Introduction to Flask

Data Boot Camp

Class Objectives

By the end of today's class you will be able to:

- Analyzing databases using the SQLAlchemy ORM.
- Join tables in SQLAlchemy
- Python Datetime + SQLAlchemy
- Create and run a Flask server.

Activity: Chinook Database Analysis

In this activity, you will practice analyzing databases using the SQLAlchemy ORM.

(Instructions sent via Slack.)

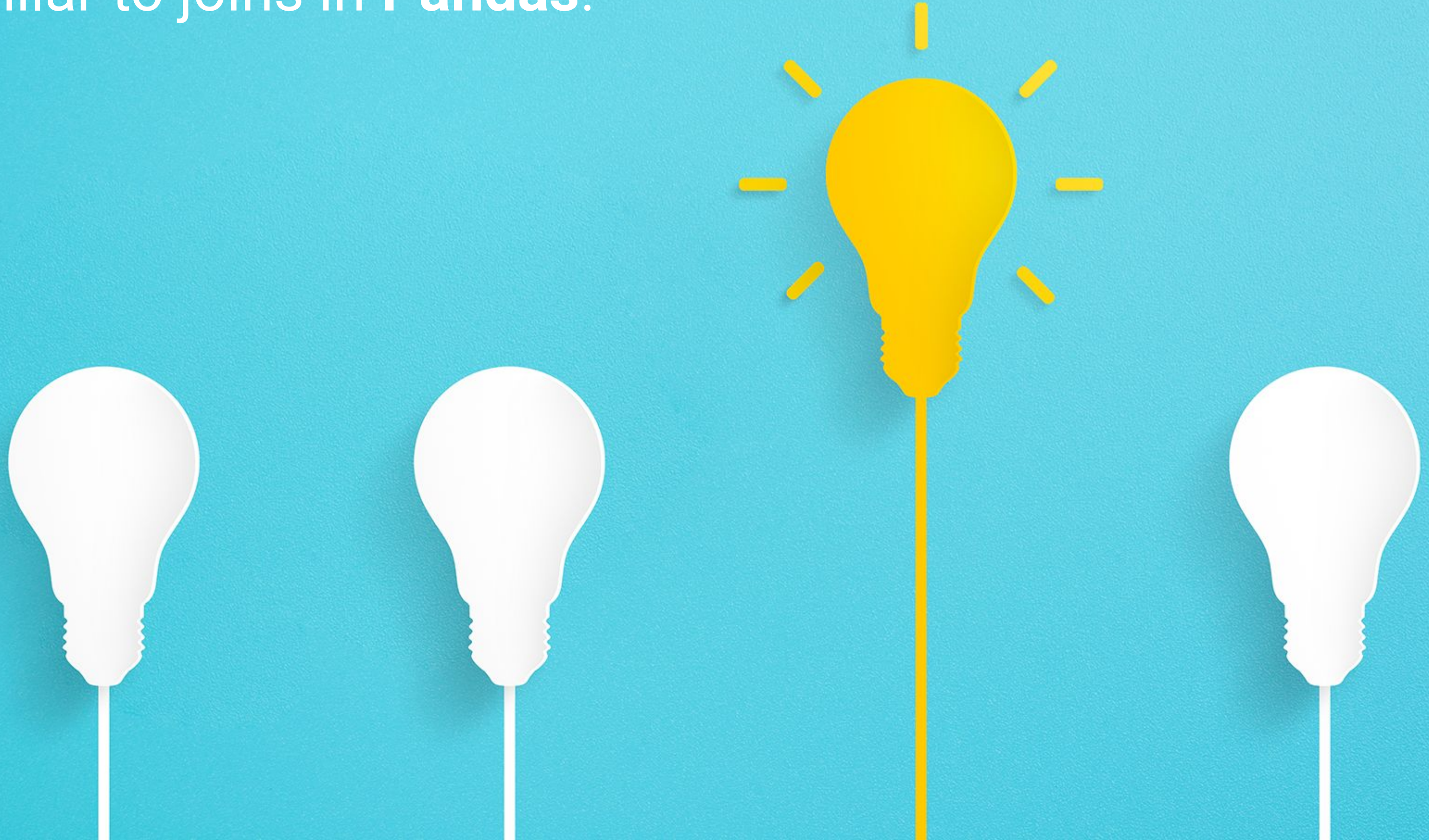
Suggested Time:
25 Minutes

Chinook Database Analysis Instructions

- Create a SQLAlchemy engine to the database chinook.sqlite.
- Design a query that lists all of the billing countries found in the invoices table.
- Design a query that lists the invoices totals for each billing country and sort the output in descending order.
- Design a query that lists all of the Billing Postal Codes for the USA.
- Calculate the invoice items totals sum (UnitPrice * Quantity) for each Billing Postal Code for the USA.

Time’s Up! Let’s Review.

Joins in **SQLAlchemy** are very similar to joins in **Pandas**!



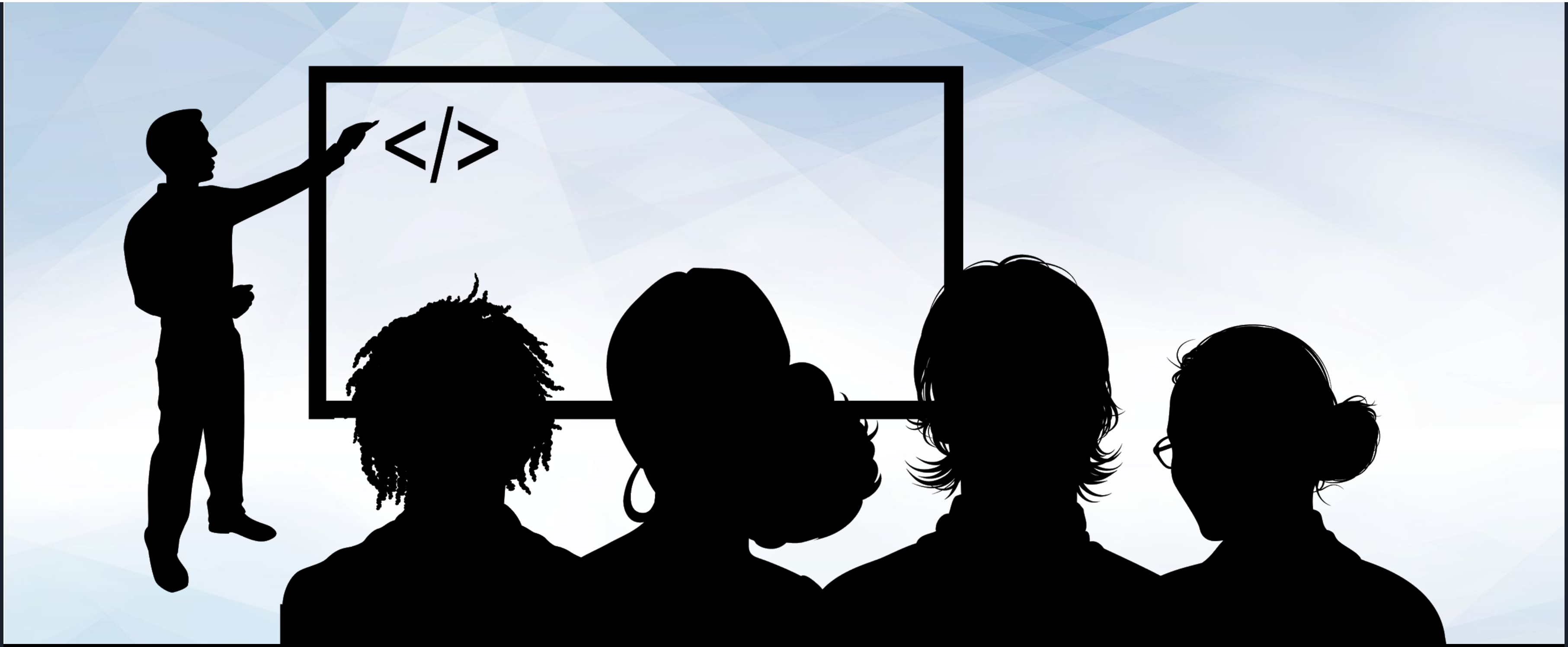
SQLAlchemy Joining Tables Step-By-Step

01 Use inspect(engine).get_table_names() to find table names in the database

02 Use inspect(engine).get_columns(table) to get the column names

03 Create a list of all table columns you wish to keep

04 Use .filter() to describe what columns to join on



Instructor Demonstration

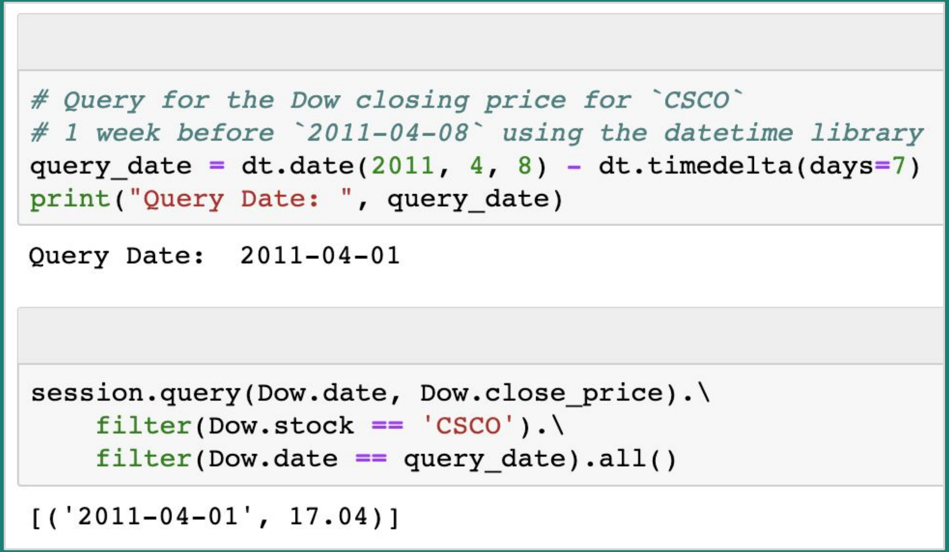
Joins

Times and dates are bit trickier than integers or decimals

- Throughout all programming
- In some cases we may need to do conversions to add or subtract time
  - Days, months, years to seconds
  - Then convert everything back!
- Many ways to annotate a date
  - 10/21/2020
  - 21/10/2020
  - 21Oct2020
  - October 21, 2020
- Python libraries like `datetime` makes things easier!

Datetime and SQLAlchemy work well together!

- Dates and times can be stored in many ways
  - Datetime objects
  - Strings
  - Integers (number of seconds)
- It could be difficult to compare, or query for a specific date/time
- Python's `datetime` library helps make dates and times easier

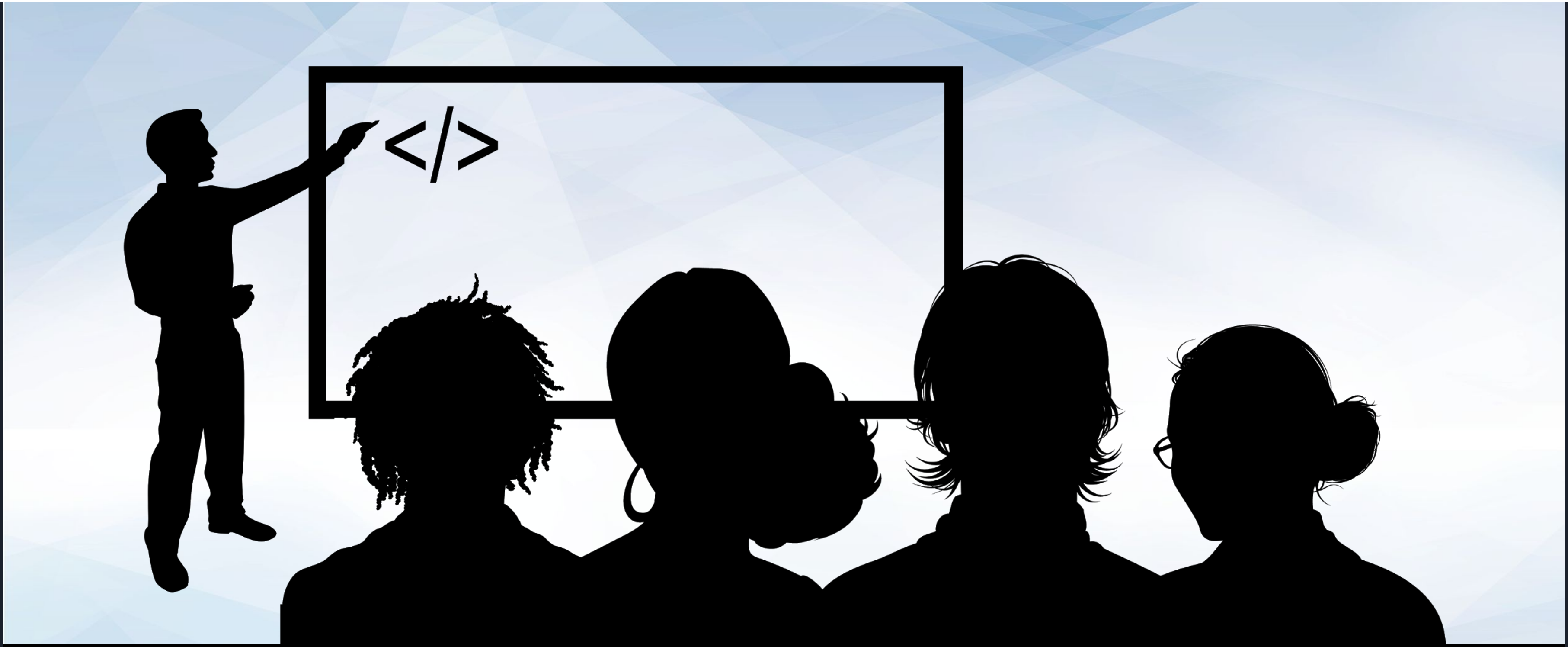


```
# Query for the Dow closing price for `CSCO`
# 1 week before `2011-04-08` using the datetime library
query_date = dt.date(2011, 4, 8) - dt.timedelta(days=7)
print("Query Date: ", query_date)
```

```
Query Date:  2011-04-01
```

```
session.query(Dow.date, Dow.close_price).\
    filter(Dow.stock == 'CSCO').\
    filter(Dow.date == query_date).all()
```

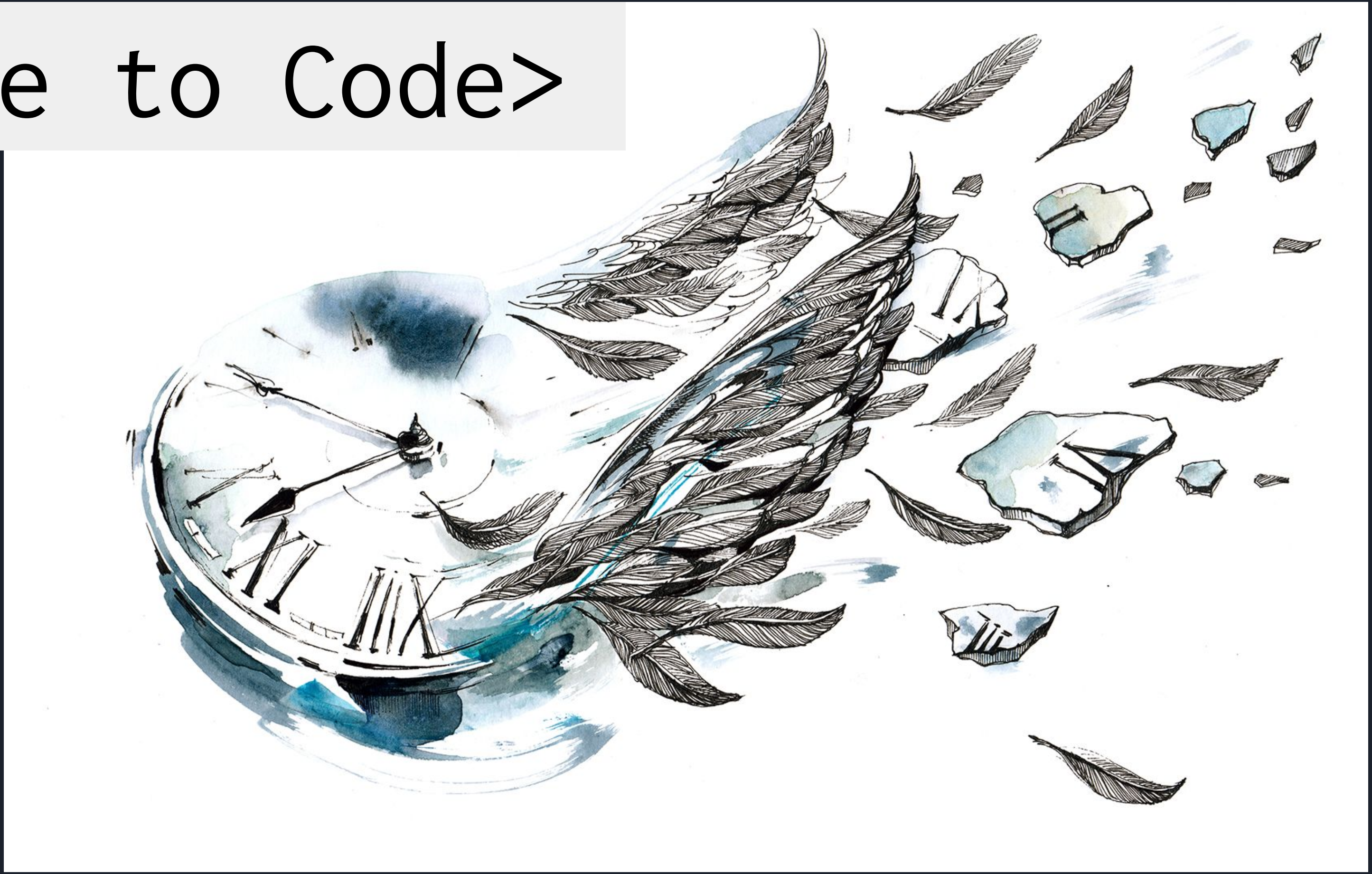
```
[('2011-04-01', 17.04)]
```



Instructor Demonstration

Dates

<Time to Code>



Activity: Dates

In this activity, you will practice working with dates, both in SQLAlchemy and with the `datetime` library.

(Instructions sent via Slack.)

Suggested Time:
20 Minutes

Dates Instructions

- Use the dow.sqlite dataset provided to analyze the average stock prices (average open, average high, average low, average close) for all stocks in the Month of May

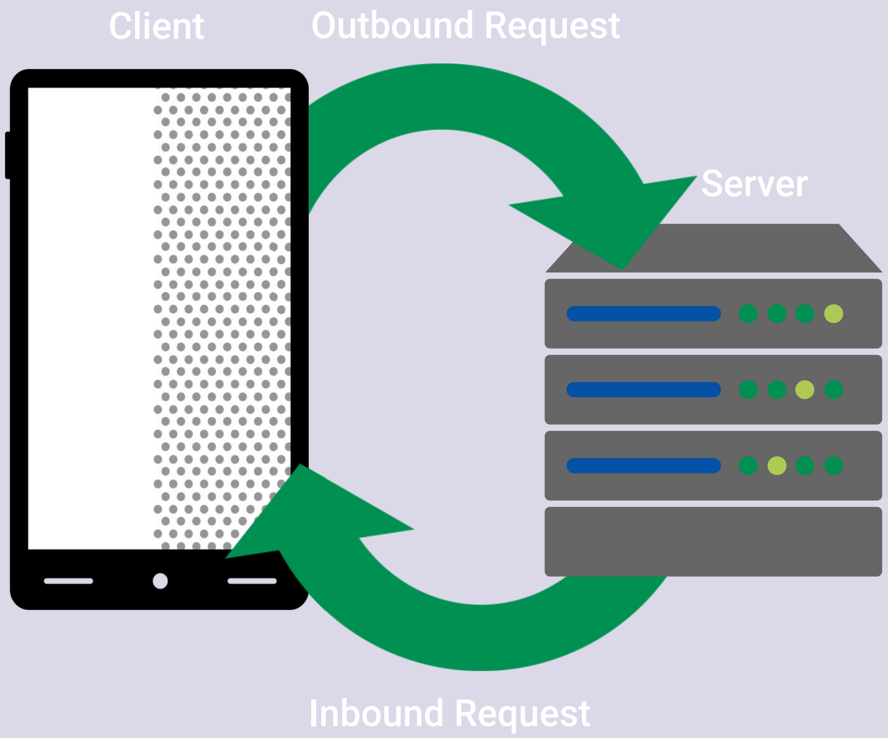
- Plot the results as a Pandas or Matplotlib Bar Chart

Bonus:

- Calculate the high-low peak-to-peak (PTP) values for IBM stock after 2011-05-31.

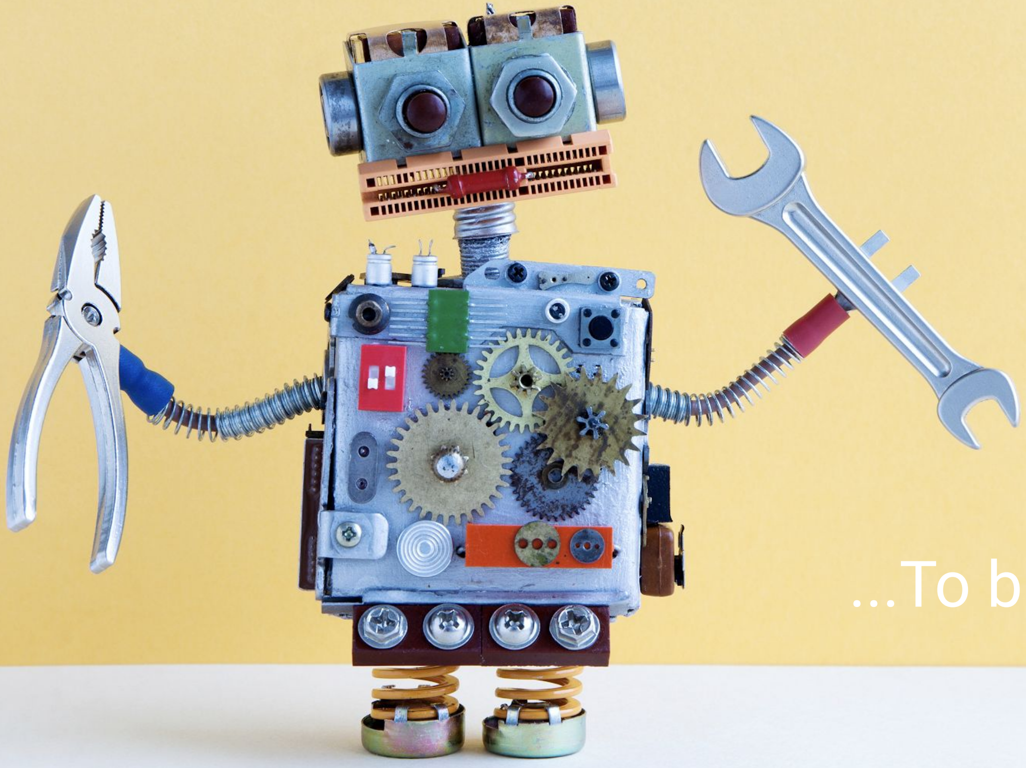
Time’s Up! Let’s Review.

Internet is Built from Clients and Servers

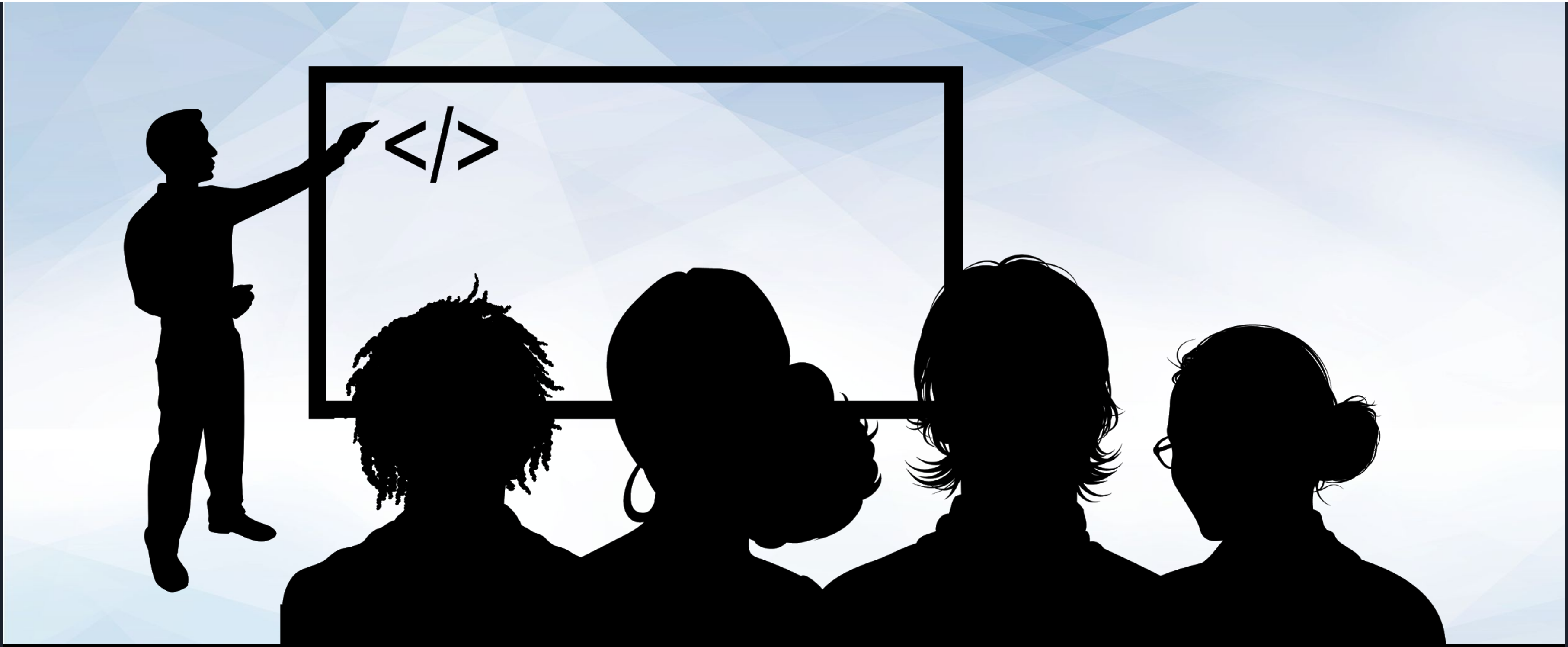


- Whatever application or device that is asking for information is called a “client”
  - A browser makes request on behalf of a user
- A “server” is a process running on a remote machine listening for requests
  - A server is essentially a *program*
- We can write the code that runs a server
  - We can determine what data is displayed
  - We can determine what data is shared

Flask is a micro web framework...



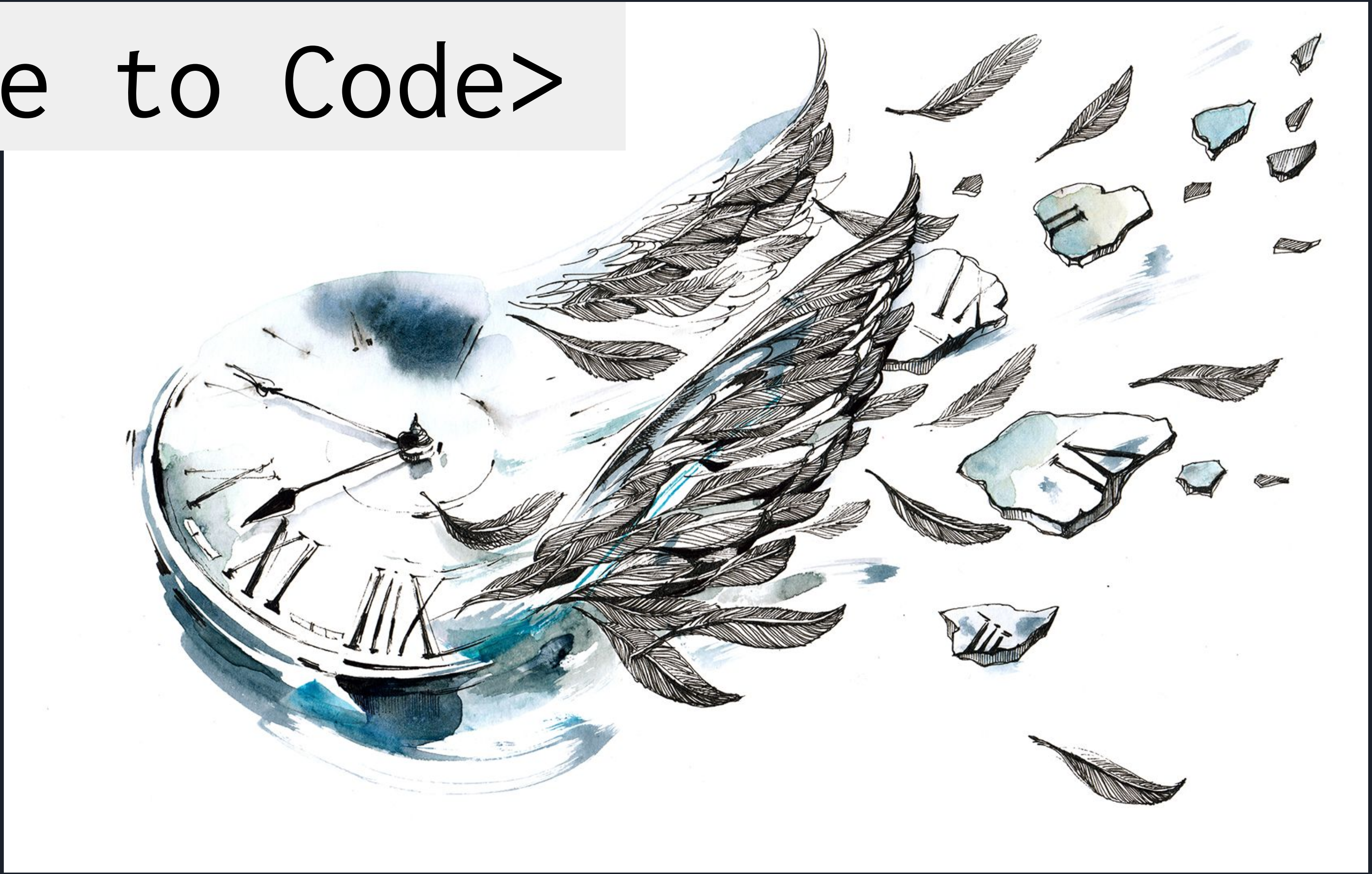
...To build **your** own APIs!



Instructor Demonstration

Introduction to Flask

<Time to Code>



Activity: Hello, Web

In this activity, you will create your first Flask server with a few endpoints.

(Instructions sent via Slack.)

Suggested Time:
20 Minutes

Hello, Web Instructions

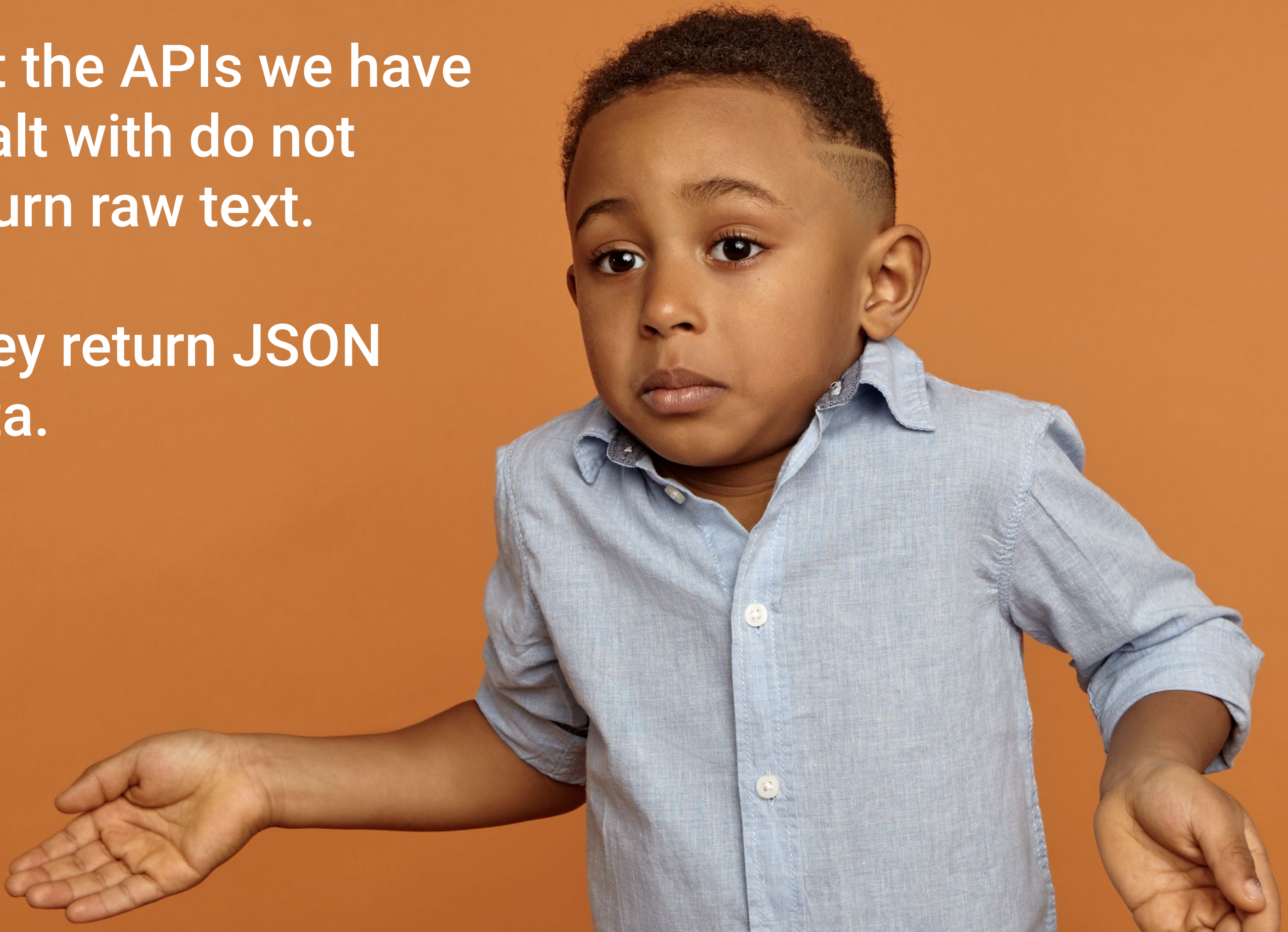
- Create an `app.py`, and make the necessary imports.
- Use Flask to create an `app` instance.
- Use route decorators to define the endpoints described in the `README.md`
- Finally, add code at the bottom of the file that allows you to run the server from the command line with: `python app.py`.

Time’s Up! Let’s Review.

All routes so far have returned *string* responses.

But the APIs we have dealt with do not return raw text.

They return JSON data.

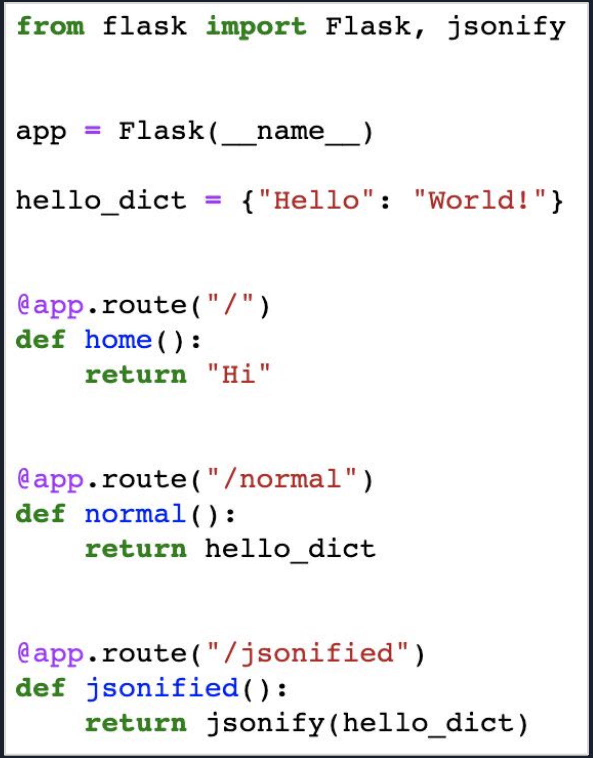


Flask has a function to create JSON responses

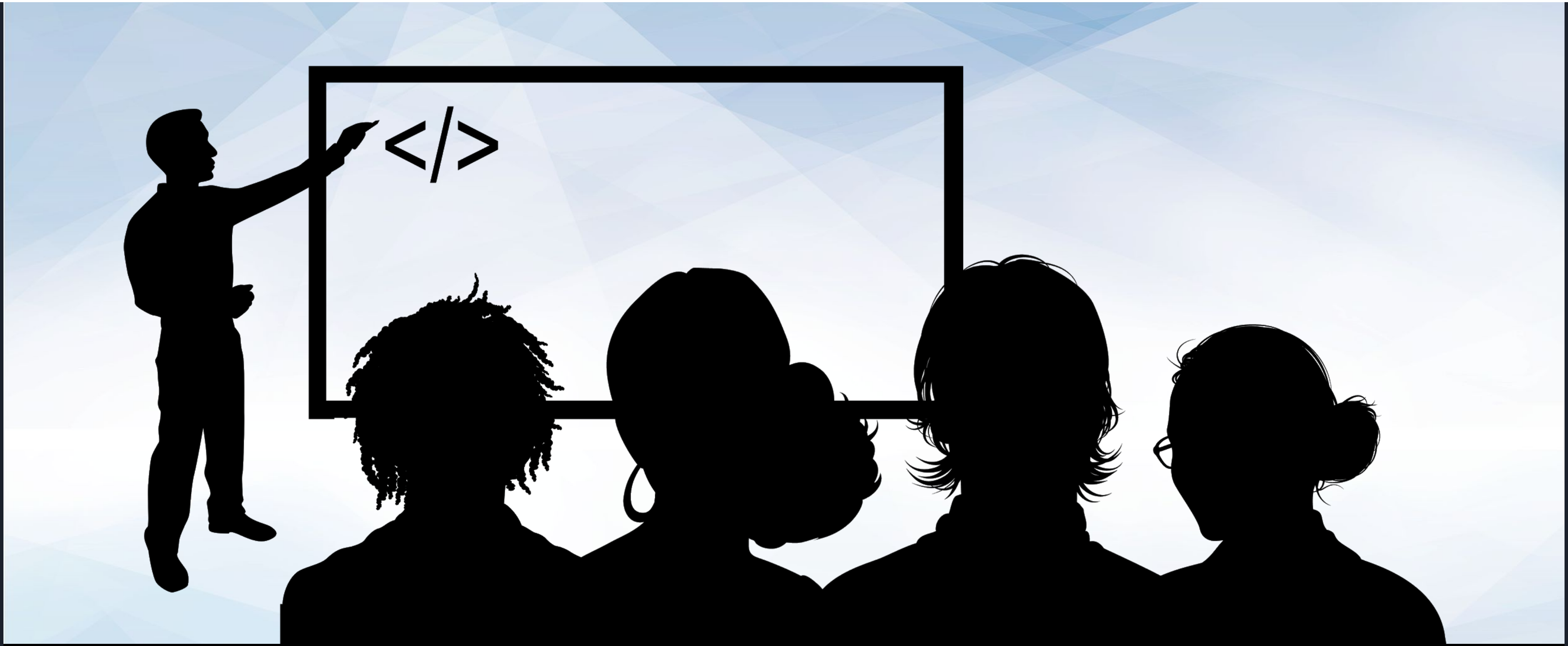
- We cannot simply return a dictionary response directly through Python
    - Routes must return HTTP responses

`jsonify`

    - The converted JSON responses are wrapped in HTTP to send back to the client



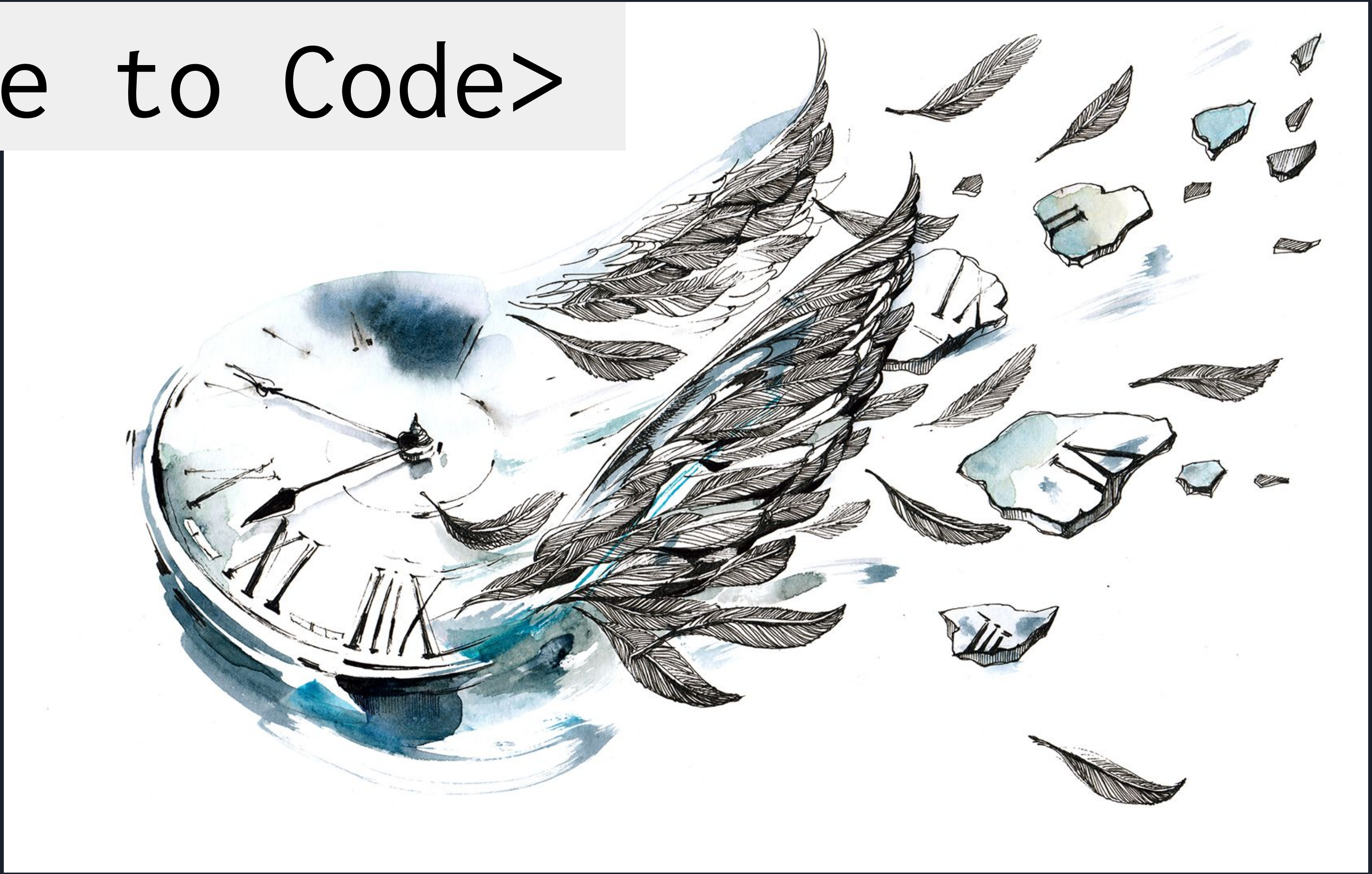
```python
from flask import Flask, jsonify


app = Flask(__name__)

hello_dict = {"Hello": "World!"}


@app.route("/")
def home():
    return "Hi"


@app.route("/normal")
def normal():
    return hello_dict


@app.route("/jsonified")
def jsonified():
    return jsonify(hello_dict)
```



Instructor Demonstration

JSON APIs with jsonify

<Time to Code>



Activity: Justice League

In this activity, you will create a server that sends welcome text at one endpoint, and JSON data at another endpoint.

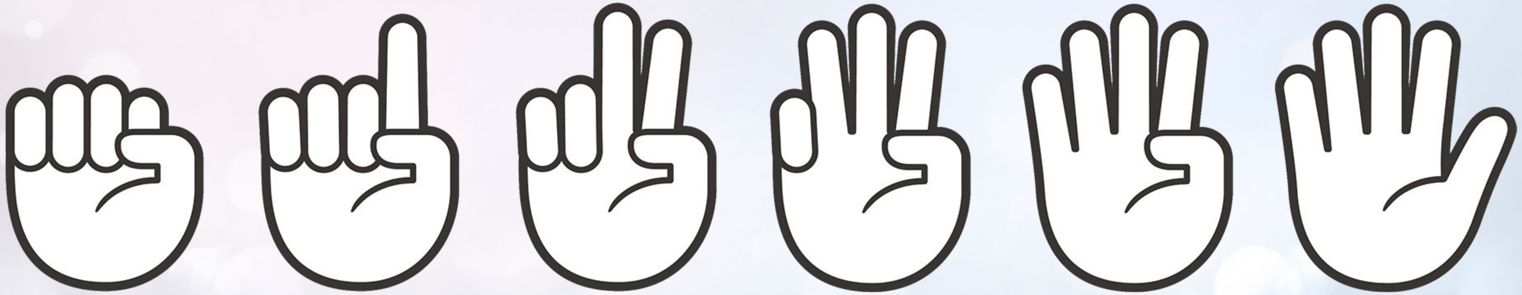
(Instructions sent via Slack.)

Suggested Time:
20 Minutes

Justice League Instructions

- Create a file called app.py for your Flask app.
- Define a Python dictionary containing the superhero name and real name for each member of the DC Comics Justice League
- Create a **GET** route called /api/v1.0/justice-league.
- Define a root route / that will return the usage statement for your API.

Time’s Up! Let’s Review.



FIST TO FIVE: