get set up for today's workshop

1. Answer a quick poll before we get started: **bit.ly/dsspoll**

2. Install Python 3.6 (Anaconda) from **https://www.anaconda.com/download**

3. Exercises and files **https://github.com/nmbrodnax/iqss-python-api**

Introduction to Using APIs with Python

NaLette Brodnax

The Institute for Quantitative Social Science

Harvard University

iq.harvard.edu

workshop structure

1	2	3	4
intro	get the tools	review Python	collect data

my goals

- Review Python programming
- Introduce RAPTOR method
- Provide opportunities to practice

your goals

Please complete the one-minute poll at
bit.ly/dsspoll

part 1: introduction

what is an API?

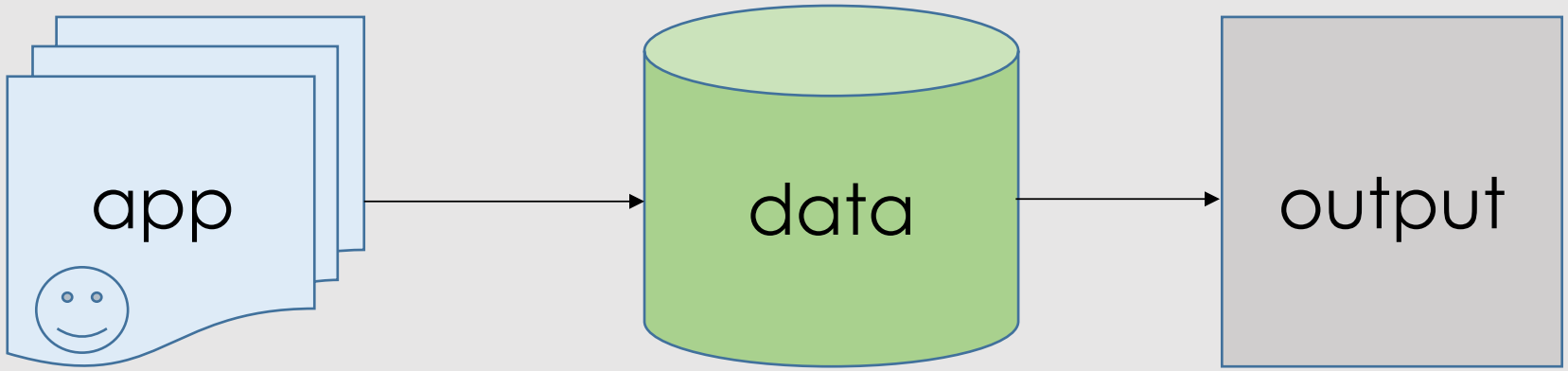
An application programming interface (api) is a tool that allows computers to exchange information.

uses of APIs

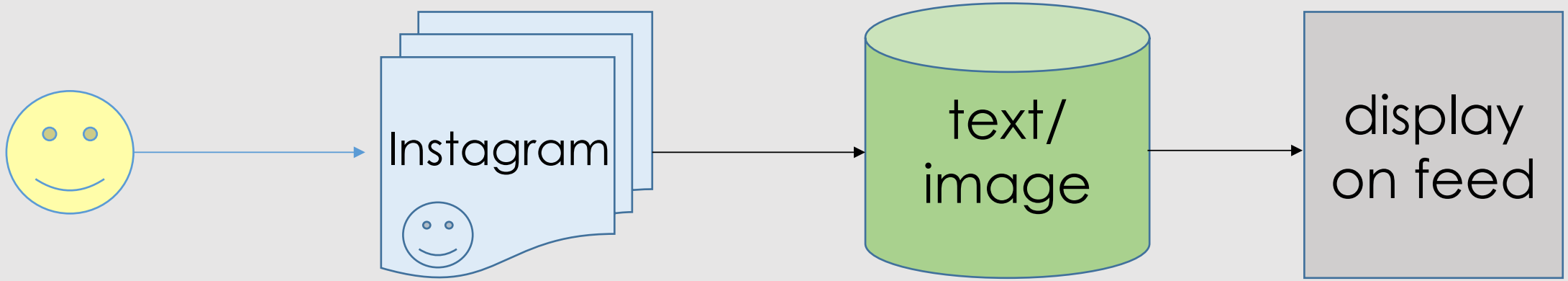
- Social – Twitter, Facebook, etc.
- Internet – bit.ly, domain registration
- Mapping – Google Maps, Bing Maps, etc.
- Search – Google, Yahoo, etc.

APIs make information transferred across the web digestible for a computer.

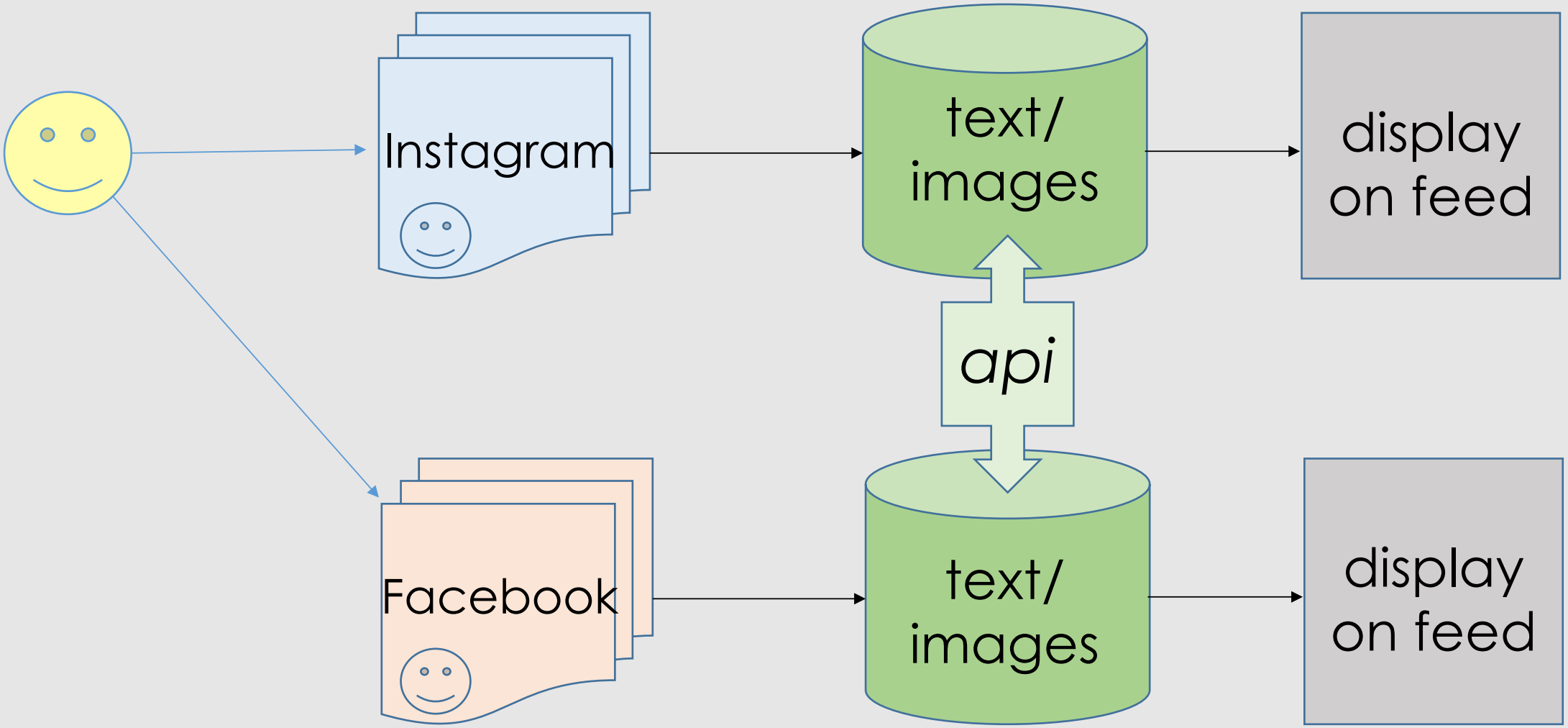
a simple model: application workflow



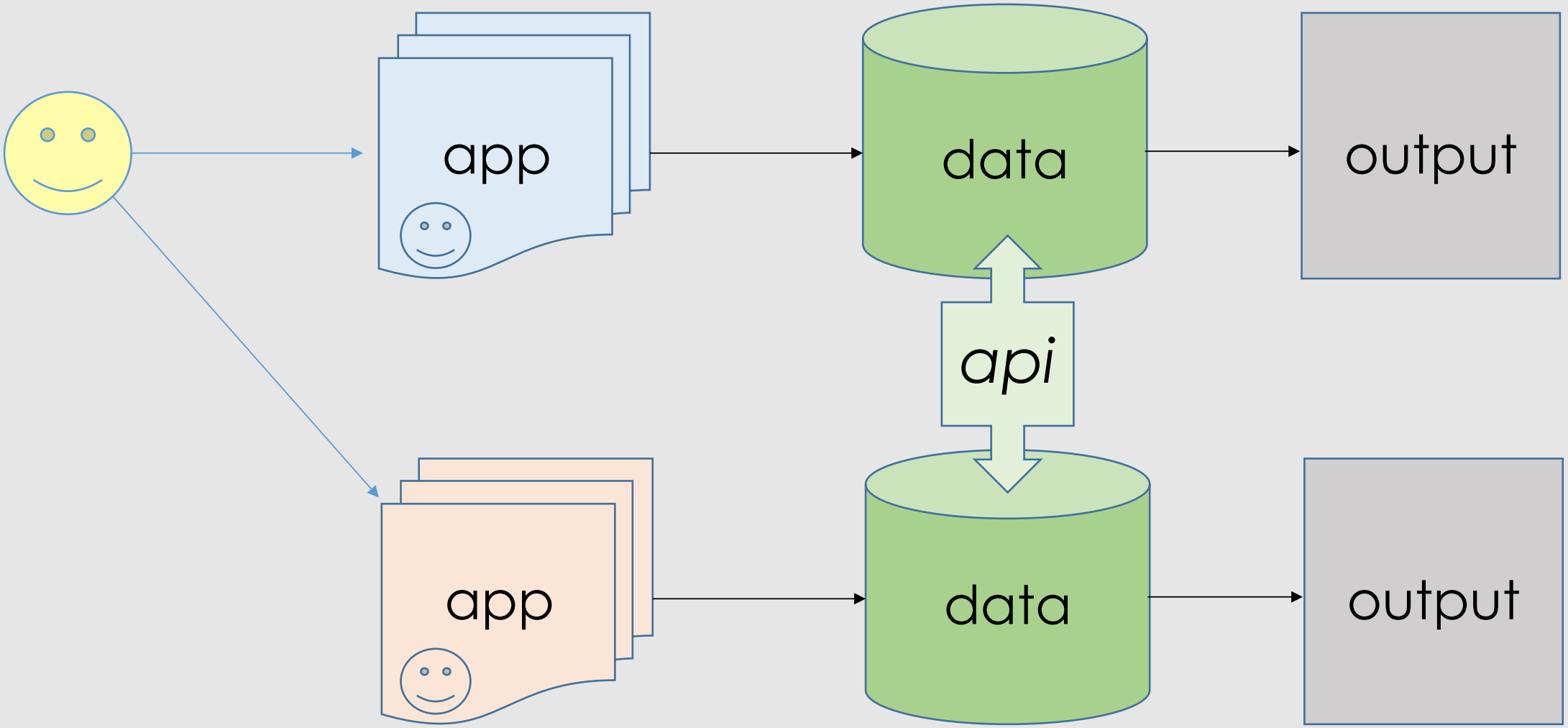
Example:



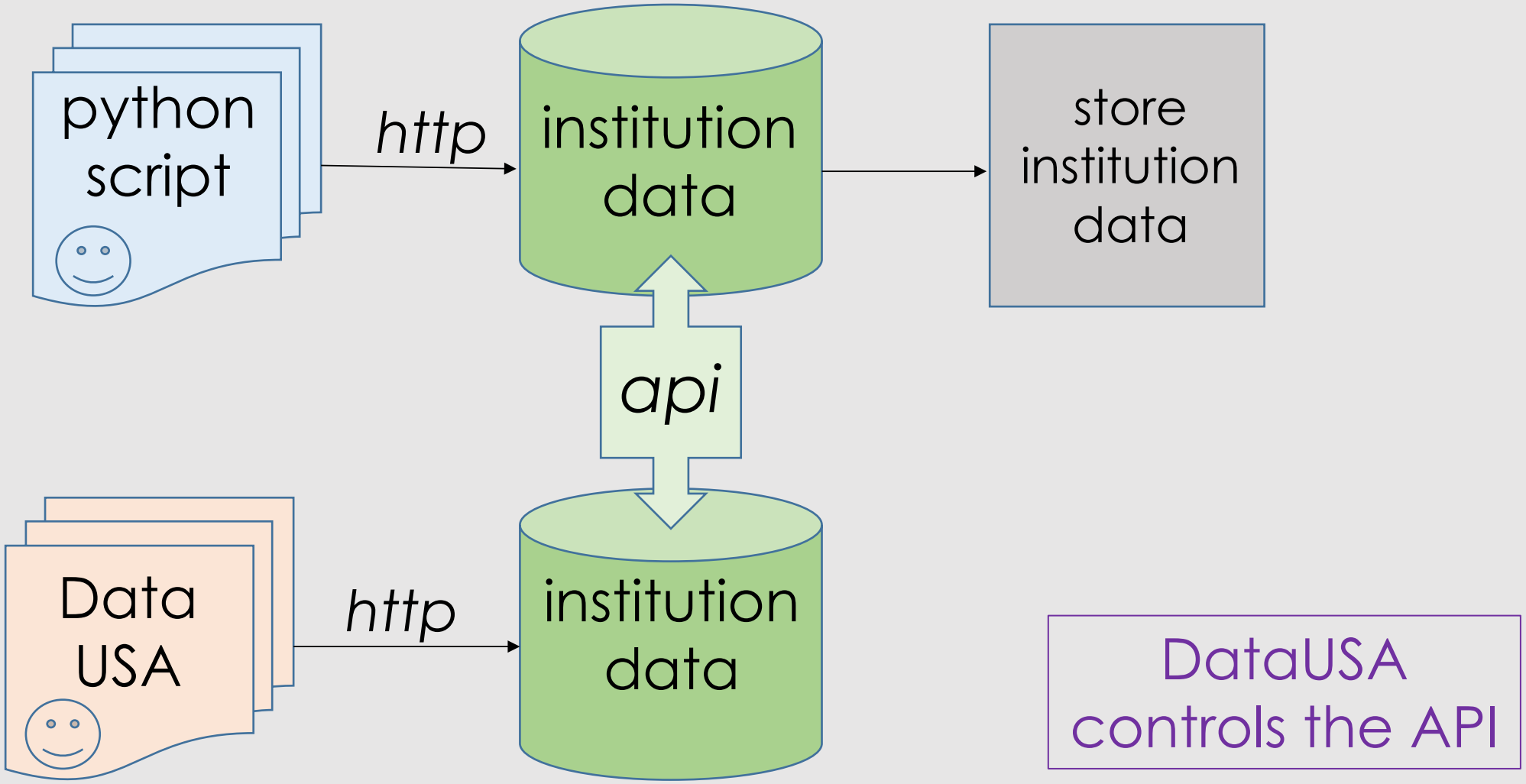
sharing data between applications



goal: mimic data sharing process



example: query institutional data



api features

- **communication** via http
  - GET – retrieve data, e.g., read a Tweet
  - POST or PUT – create or update data, e.g., post a Tweet
  - Response Codes – indicate success or failure
    - 200 OK
    - 401 unauthorized
    - 404 data not found
- standardized **output** – usually JSON
- **authentication** – may require developer registration
- enforced **limitations**

part 2: getting set up

getting the tools

Interpreter → Output

Text Editor + Interpreter → Output

Command Line + Text Editor + Interpreter → Output

Integrated Development Environment (IDE) → Output

installing the tools on your machine

Install Anaconda Python 3.6 from Continuum
https://www.anaconda.com/download

- **Spyder** IDE
- Navigator graphical interface
- Conda command line utility
- Includes popular data science packages

Exercises and files
https://github.com/nmbrodnax/iqss-python-api

not using anaconda?

Download Python 3.6
https://www.python.org/downloads/
→Includes IDLE, an IDE with a text editor and interpreter
→Includes pip, Python's standard package manager

Install the necessary libraries **from the command line**:

```
$ pip3 install --upgrade pip
$ pip3 install requests
```

ways to participate

- **On your machine**: Anaconda3 → Spyder
- **In the lab**: All Programs → Anaconda3 → Spyder

create a new script: *api.py*

```
print(“Hello, world.”)
```

Save the program/script.
Run the program in the IDE: **F5** or **Run** or 

part 3: Python review

programming in Python

1. **sequences**
2. **control structures**
3. **writing functions**
4. **using modules and packages**

data types: sequences

String—ordered sequence of characters

```
‘happy’
```

List—ordered sequence of items

```
[‘Leia’, ‘Rey’, ‘Maz’]
```

Dictionary—unordered sequence of key-value pairs

```
{‘name’:‘Kylo’, ‘side’:‘dark’}
```

Exercise 3.1

working with sequences

- Sequences bound by different characters
  - string ‘’
  - list **[]**
  - dictionary **{}**
- Reference items in an **ordered** sequence by number, *starting from 0 or ending at -1*
- Reference dictionary items by key

loops

control structures that allow repeated behavior within a program

- **for** – repeats commands for a finite number of iterations
- **while** – evaluates a conditional statement and repeats commands as long as the condition is True

functions v. methods

function-–named block of code that can accept any number of arguments

```
my_string = ‘aBcDe’
print(my_string)
```

method—a function with a built-in parameter for the object being acted on

```
print(my_string.lower())
```

writing functions

def function_name(argument1, argument2, ...):
    first command
    second command
    return output

```
def say_hello(name_string):
    print('Hello, ' + str(name_string) + '!')
    return None

say_hello('NaLette')
```

modules

import statements allow you to add functions

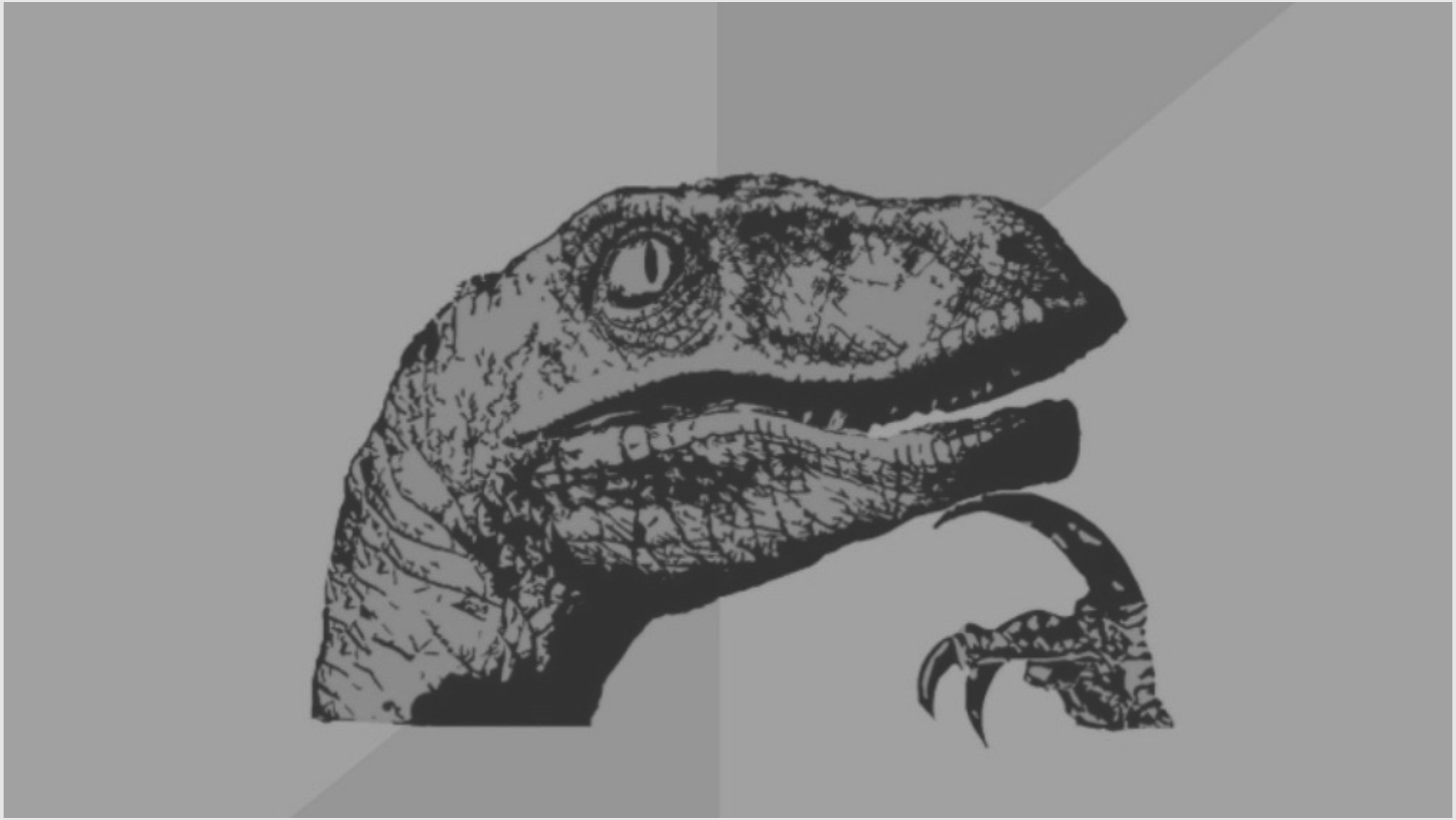
```
import csv

with open("workshop.csv", 'w') as csvfile, \
     open(filename, 'r') as txtfile:
    writer = csv.writer(csvfile)
```

use module name to call functions

part 4: api programming

files: https://github.com/nmbrodnax/iqss-python-api



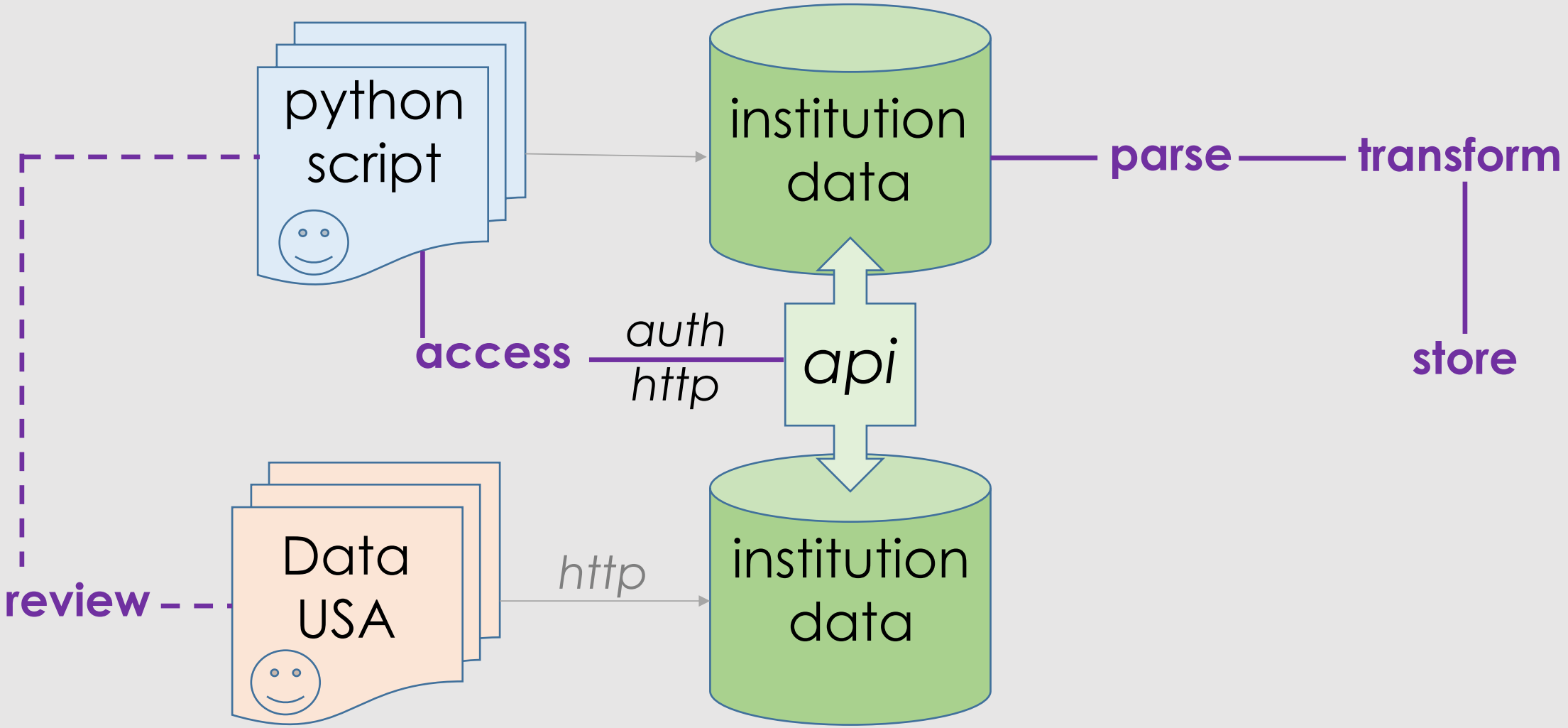
Review • **A**ccess • **P**arse • **T**ransform • st**OR**e

RAPTOR

Review	API documentation (host, parameters, etc.)
Access*	Registration, authentication, and limits
Parse	Data in JSON or XML format
Transform	Convert nested dictionary to flat file
St**OR**e	Write to text, CSV, or other file format

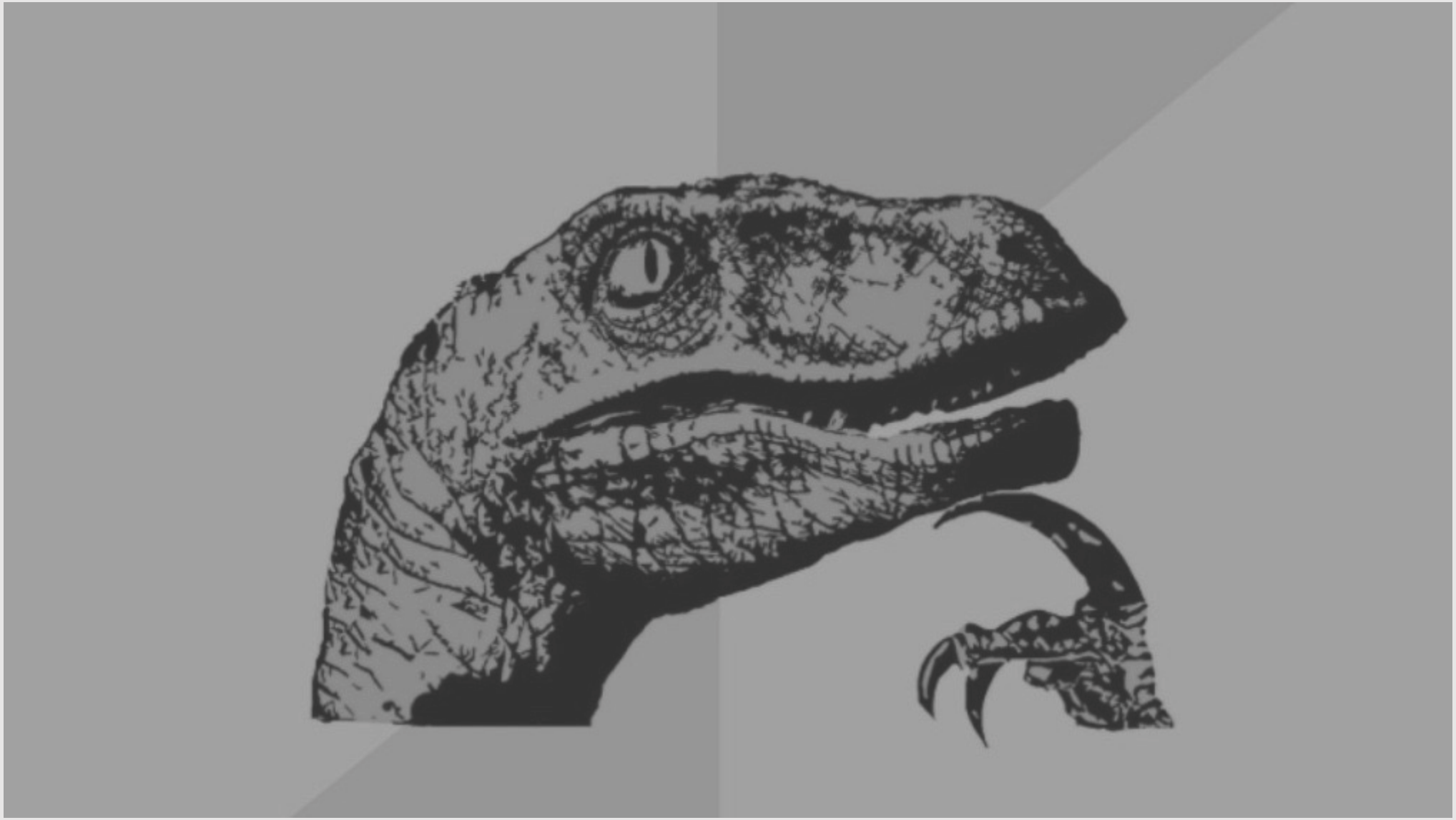
*Exercise caution with tools that automate this process, called “wrappers”

script workflow



take a 10-minute break

https://github.com/nmbrodnax/iqss-python-api



Review • **A**ccess • **P**arse • **T**ransform • st**OR**e

review

Data: college degrees awarded to men and women by field

https://github.com/DataUSA/datausa-api

api request elements

- **Host:** API address https://github.com/DataUSA/datausa-api
- **Required:** Classification of Instructional Programs https://nces.ed.gov/ipeds/cipcode/
- **Optional**: Year and Institution https://nces.ed.gov/ipeds/datacenter/InstitutionByName.aspx
- **Columns**: grads_total, grads_men, grads_women

review

```
import csv
import json
import requests


host = 'http://api.datausa.io/api/'
```

https://github.com/DataUSA/datausa-api

access

```
# build api GET request

required_params = "?show=cip&sumlevel=2”

optional_params = "&year=2015&university=166027”

columns = "&required=grads_total,grads_men,grads_women”
```

access

```
url = host + required_params + optional_params + columns

# check the HTTP response code

response = requests.request('GET', url)

print(response)
```

check the response code (see https://httpstatuses.com/)

our GET request

```
http://api.datausa.io/api/?show=cip&sumlevel=2&year=2015&university=166027&required=grads_total,grads_men,grads_women
```

you can enter your GET request string into a browser

parse

convert JSON data to a Python object

```
degrees = response.json()
```

"pretty print" to see the nested structure

```
print(json.dumps(degrees, indent = 4, sort_keys = True))
```

transform

create a set of dictionaries

```
data = [dict(zip(degrees["headers"], d)) for d in degrees["data"]]
```

store

```
fieldnames = degrees["headers"]

# save data to a csv file
with open("ipeds_degrees_cip.csv", "w") as csvfile:
    writer = csv.DictWriter(csvfile, fieldnames)
    writer.writeheader()
    for row in data:
        writer.writerow(row)
```

run your api script

From IDE:
Run the program in the interpreter: **F5** or **Run** or 

OR

From the Command Line:

```
$ python3 ipeds_api.py
```

Questions?

Please complete the brief evaluation survey:

bit.ly/dssevaluation

linkedin: nalettebrodnax

github: nmbrodnax

twitter: @nbrodnax

email: nbrodnax@iq.harvard.edu

web: www.nalettebrodnax.com