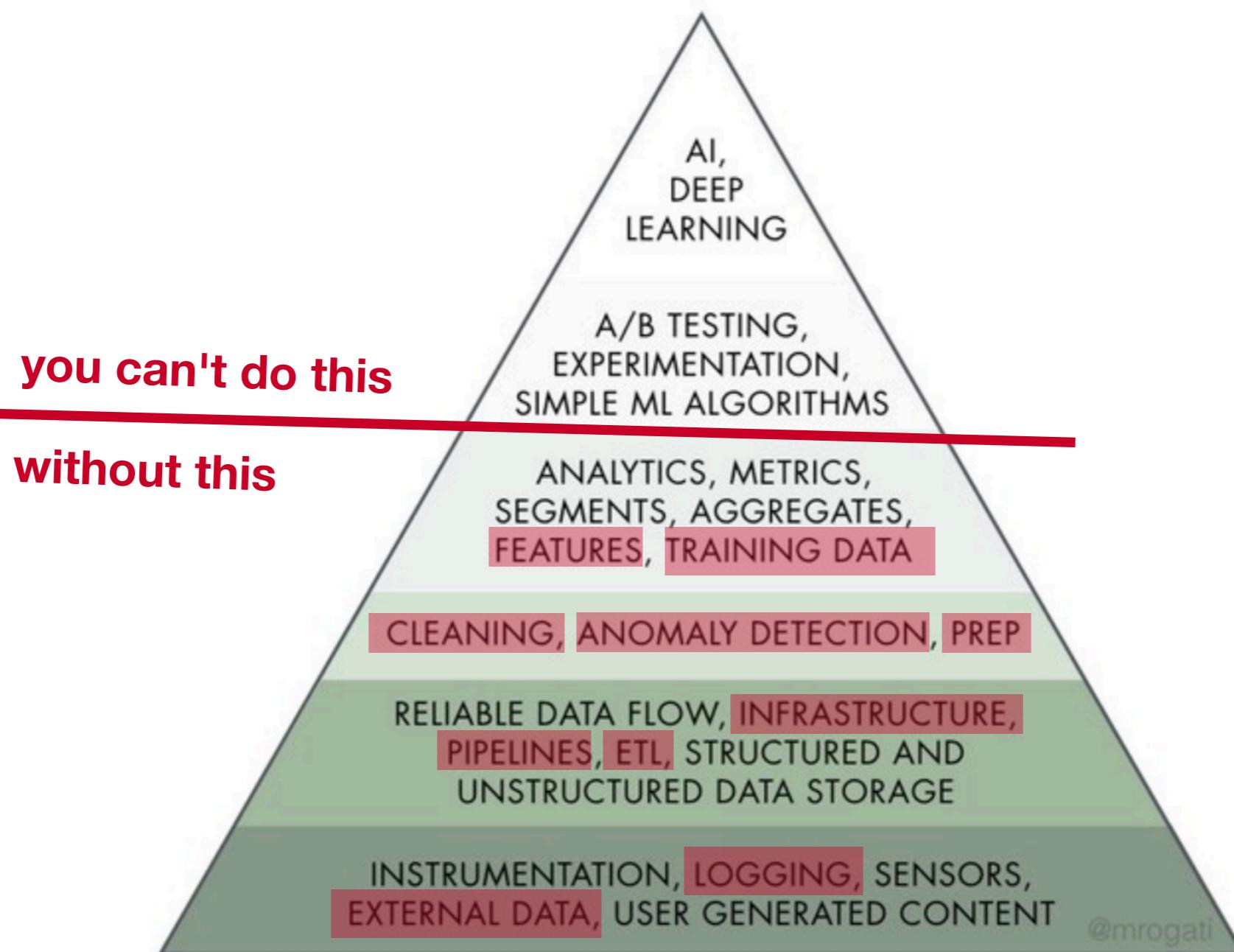
Data Hierarchy of Needs

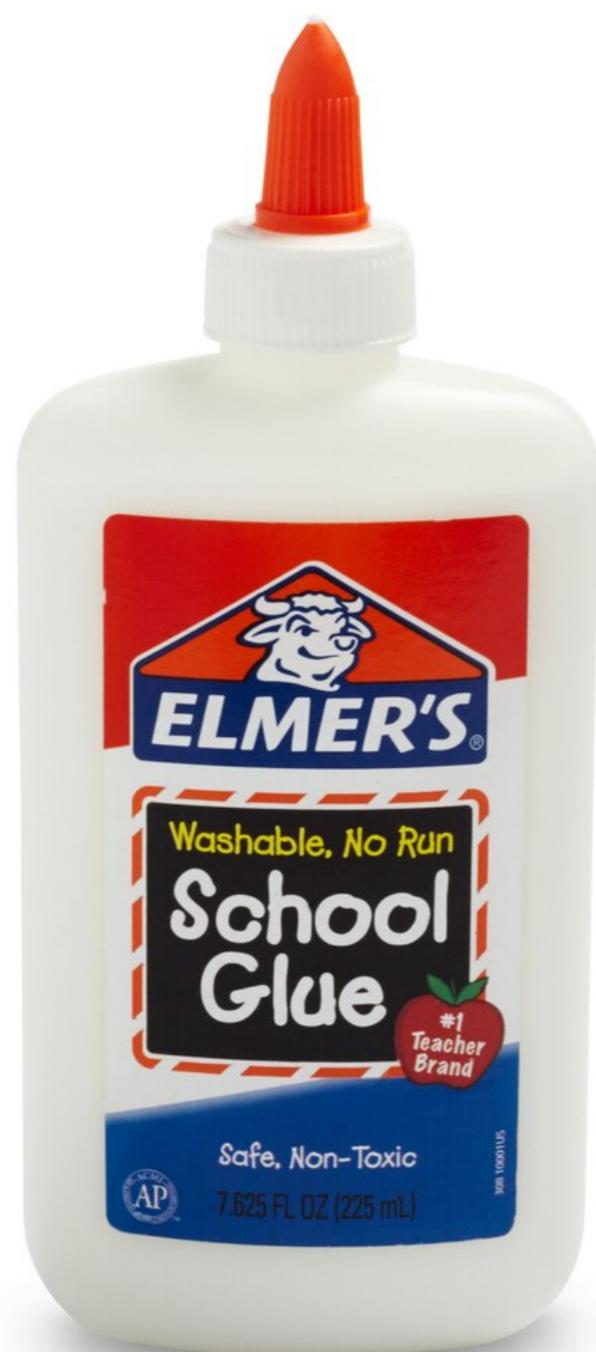


Data Hierarchy of Needs



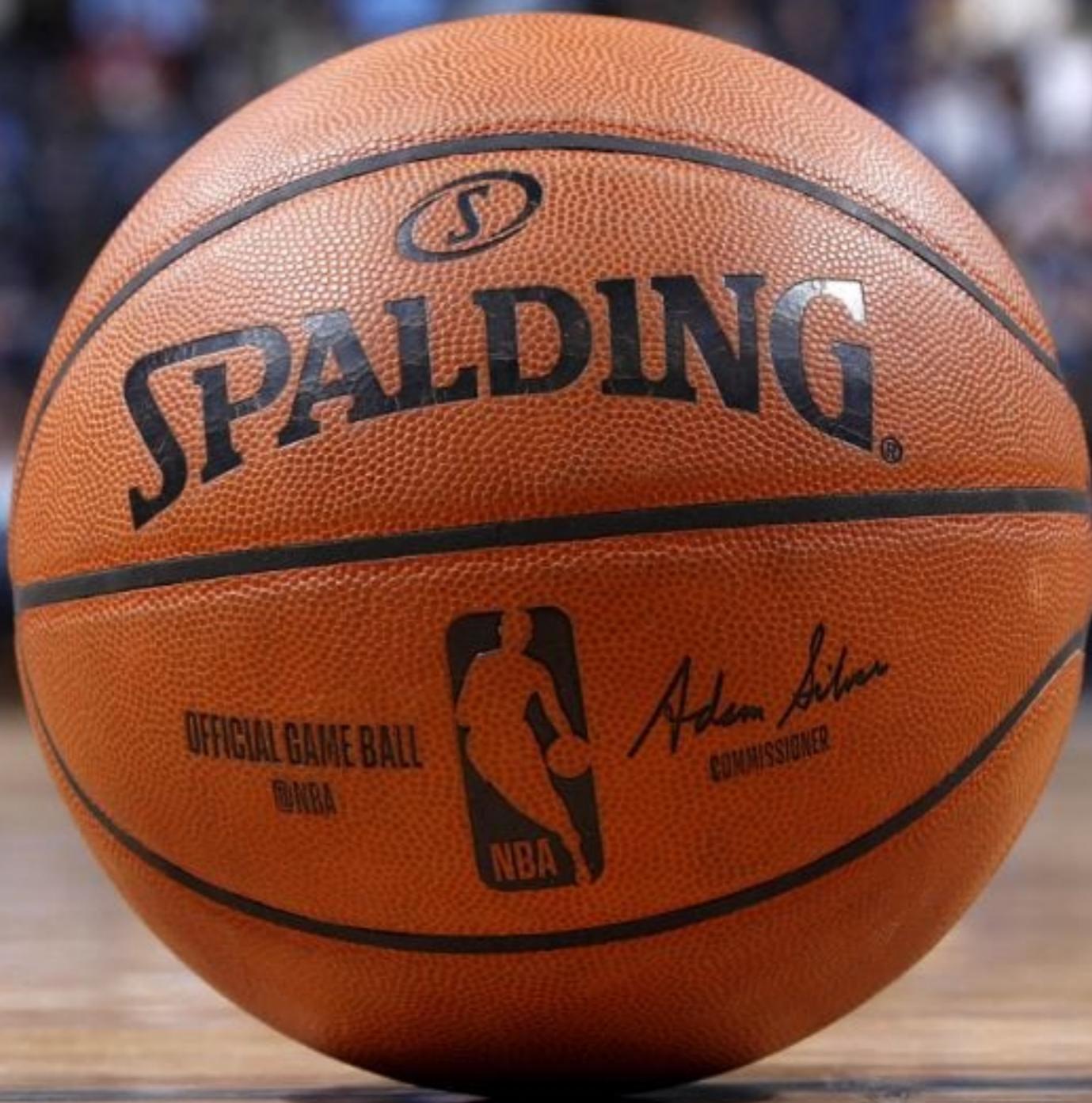
Data Hierarchy of Needs











1

Extend



01-model.ipynb

Q&A

Jupyter to Hydrogen



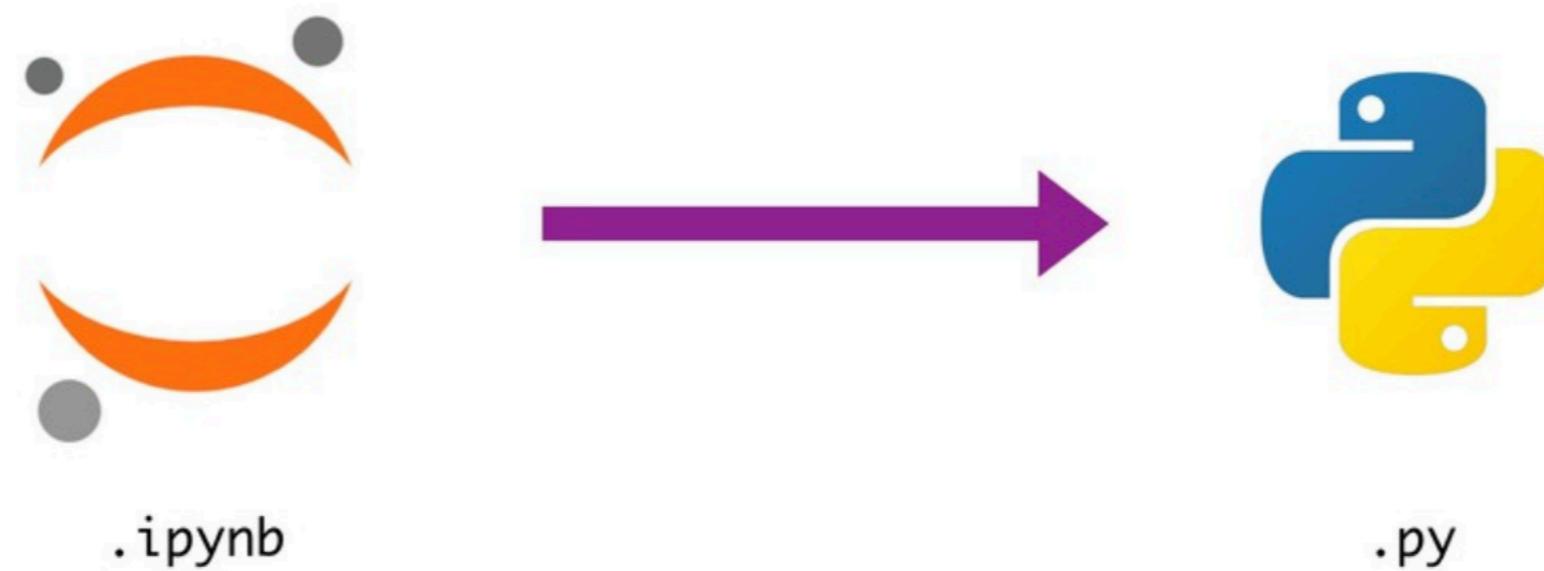
Jupyter to Hydrogen



- exploratory analysis
- visualizing ideas
- prototyping

- messy
- bad at versioning
- not ideal for production

Jupyter to Hydrogen

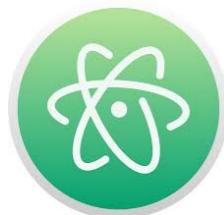


Hydrogen



slack 12/773 build passing

Hydrogen is an interactive coding environment that supports Python, R, JavaScript and other Jupyter kernels.



The screenshot shows the Hydrogen Atom package integrated into the Atom text editor. The main window displays a Python script named 'demo.py' with the following code:

```
demo.py — /Users/lukasgeiger/Desktop
demo.py
24      D = np.array([1, 2, 3, 4, 5], dtype= int32 ),
25      'E': pd.Categorical(["test", "train", "test", "train"]),
26      'F': 'foo'})'
27
28 # %% Render Latex
29 x, y, z = sp.symbols('x, y, z')
30 f = sp.sin(x * y) + sp.cos(y * z)
31
32 sp.integrate(f, x)
33
```

Below the code, a data frame 'df' is displayed as a table:

	A	B	C	D	E	F
0	1.0	2013-01-02	1.0	3	test	foo
1	1.0	2013-01-02	1.0	3	train	foo
2	1.0	2013-01-02	1.0	3	test	foo
3	1.0	2013-01-02	1.0	3	train	foo

The bottom status bar indicates 'demo.py 33:1 Python 3: idle' and includes icons for file operations and settings.

<https://atom.io/packages/hydrogen>



02-model.py

Q&A





Python Fire

python 2.7 | 3.4 | 3.5 | 3.6

Python Fire is a library for automatically generating command line interfaces (CLIs) from absolutely any Python object.

- Python Fire is a simple way to create a CLI in Python. [\[1\]](#)
- Python Fire is a helpful tool for developing and debugging Python code. [\[2\]](#)
- Python Fire helps with exploring existing code or turning other people's code into a CLI. [\[3\]](#)
- Python Fire makes transitioning between Bash and Python easier. [\[4\]](#)
- Python Fire makes using a Python REPL easier by setting up the REPL with the modules and variables you'll need already imported and created. [\[5\]](#)

Installation

To install Python Fire with pip, run: `pip install fire`

To install Python Fire with conda, run: `conda install fire -c conda-forge`

To install Python Fire from source, first clone the repository and then run: `python setup.py install`



03-predict.py



04-fire.py

🔥 python 04-fire.py "Fred VanVleet"

Q&A

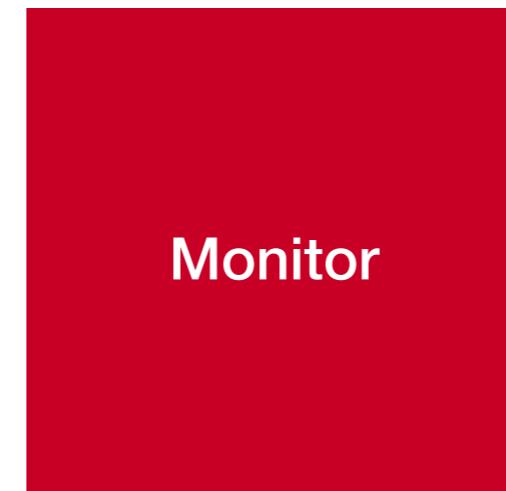
2

Save



05-database.py

🔥 python 05-database.py “Kyle Lowery”



Monitor







FREE EDITION

5,000 events per month.
Includes all Standard features.

[Try it free](#)

⌚ Waiting for data from your Python app...

This page will update once we receive an error from your app.

Don't want to start with Python? [Choose a different SDK.](#)

Installation

To send errors to Rollbar from your Python application you should use the [pyrollbar](#) SDK. Install pyrollbar with pip:

```
pip install rollbar
```

COPY

Send a Message

You'll need your project's server-side access token to initialize the pyrollbar SDK. Sending a message to the Rollbar server is as simple as:

```
import rollbar  
  
rollbar.init('49e86c1178e346899c2f29e3744420a9')  
rollbar.report_message('Rollbar is configured correctly')
```

COPY

A few seconds after you execute this code, the message should appear on your project's "Items" page.



remove password

Search

Repositories	244
Code	50M+
Commits	508K
Issues	280K
Packages	1
Marketplace	0
Topics	0
Wikis	34K
Users	0

[Advanced search](#) [Cheat sheet](#)

Showing 445,215 available commit results ⓘ

Sort: Recently committed ▾

removed password ...

Verified



c64833c

 reharr4 committed to [reharr4/blamazon](#) 2 hours ago**Onmibus updates (#7)** ...

Verified



b1b80c9

 stevector committed to [stevector/internet-button-playground](#) 4 hours ago ✓**Merge pull request #534 in EDD/edd-django from ~WCMORRELL/edd:feature...** ...

8aa8978

 wmorrell committed to [JBEI/edd](#) 7 hours ago**File restructuring** ...

8735106

 caitlin authored and PilotBob committed to [CassandraWerewolf/CassandraWeb](#) 7 hours ago

removed password

removed password from bamazonCustomer.js

master

 reharr4 committed 2 hours ago Verified

1 parent 7e38084 commit c64833c915cc0f604cb0eeab

 Showing 1 changed file with 2 additions and 2 deletions.

▼ 4  bamazonCustomer.js 

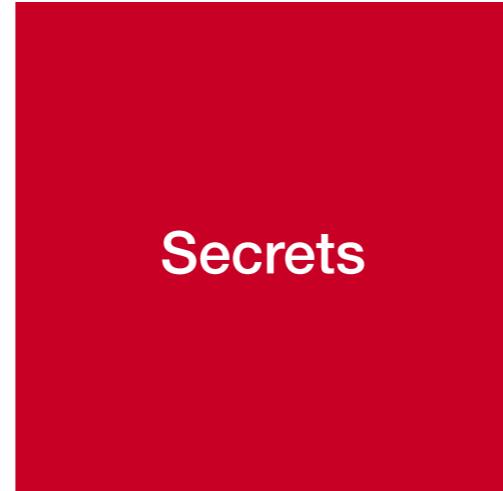
 @@ -9,7 +9,7 @@ var connection = mysql.createConnection({

```
 9    9      // username
 10   10      user: "root",
 11   11      // password
 12   -      password: "L!sb0n13",
 12   +      password: "",  
 13   13      database: "bamazon_db"
 14   14      });
 15   15
```

  @@ -94,4 +94,4 @@ connection.connect(function (err) {

```
94   94      //      }
95   95      //      }
96   96      // }
97   -      connection.end(); ↵
97   +      connection.end();
```





Secrets



python-dotenv |

build passing

coverage 90%

pypi package 0.10.3

Say Thanks !

Reads the key,value pair from `.env` file and adds them to environment variable. It is great for managing app settings during development and in production using [12-factor](#) principles.

| Do one thing, do it well!

- [Usages](#)
- [Installation](#)
- [Command-line interface](#)
- [iPython Support](#)
- [Setting config on remote servers](#)
- [Related Projects](#)
- [Contributing](#)
- [Changelog](#)

Installation

```
pip install -U python-dotenv
```



06-rollbar.py

Q&A

3

Schedule



Airflow

Apache Airflow

[pypi package 1.10.3](#) [build passing](#) [coverage 79%](#) [docs passing](#) [license Apache 2](#) [python 2.7 | 3.5](#) [Follow](#) [2.4k](#) [Slack](#) [join chat](#)

Apache Airflow (or simply Airflow) is a platform to programmatically author, schedule, and monitor workflows.

When workflows are defined as code, they become more maintainable, versionable, testable, and collaborative.

Use Airflow to author workflows as directed acyclic graphs (DAGs) of tasks. The Airflow scheduler executes your tasks on an array of workers while following the specified dependencies. Rich command line utilities make performing complex surgeries on DAGs a snap. The rich user interface makes it easy to visualize pipelines running in production, monitor progress, and troubleshoot issues when needed.

Getting started

Please visit the Airflow Platform documentation (latest **stable** release) for help with [installing Airflow](#), getting a [quick start](#), or a more complete [tutorial](#).

Documentation of GitHub master (latest development branch): [ReadTheDocs Documentation](#)

For further information, please visit the [Airflow Wiki](#).

Tutorial

This tutorial walks you through some of the fundamental Airflow concepts, objects, and their usage while writing your first pipeline.

Example Pipeline definition

Here is an example of a basic pipeline definition. Do not worry if this looks complicated, a line by line explanation follows below.

```
"""
Code that goes along with the Airflow tutorial located at:
https://github.com/apache/airflow/blob/master/airflow/example_dags/tutorial.py
"""

from airflow import DAG
from airflow.operators.bash_operator import BashOperator
from datetime import datetime, timedelta

default_args = {
    'owner': 'airflow',
    'depends_on_past': False,
    'start_date': datetime(2015, 6, 1),
    'email': ['airflow@example.com'],
    'email_on_failure': False,
    'email_on_retry': False,
    'retries': 1,
    'retry_delay': timedelta(minutes=5),
    # 'queue': 'bash_queue',
    # 'pool': 'backfill',
    # 'priority_weight': 10,
    # 'end_date': datetime(2016, 1, 1),
}

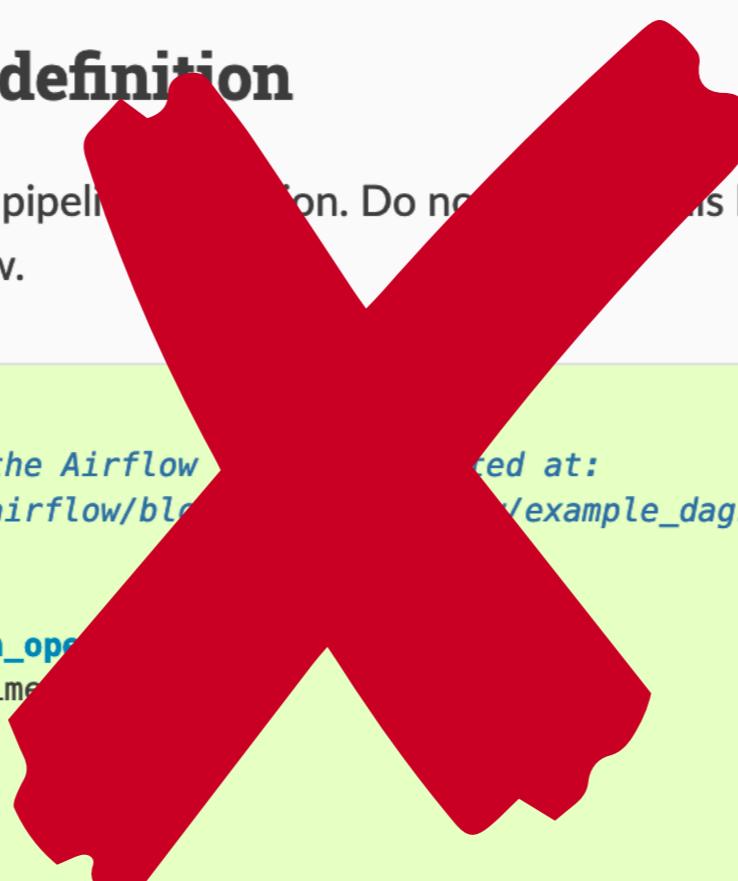
dag = DAG('tutorial', default_args=default_args, schedule_interval=timedelta(days=1))
```

Tutorial

This tutorial walks you through some of the fundamental Airflow concepts, objects, and their usage while writing your first pipeline.

Example Pipeline definition

Here is an example of a basic pipeline definition. Do not worry if this looks complicated, a line by line explanation follows below.



```
"""
Code that goes along with the Airflow tutorial located at:
https://github.com/apache/airflow/blob/master/airflow/example_dags/tutorial.py
"""

from airflow import DAG
from airflow.operators.bash_operator import BashOperator
from datetime import datetime, timedelta

default_args = {
    'owner': 'airflow',
    'depends_on_past': False,
    'start_date': datetime(2015, 6, 1),
    'email': ['airflow@example.com'],
    'email_on_failure': False,
    'email_on_retry': False,
    'retries': 1,
    'retry_delay': timedelta(minutes=5),
    # 'queue': 'bash_queue',
    # 'pool': 'backfill',
    # 'priority_weight': 10,
    # 'end_date': datetime(2016, 1, 1),
}

dag = DAG('tutorial', default_args=default_args, schedule_interval=timedelta(days=1))
```

live coding

```
# make sure you're in the right folder
cd DE4DS

# install
pip install -U apache-airflow

# set a home
export AIRFLOW_HOME=`pwd`/airflow

# initialize the database
airflow db init

# in airflow.cfg set:
# load_examples = False
```



user setup

```
airflow users create \
    --username admin \
    --firstname Peter \
    --lastname Parker \
    --role Admin \
    --email spiderman@superhero.org
```





airflow/dags/basketball.py

💨 airflow tasks test basketball fetch <date>



```
airflow tasks test basketball fetch <date>

# start the web server

airflow webserver -p 8080

# start the scheduler (new window)

export AIRFLOW_HOME=`pwd`/airflow
airflow scheduler
```





<https://github.com/maxhumber/hickory>

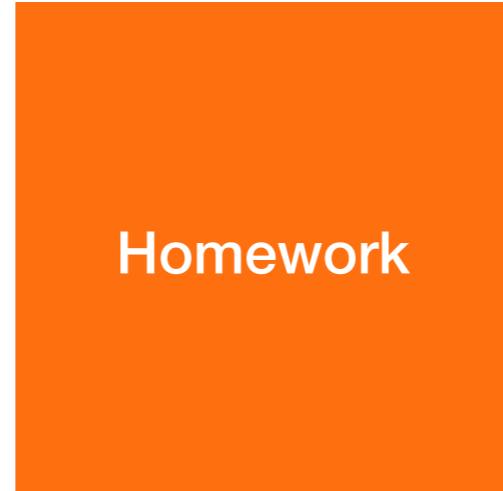
```
pip install hickory
```

```
hickory schedule 07-continuous.py --every=day@9:30am
```

```
hickory status
```

```
hickory kill 07-continuous.py
```





Homework



add t3 to save predictions to the db!

Q&A

That's all Folks!