

WiFi LIVE ONLINE TRAINING

# Data Engineering for Data Scientists

Build resilient pipelines to support stronger models



MAX HUMBER



# Agenda

1

Extend

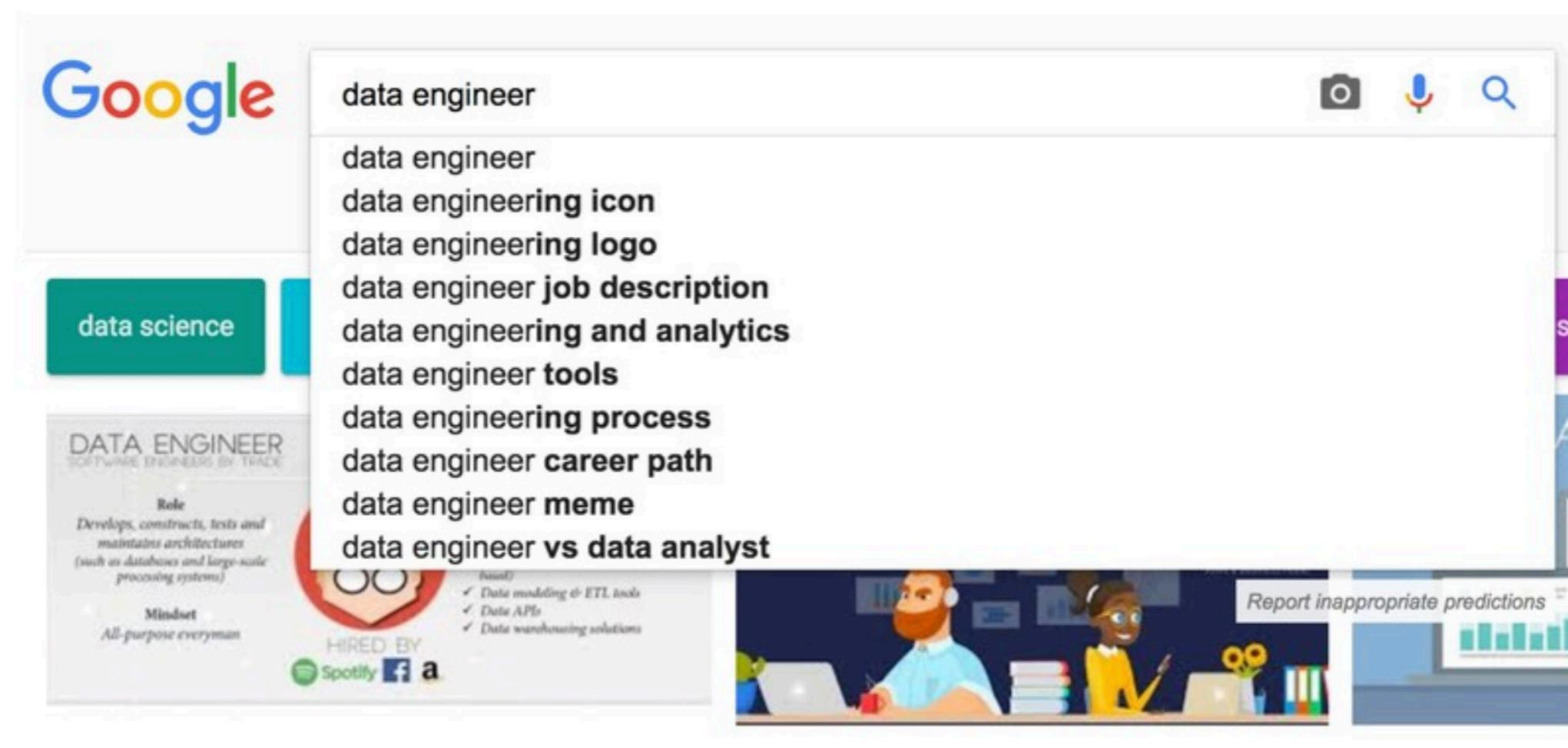
2

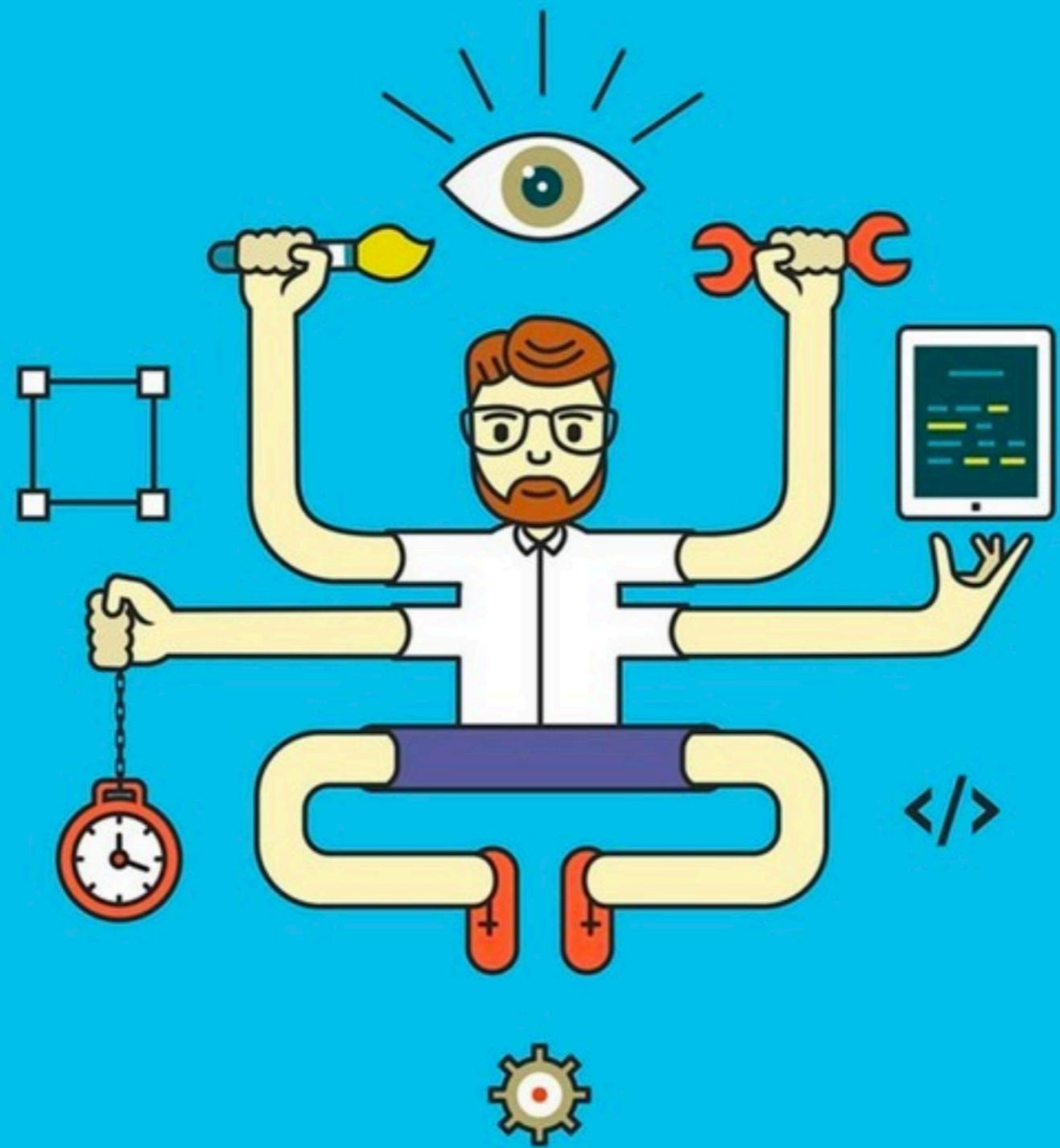
Save

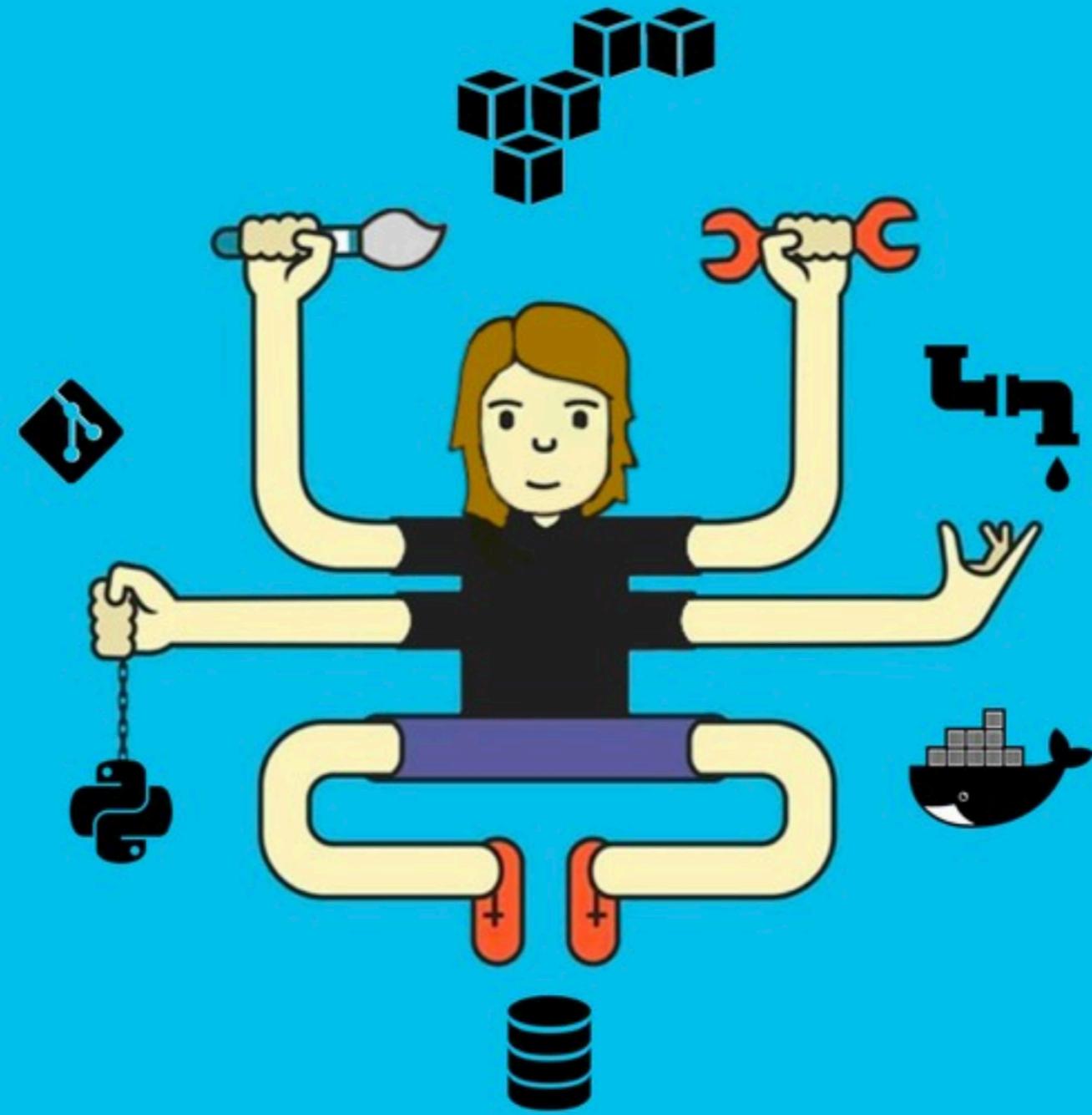
3

Schedule

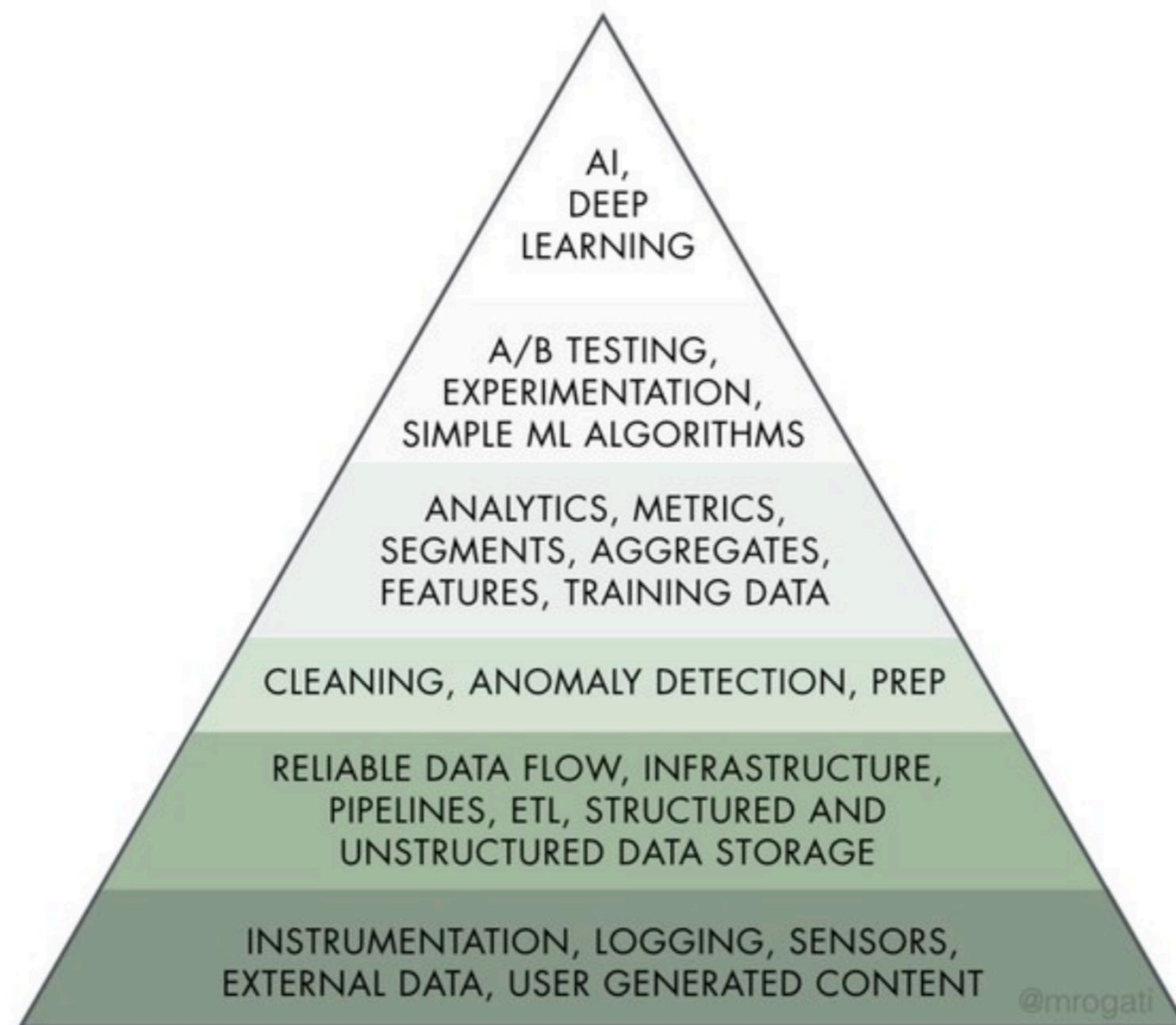
# Data Engineering



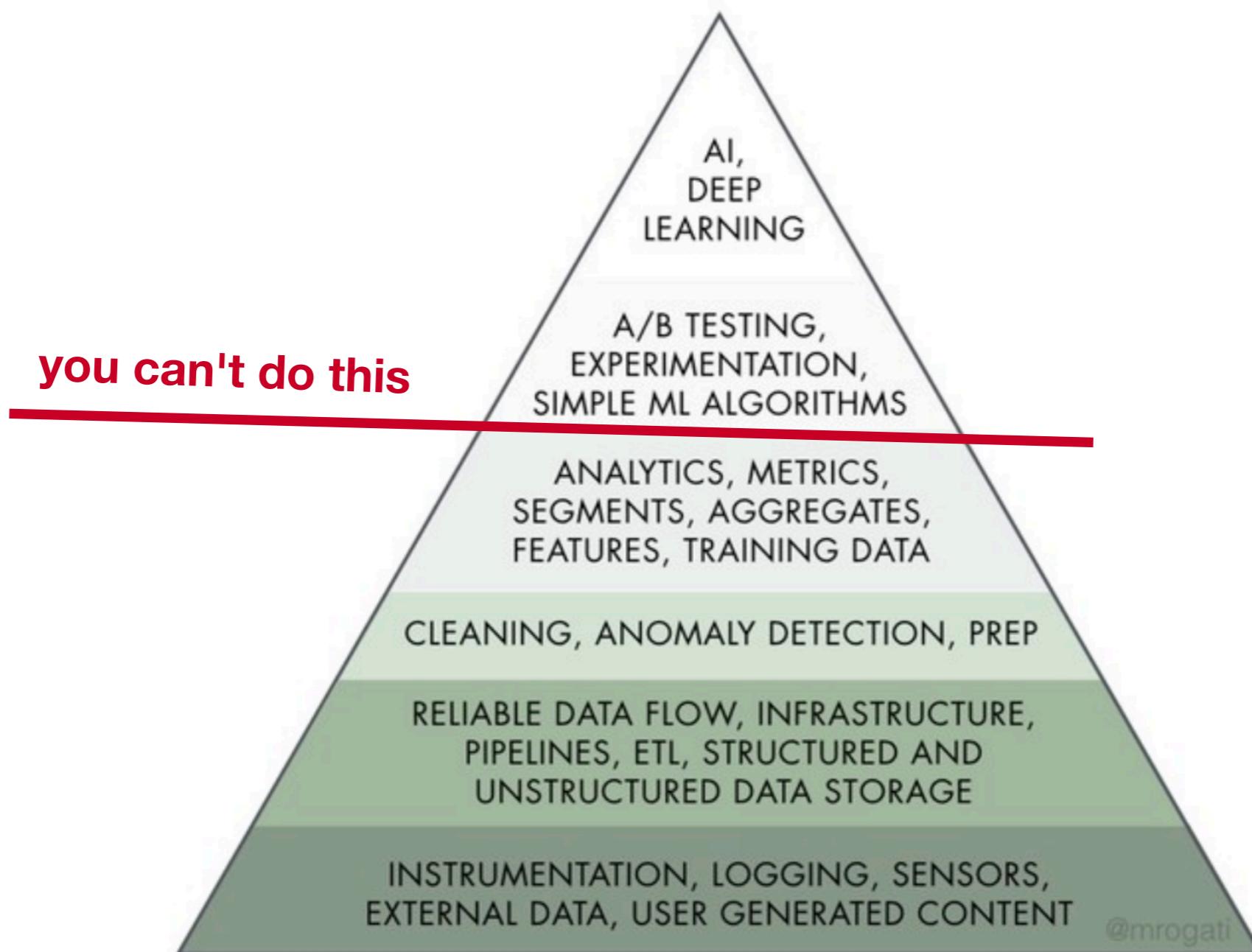




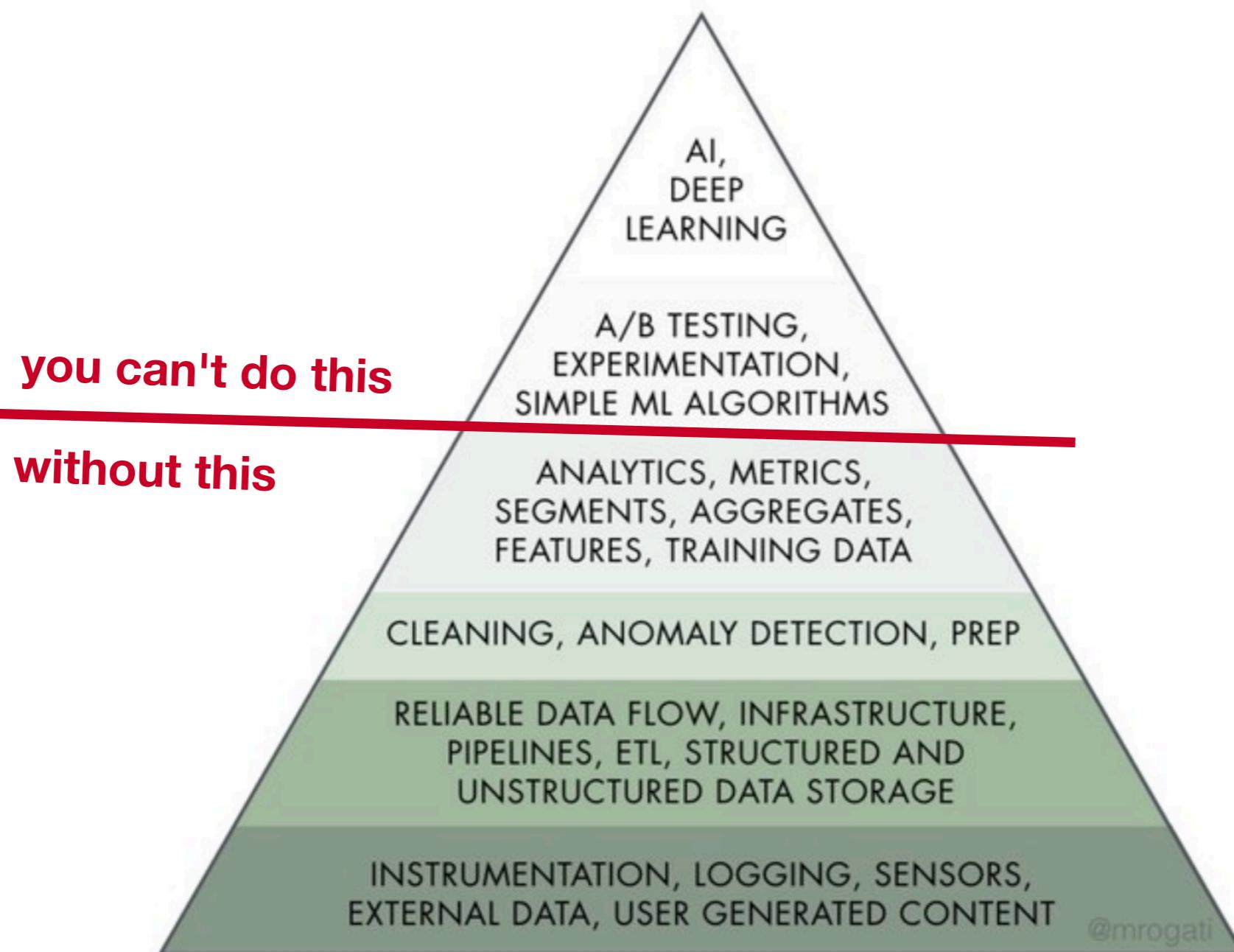
# Data Hierarchy of Needs



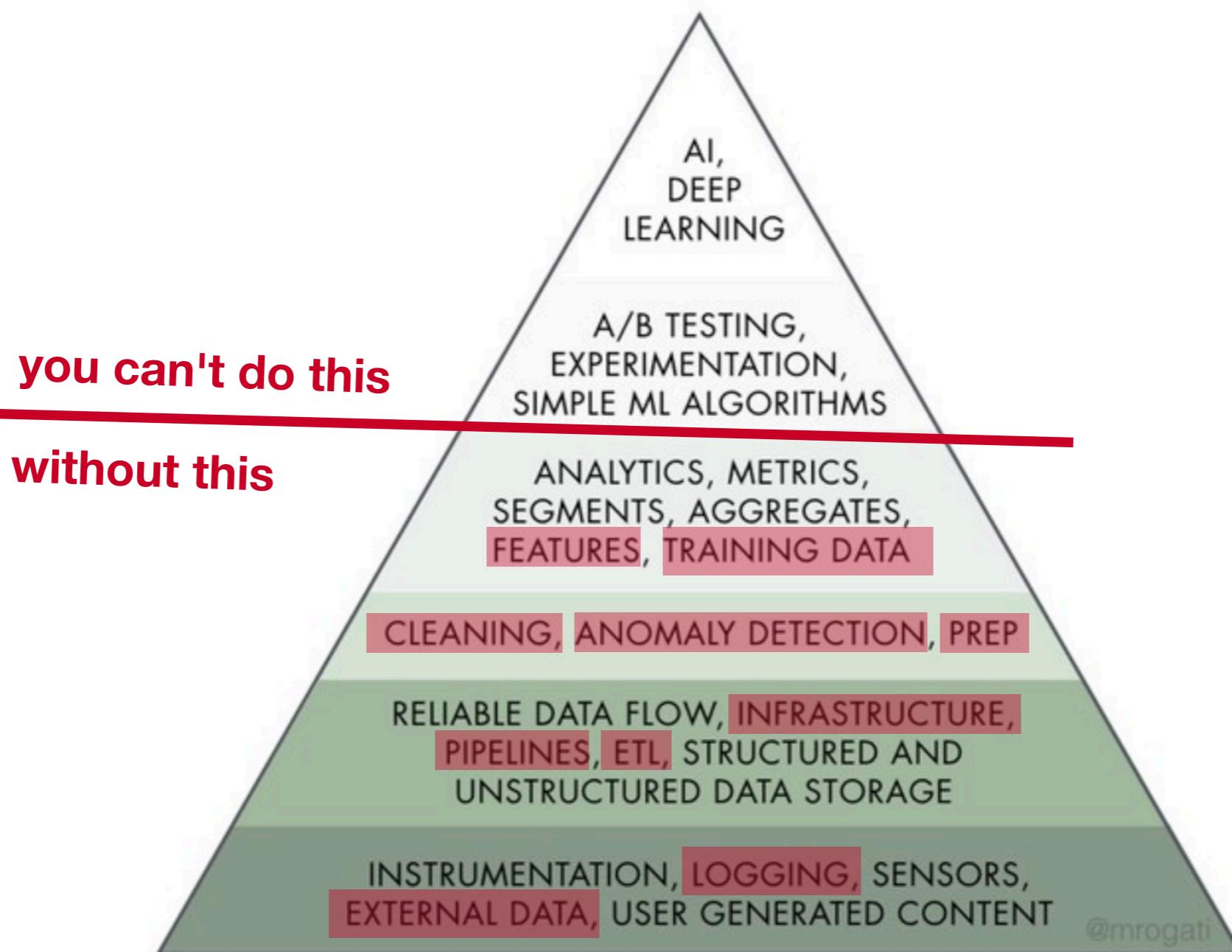
# Data Hierarchy of Needs

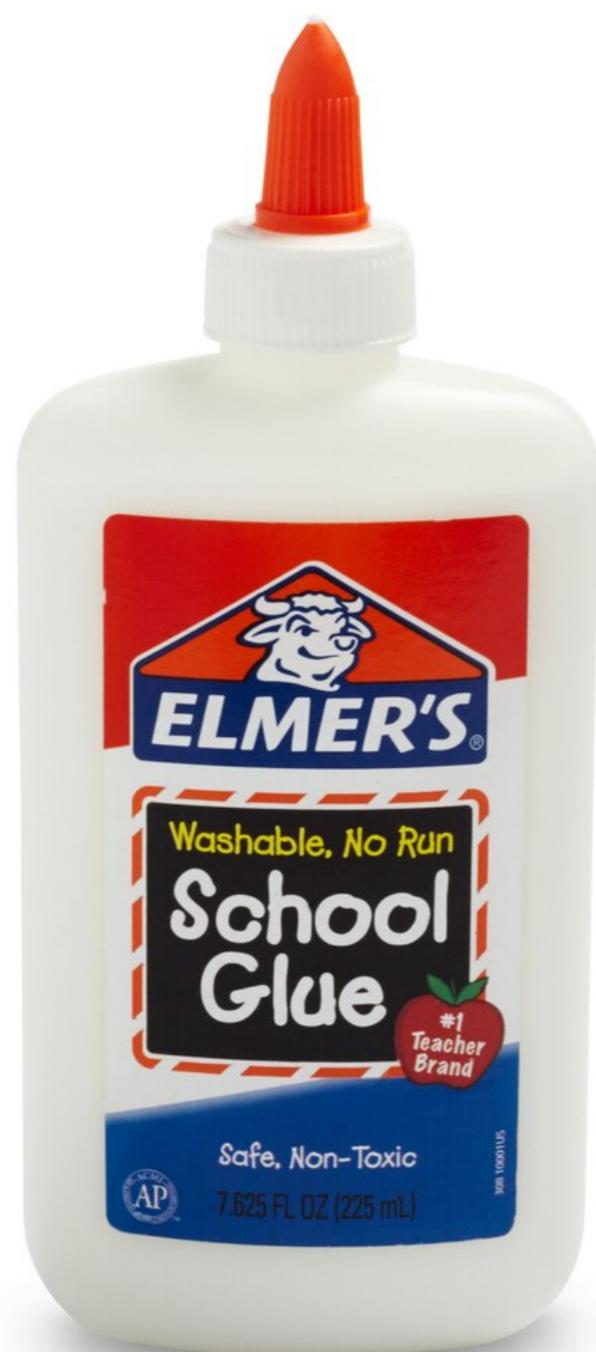


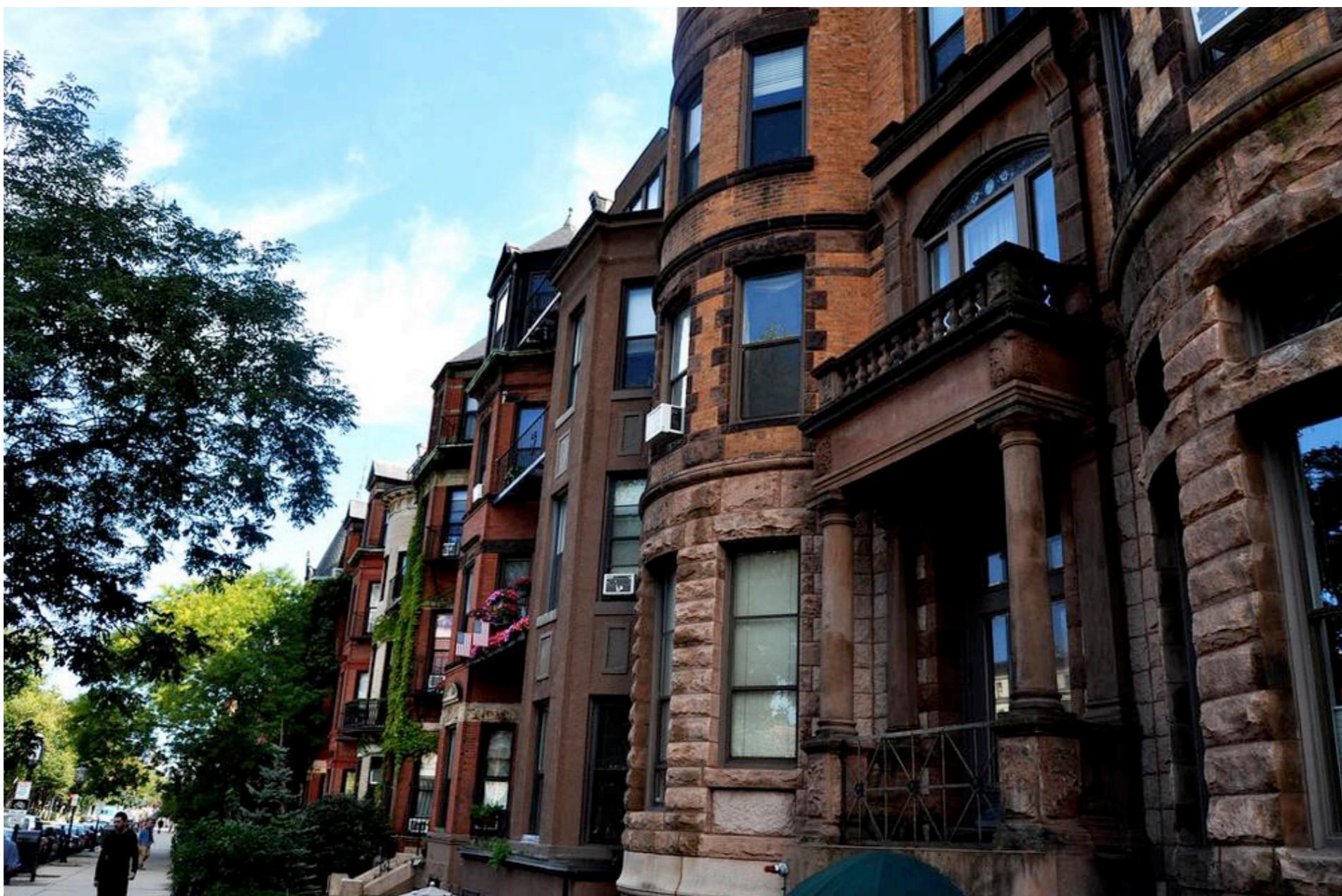
# Data Hierarchy of Needs



# Data Hierarchy of Needs









SPALDING

OFFICIAL GAME BALL

NBA

NBA

NBA



1

Extend



# 01-model.ipynb

Q&A

# Jupyter to Hydrogen

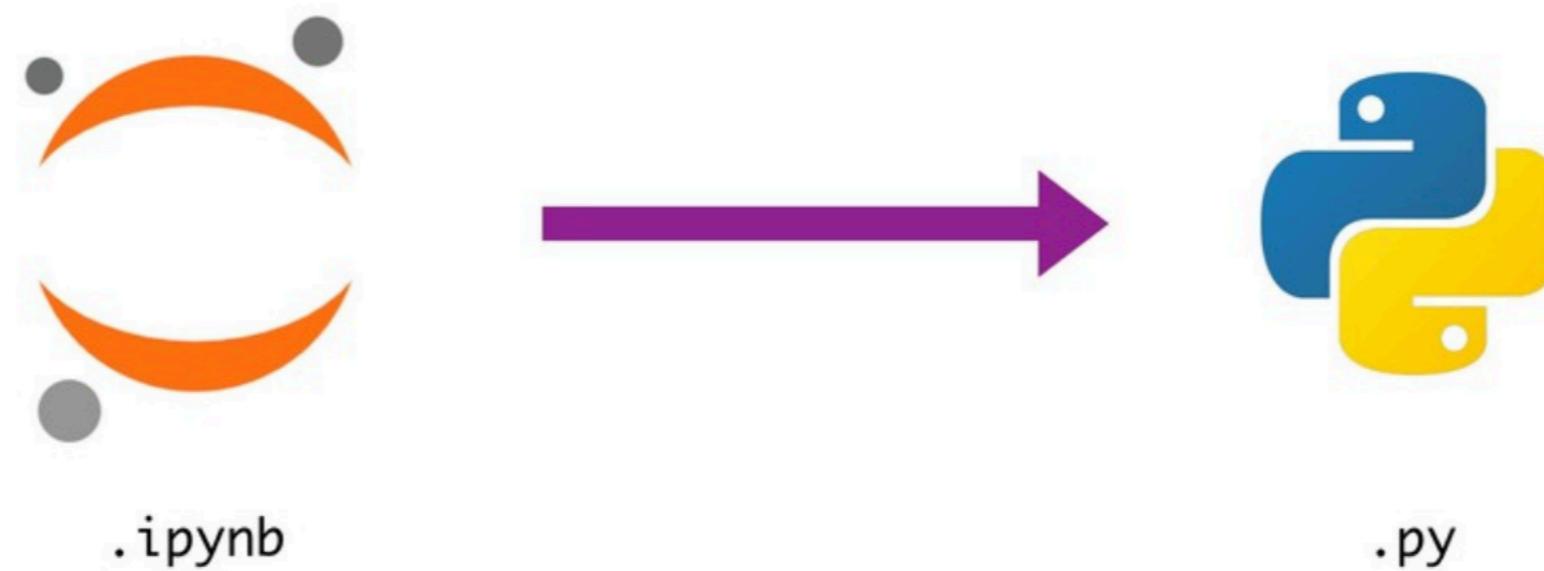


# Jupyter to Hydrogen



- exploratory analysis
- visualizing ideas
- prototyping
  
- messy
- bad at versioning
- not ideal for production

# Jupyter to Hydrogen

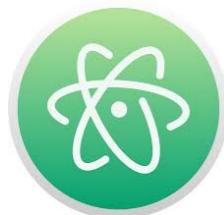


# Hydrogen



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





☰ README.md

# Python Fire

python 2.7 | 3.4 | 3.5 | 3.6 | 3.7 | 3.8 | 3.9

*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`

## Basic Usage

You can call `Fire` on any Python object:  
functions, classes, modules, objects, dictionaries, lists, tuples, etc. They all work!



# 03-predict.py



# 04-fire.py

🔥 python 04-fire.py "LeBron James"

Q&A

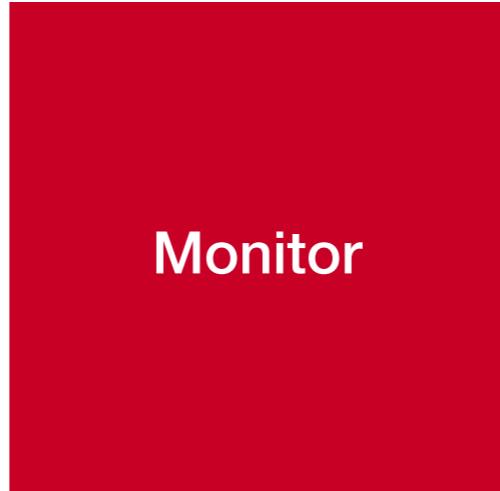
2

Save



# 05-database.py

🔥 python 05-database.py "LeBron James"



Monitor



# ⌚ 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

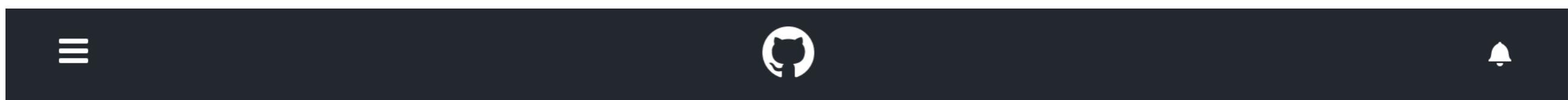
## Send a Message

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

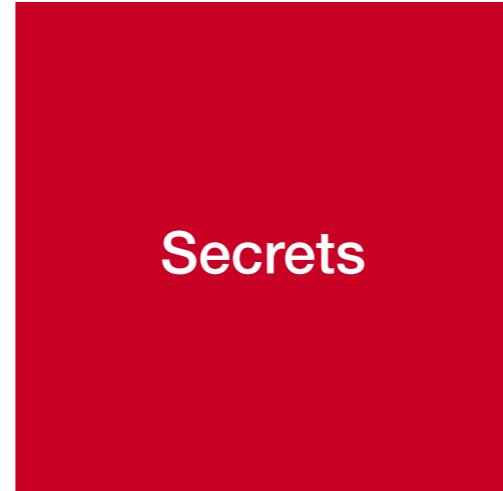
Showing 1 changed file with 2 additions and 2 deletions.

▼ 4 ████ bamazonCustomer.js

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

Line	Change Type	Text
94	+	// }
95	+	// }
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



Apache  
Airflow



Community Meetups Documentation Use cases Announcements Blog Ecosystem

Airflow Summit 2022  
May 23-27 REGISTER NOW

Install

# Apache Airflow

Airflow is a platform created by the community to programmatically author, schedule and monitor workflows.

Install



Version: 2.3.0 ▾

Search docs

**CONTENT**
[Home](#)  
[Project](#)  
[License](#)  
[Quick Start](#)  
[Installation](#)  
[Upgrading from 1.10 to 2](#)
**Tutorial**
[Example Pipeline definition](#)  
[It's a DAG definition file](#)  
[Importing Modules](#)  
[Default Arguments](#)  
[Instantiate a DAG](#)  
[Tasks](#)  
[Templating with Jinja](#)  
[Adding DAG and Tasks documentation](#)  
[Setting up Dependencies](#)  
[Using time zones](#)  
[Recap](#)  
[Testing](#)

## Templating with Jinja

Airflow leverages the power of [Jinja Templating](#) and provides the pipeline author with a set of built-in parameters and macros.

Airflow also provides hooks for the pipeline author to define their own parameters, macros and templates.

This tutorial barely scratches the surface of what you can do with templating in Airflow, but the goal of this section is to let you know this feature exists, get you familiar with double curly brackets, and point to the most common template variable: `{{ ds }}` (today's "date stamp").

airflow/example\_dags/tutorial.py

[source]

```
templated_command = dedent(
    """
    {% for i in range(5) %}
        echo "{{ ds }}"
        echo "{{ macros.ds_add(ds, 7) }}"
    {% endfor %}
    """
)

t3 = BashOperator(
    task_id='templated',
    depends_on_past=False,
    bash_command=templated_command,
)
```

Notice that the `templated_command` contains code logic in `{% %}` blocks, references parameters like `{{ ds }}` function as in `{{ macros.ds_add(ds, 7) }}`.



Suggest a change on this page

Tutorial

Example Pipeline definition

It's a DAG definition file

Importing Modules

Default Arguments

Instantiate a DAG

Tasks

[Templating with Jinja](#)

Adding DAG and Tasks documentation

Setting up Dependencies

Using time zones

Recap

Testing

Running the Script

Command Line Metadata Validation

Pipeline Example



## Templating with Jinja

Airflow leverages the power of [Jinja Template Engine](#) to provide the author with a set of built-in parameters and macros.

Airflow also provides hooks for the pipeline author to define their own parameters, macros and templates.

This tutorial barely scratches the surface of what's possible with templating in Airflow, but the goal of this section is to let you know this feature exists, get you familiar with the basic concepts, and point to the most common template variable: `{{ ds }}` (today's "date stamp").

[airflow/example\\_dags/tutorial.py](#)

[\[source\]](#)

```
templated_command = dsl templated_command = dsl
"""
{% for i in range(5) %}
    echo "{{ ds }}"
    echo "{{ macros.ds_add(ds, 7) }}"
{% endfor %}
"""

t3 = BashOperator(
    task_id='templated',
    depends_on_past=False,
    bash_command=templated_command,
)
```

Notice that the `templated_command` contains code logic in `{% %}` blocks, references parameters like `{{ ds }}`, and uses a function as in `{{ macros.ds_add(ds, 7) }}`.



Suggest a change on this page

Tutorial

[Example Pipeline definition](#)

[It's a DAG definition file](#)

[Importing Modules](#)

[Default Arguments](#)

[Instantiate a DAG](#)

Tasks

[Templating with Jinja](#)

[Adding DAG and Tasks documentation](#)

[Setting up Dependencies](#)

[Using time zones](#)

Recap

Testing

[Running the Script](#)

[Command Line Metadata Validation](#)

Pipeline Example

```
# Airflow needs a home. `~/airflow` is the default, but you can put it anywhere
export AIRFLOW_HOME=`pwd`/airflow

# Install Airflow using the constraints file
AIRFLOW_VERSION=2.3.0
PYTHON_VERSION=$(python --version | cut -d " " -f 2 | cut -d "." -f 1-2)
CONSTRAINT_URL="https://raw.githubusercontent.com/apache/airflow/constraints-${AIRFLOW_VERSION}/
constraints-${PYTHON_VERSION}.txt"
pip install "apache-airflow==${AIRFLOW_VERSION}" --constraint "${CONSTRAINT_URL}"

# The Standalone command will initialise the database, make a user,
# and start all components for you.
airflow standalone

# Visit localhost:8080 in the browser and use the admin account details
# shown on the terminal to login.
# Enable the example_bash_operator dag in the home page
```



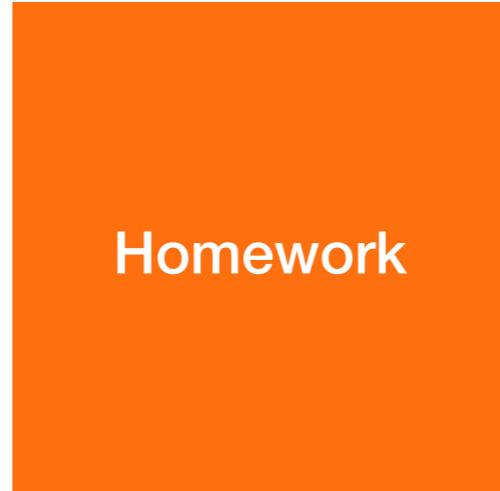


# airflow/dags/basketball.py



airflow tasks test basketball fetch <date>





Homework



**add t3 to save predictions to the db!**

Q&A

*That's all Folks!*