Unit 6 Day 2 - Python APIs

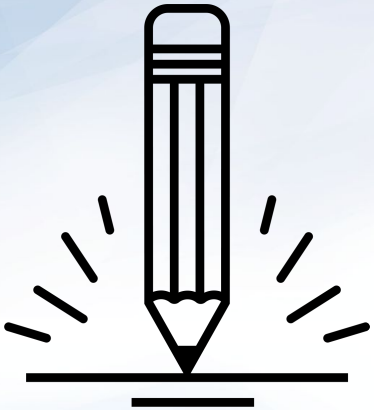
Data Boot Camp

Lesson 6.2

Class Objectives

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

- Load JSONs into pandas DataFrames
- Use `try` and `except` to resolve missing key values without terminating the code
- Use linear regression to predict temperature at certain latitudes



Activity: JSON Traversal

In this activity, you will be traversing a JSON file using your knowledge of Python..

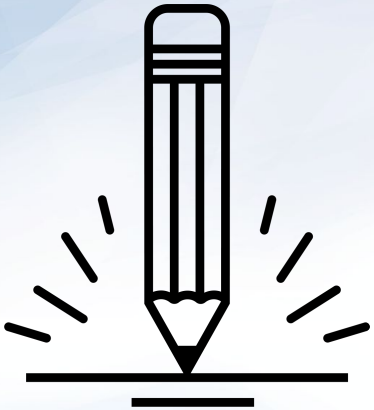
(Instructions sent via Slack.)

Suggested Time:
10 Minutes

JSON Traversal Instructions

- Load the provided JSON.
- Retrieve the video's title.
- Retrieve the video's rating.
- Retrieve the link to the video's first tag.
- Retrieve the number of views this video has.

Time’s Up! Let’s Review.



Activity: Requests Review

In this activity, you will be making a request to remote JSON data and printing out data from the response.

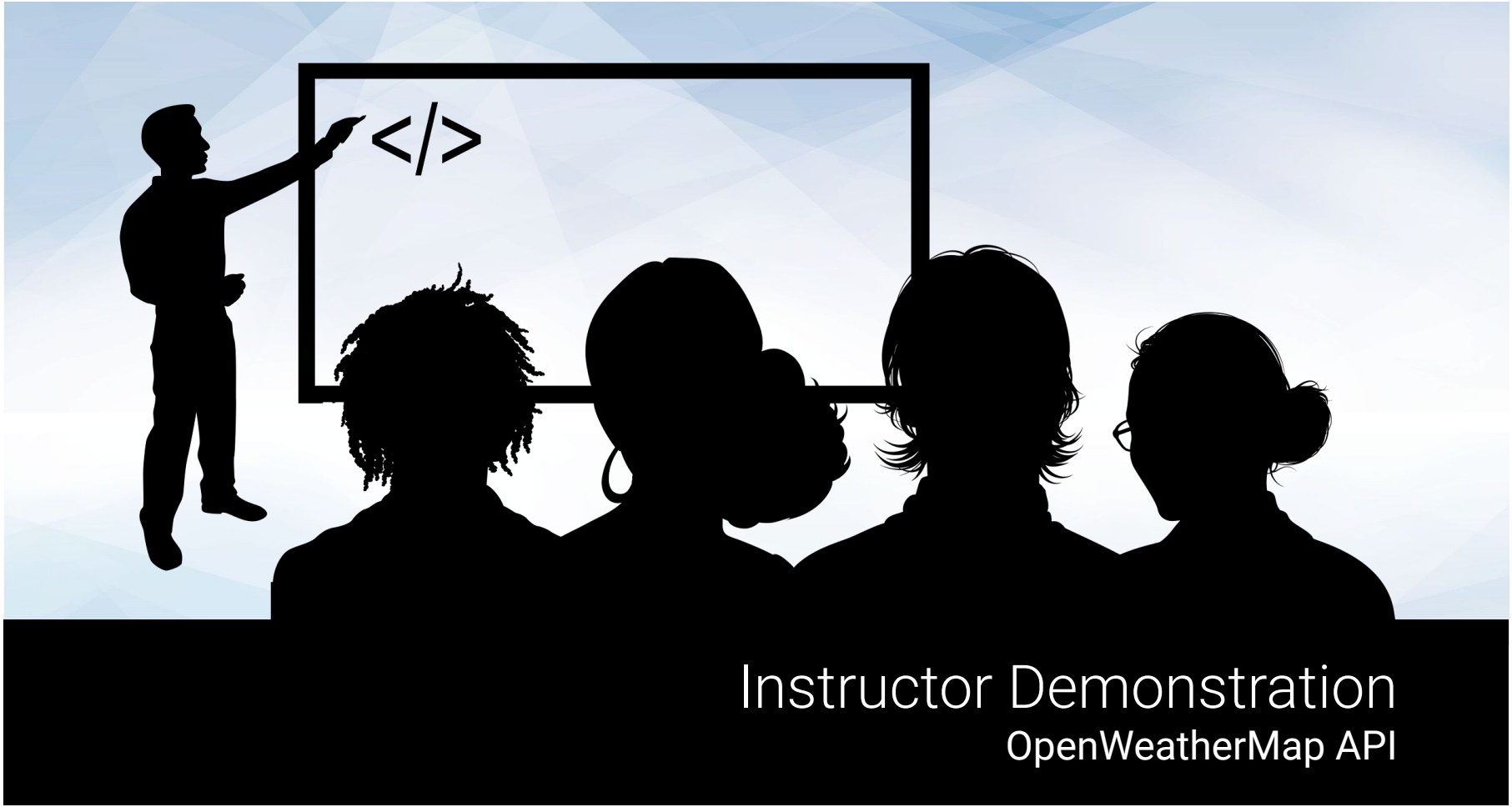
(Instructions sent via Slack.)

Suggested Time:
15 Minutes

Request Review Instructions

- Make a request to the following endpoint (https://2u-data-curriculum-team.s3.amazonaws.com/dataviz-classroom/v1.1/06-Python-APIs/request_review.json), and store the response.
- JSON-ify the response.
- Print the JSON representations of the first and last posts.
- Print number of posts received.

Time’s Up! Let’s Review.

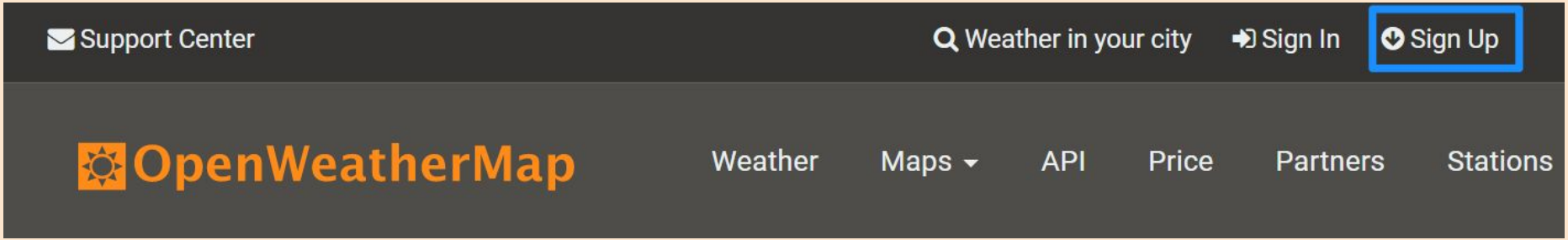


Instructor Demonstration

OpenWeatherMap API

OpenWeatherMap API

- Provides various sorts of meteorological data.
- Sign up for a key at https://home.openweathermap.org/users/sign_up

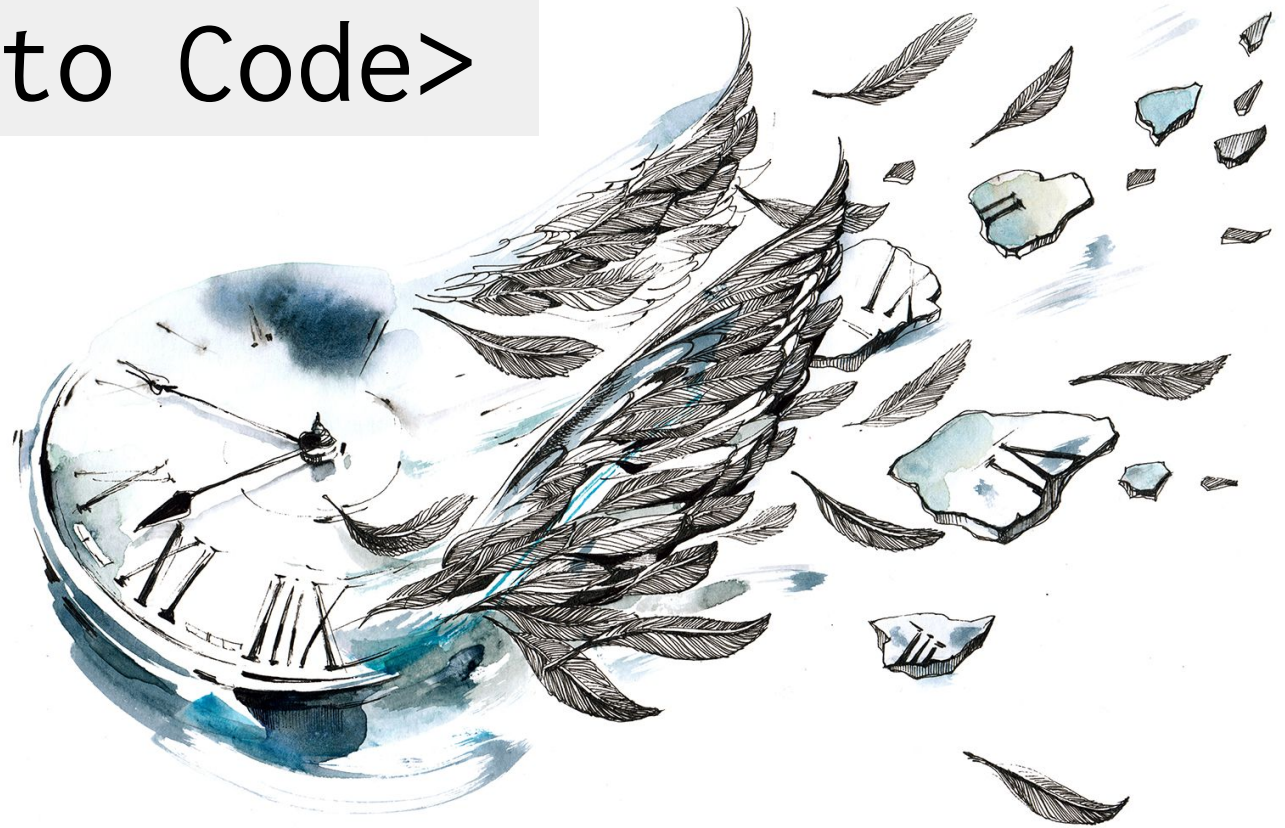


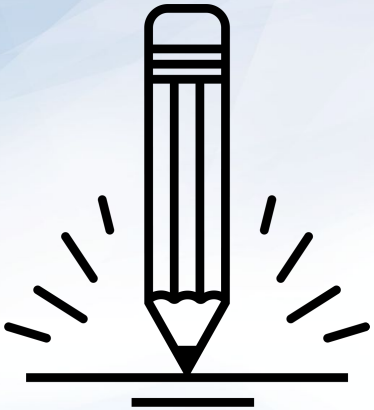
OpenWeatherMap API

- Remember to store keys in a config.py file.
- Similar patterns to previous api calls.


```
The weather API responded with: {'coord': {'lon': -0.13, 'lat': 51.51}, 'weather': [{'id': 500, 'main': 'Rain', 'desc
ription': 'light rain', 'icon': '10n'}], 'base': 'stations', 'main': {'temp': 280.25, 'pressure': 994, 'humidity': 6
6, 'temp_min': 279.15, 'temp_max': 282.15}, 'visibility': 10000, 'wind': {'speed': 7.7, 'deg': 260}, 'rain': {'3h':
1.235}, 'clouds': {'all': 92}, 'dt': 1516042200, 'sys': {'type': 1, 'id': 5091, 'message': 0.0047, 'country': 'GB',
'sunrise': 1516003129, 'sunset': 1516033320}, 'id': 2643743, 'name': 'London', 'cod': 200}.
```

<Time to Code>





Activity: Weather in Burundi

In this activity, you will work with the OpenWeather API and create an application which provides the user with the current temperature in the largest city of Burundi.

(Instructions sent via Slack.)

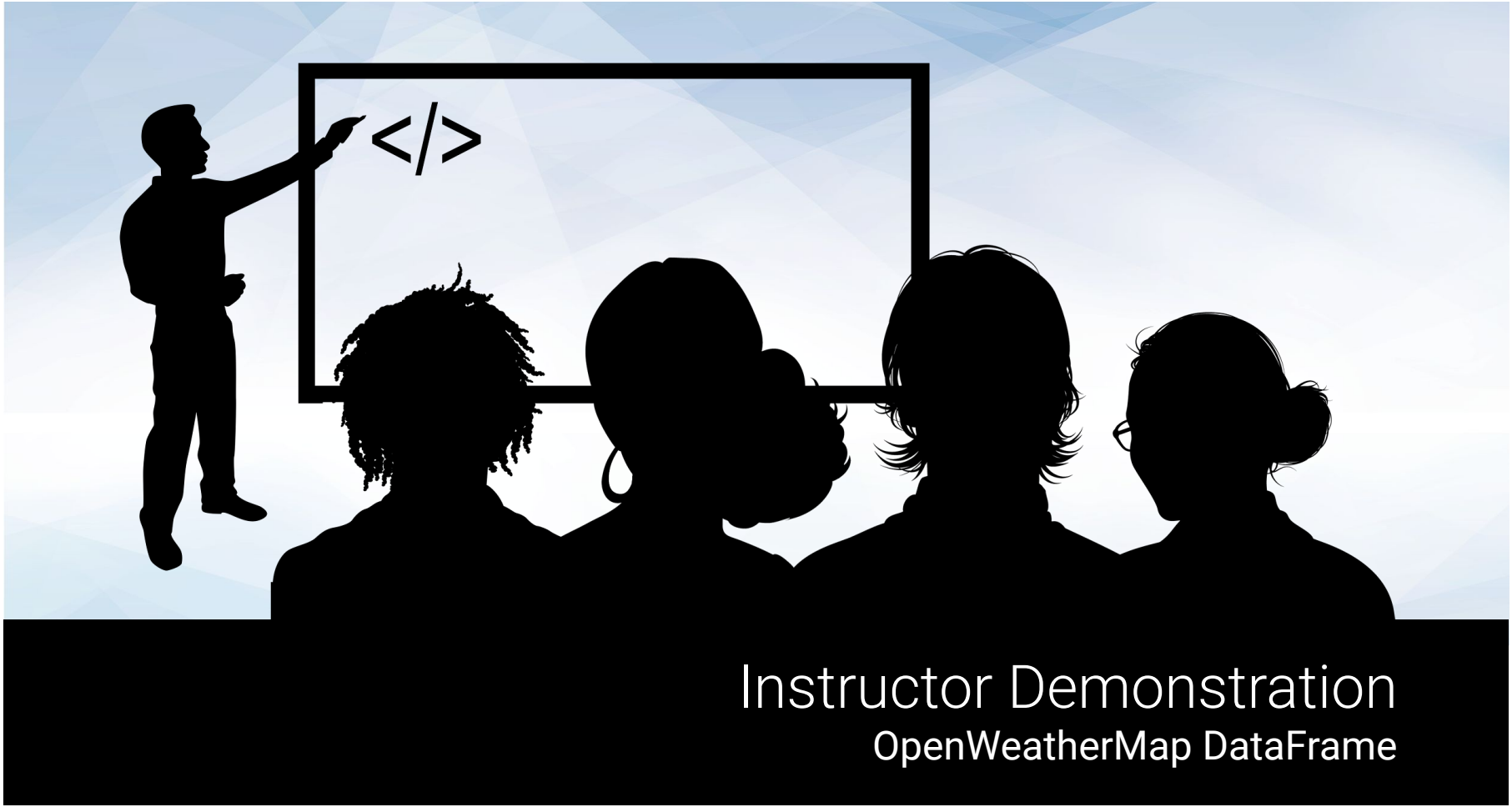
Suggested Time:
15 Minutes

Weather in Burundi Instructions

- Save all of your "config" information—i.e., your API key; the base URL; etc.—before moving on.
- Build your query URL. Check the documentation to figure out how to request temperatures in Celsius.
- Make your request, and save the API response.
- Retrieve the current temperature in Bujumbura from the JSON response.
- Print the temperature to the console.
- **Bonus:** Augment your code to report the temperature in both Fahrenheit _*and*_ Celsius.
- **Note: Don't forget to change the API key in config.py!**

```
The temperature in Bujumbura is 75.2 C.
```

Time’s Up! Let’s Review.



Instructor Demonstration

OpenWeatherMap DataFrame

OpenWeatherMap DataFrame

- Using our previous OpenWeatherMap API requests
- The API response contains fields such as temperature and latitude.
- A for loop is used to loop through the cities list, make a request and append to a list.
- What would be an easy way to analyze the different metrics?

```
cities = ["Paris","London","Oslo","Beijing"]

# set up lists to hold reponse info
lat = []
temp = []

# Loop through the list of cities and perform a request for data on each
for city in cities:
    response = requests.get(query_url + city).json()
    lat.append(response['coord']['lat'])
    temp.append(response['main']['temp'])

print(f"The latitude information received is: {lat}")
print(f"The temperature information received is: {temp}")
```

```
The latitude information received is: [48.86, 51.51, 59.91, 39.91]
The temperature information received is: [8.59, 6, 0, 1]
```

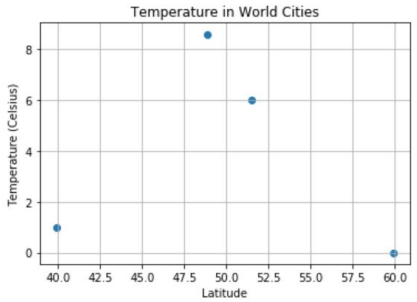
OpenWeatherMap DataFrame cont.

- Once all the data is collected the list can be stored into a dictionary then into a DataFrame.
- With the data now in a DataFrame it can be plotted with MatplotLib

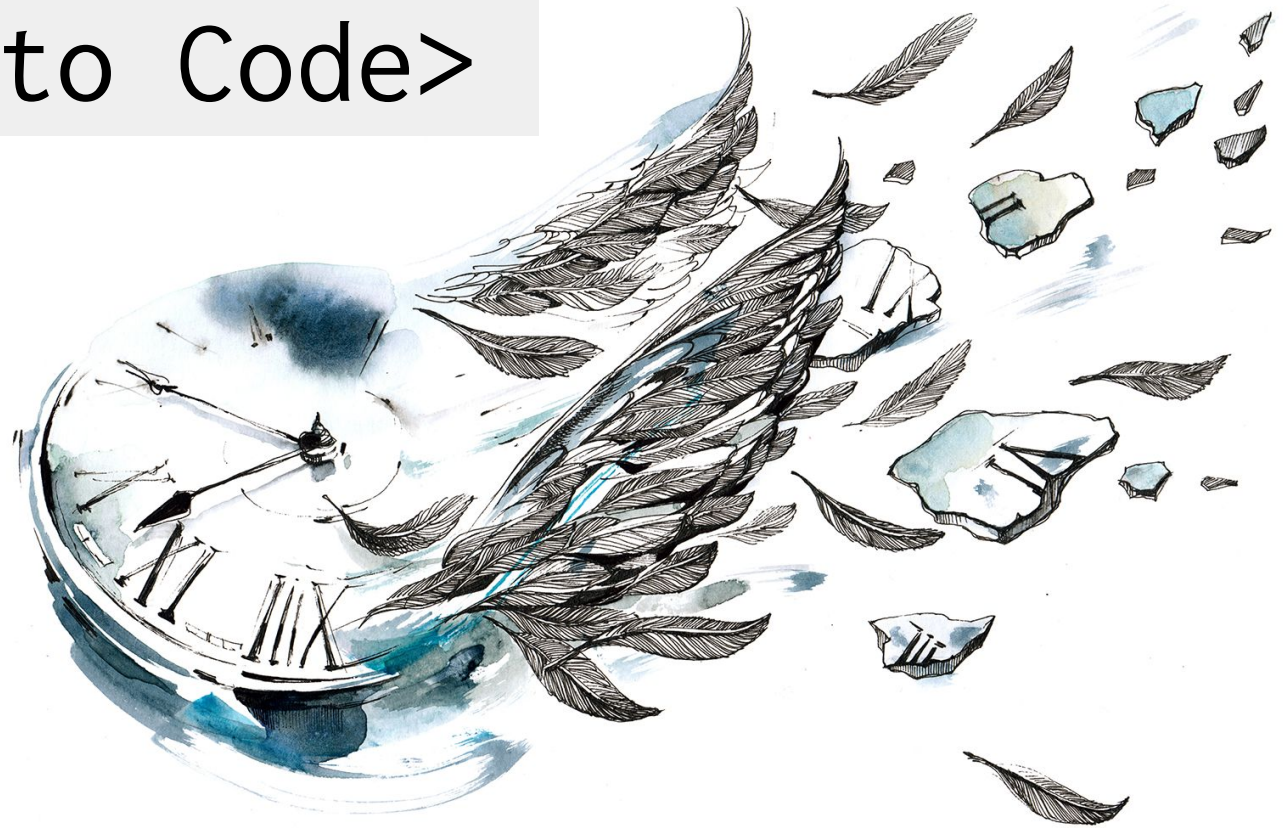
```
# create a data frame from cities, lat, and temp
weather_dict = {
    "city": cities,
    "lat": lat,
    "temp": temp
    }
weather_data = pd.DataFrame(weather_dict)
weather_data.head()
```

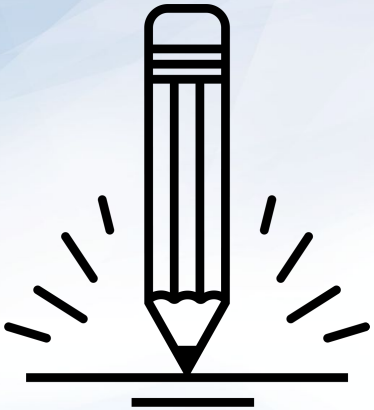
	city	lat	temp
0	Paris	48.86	8.59
1	London	51.51	6.00
2	Oslo	59.91	0.00
3	Beijing	39.91	1.00

```
# Build a scatter plot for each data type
plt.scatter(weather_data["lat"], weather_data["temp"], marker="o")

# Incorporate the other graph properties
plt.title("Temperature in World Cities")
plt.ylabel("Temperature (Celsius)")
plt.xlabel("Latitude")
plt.grid(True)

# Save the figure
plt.savefig("TemperatureInWorldCities.png")

# Show plot
plt.show()
```



<Time to Code>





Activity: TV Ratings

In this activity, you will take some time to create an application that reads in a list of TV shows, makes multiple requests from an API to retrieve rating information, creates a pandas dataframe, and visually displays the data.

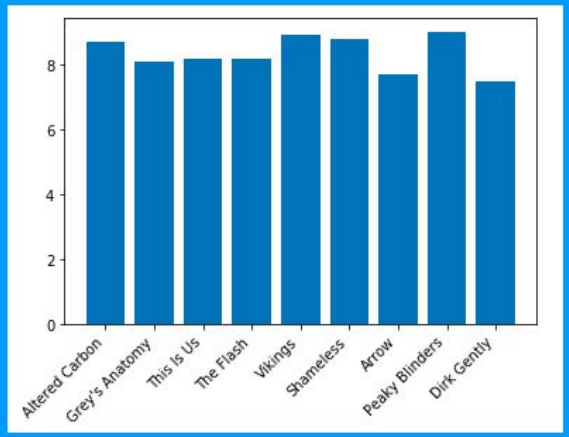
(Instructions sent via Slack.)

Suggested Time:
15 Minutes

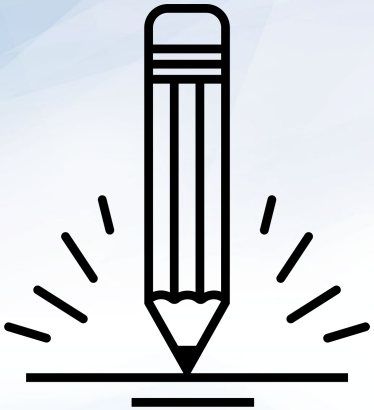
TV Ratings Instructions

- You may use the list of TV shows provided in the starter file or create your own.
- Request information on each TV show from https://www.tvmaze.com/api#show-search
- Store the name and rating information into lists.
- Store this data in a dictionary and use it to create a Pandas DataFrame.
- Use matplotlib to create a bar chart comparing the ratings of each show.

	rating	title
0	8.7	Altered Carbon
1	8.1	Grey's Anatomy
2	8.2	This Is Us
3	8.2	The Flash
4	8.9	Vikings
5	8.8	Shameless
6	7.7	Arrow
7	9.0	Peaky Blinders
8	7.5	Dirk Gently



Time’s Up! Let’s Review.



Activity: Weather Statistics

In this activity, you will generate a regression model on a dataset from the Open Weather API to predict the temperature of a city.

(Instructions sent via Slack.)

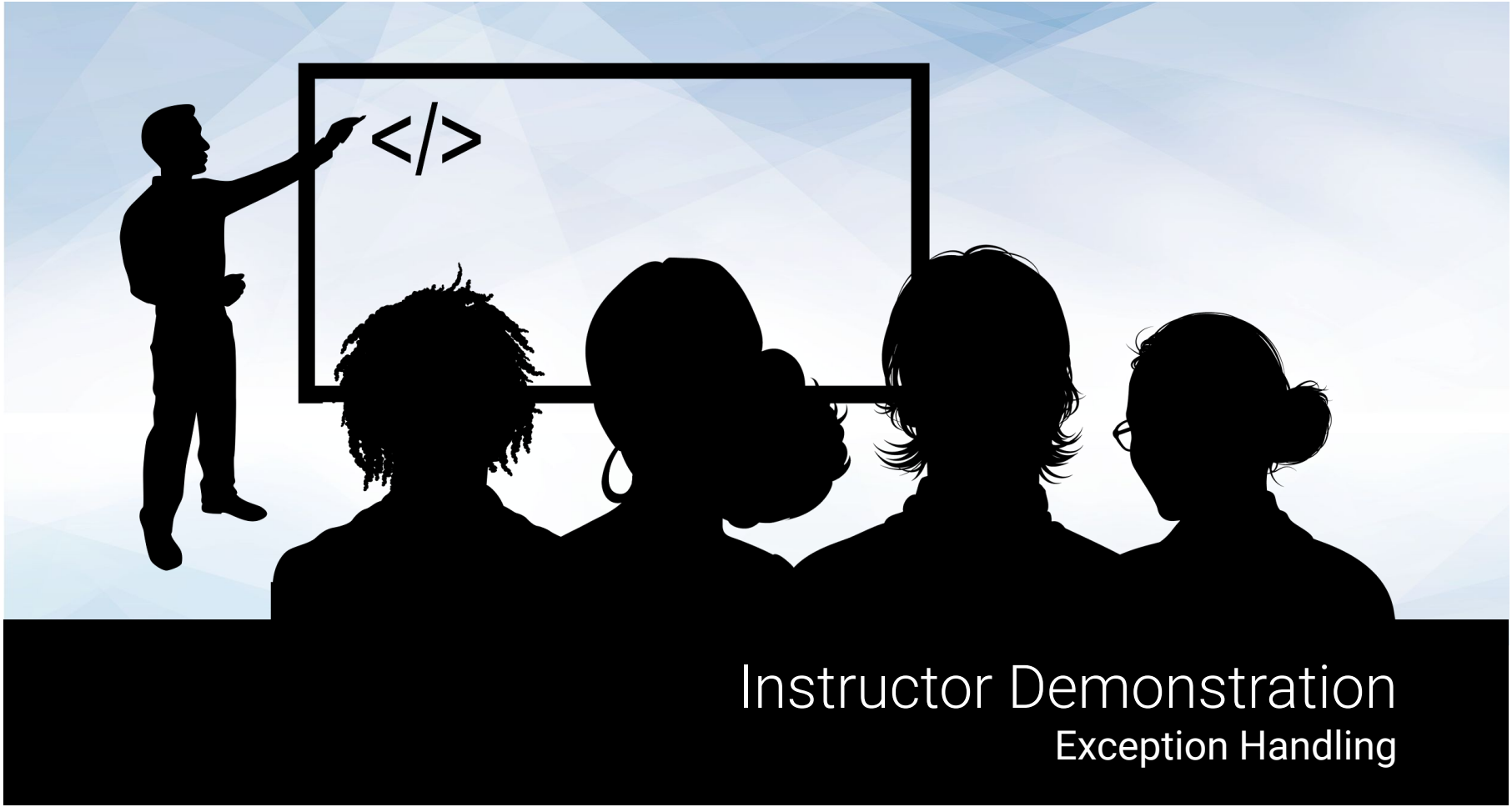
Suggested Time:
15 Minutes

Weather Statistics Instructions

- Using the starter file as a guide complete the following:
  - Create a scatter plot of Temperature vs. Latitude.
  - Perform linear regression.
  - Create a line equation for the regression.
  - Create a scatter plot with the linear regression line.
  - Predict the temperature of Florence at latitude 34.8
  - Use the API to determine the actual temperature of Florence.
- If you finish early feel free to try and predict the temperature at other cities.
- **HINT:** if you need help be sure to revisit your stats material from 5.3.

Time’s Up! Let’s Review.

Take a Break!



Instructor Demonstration

Exception Handling

What would happen if an application tried to look up a key within a dictionary that doesn’t exist?

Errors

```
students = {
    # Name  : Age
    "James": 27,
    "Sarah": 19,
    "Jocelyn": 28
}

print(students["Jezebel"])

print("This line will never print.")
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
<ipython-input-1-4692324b5d88> in <module>()
      6 }
      7
----> 8 print(students["Jezebel"])
      9
     10 print("This line will never print.")

KeyError: 'Jezebel'
```

- So far our requests to our APIs have had the values we are looking for.
- When a value is not found Python returns an error.
- As we can see in the notebook when `students` dictionary does not have a key for “Jezebel”.

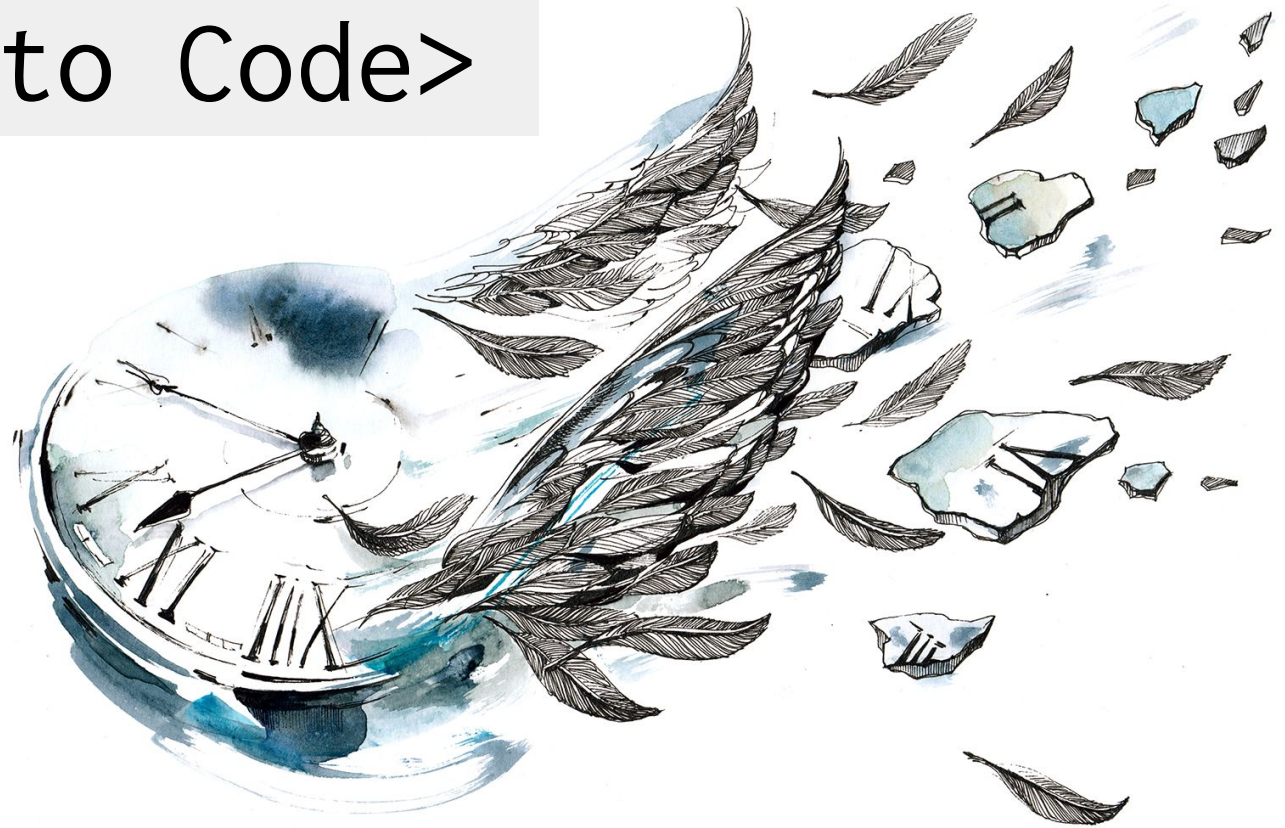
Try/Except

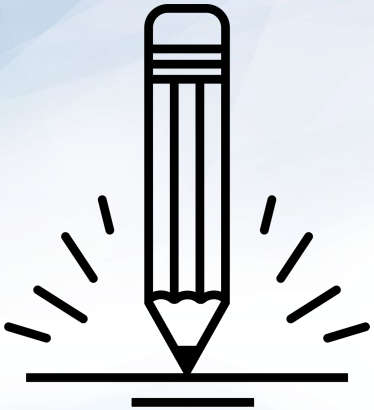
- The try/except code will let an application recover from errors like the one previously.
- Try and except are statements like for and if.
- Python will “try” to run the code.
- If the code throws and error or exception, the code in the except block in executed

```
students = {
    # Name  : Age
    "James": 27,
    "Sarah": 19,
    "Jocelyn": 28
}

# Try to access key that doesn't exist
try:
    students["Jezebel"]
except KeyError:
    print("Oops, that key doesn't exist.")

# "Catching" the error lets the rest of our code execute
print("...But the program doesn't die early!")
```

```
Oops, that key doesn't exist.
...But the program doesn't die early!
```

<Time to Code>





Activity: Making Exceptions

In this activity, you will create an application that, through `try` and `except`, resolves a number of errors.

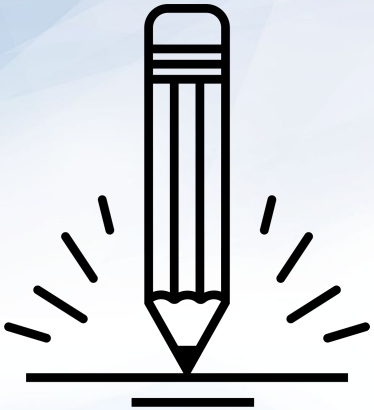
(Instructions sent via Slack.)

Suggested Time:
5 Minutes

Making Exceptions Instructions

- Without removing any of the lines from the starter code provided, create `try` and `except` blocks that will allow the application to run without terminating.

Time’s Up! Let’s Review.



Activity: API Call Exceptions

In this activity, you will implement **try/except** as you make API calls to narrow down a list of fictional characters to include only characters from Star Wars.

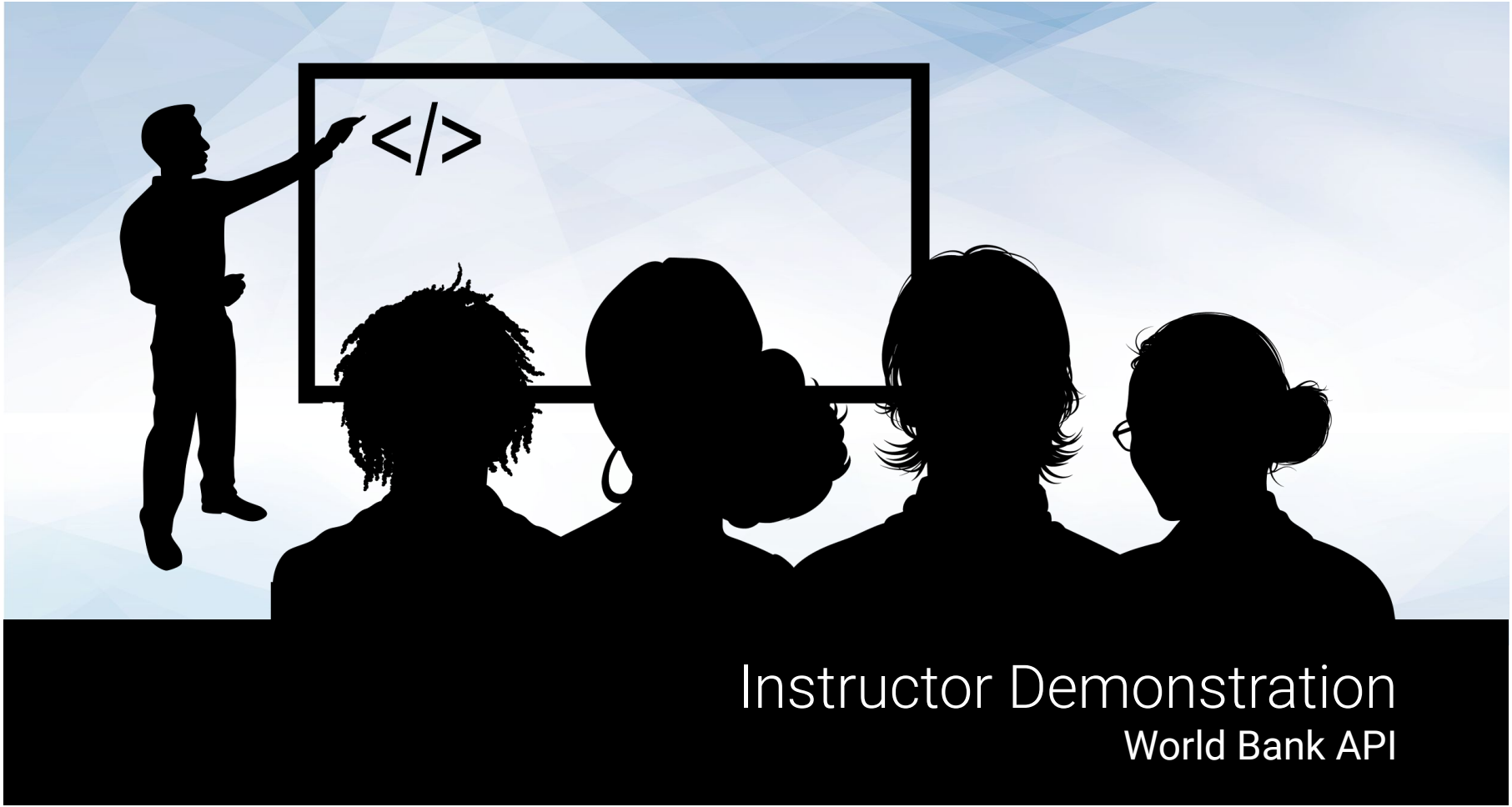
(Instructions sent via Slack.)

Suggested Time:
15 Minutes

Making Exceptions Instructions

- Loop through the characters in the list and send a request to the Star Wars API.
- Create a try clause to append the height and mass for characters available in the Star Wars API and an except clause to append None for those that do not.
- Create a DataFrame from the results.
- Drop any rows with null values to remove characters not in the Star Wars universe.

Time’s Up! Let’s Review.

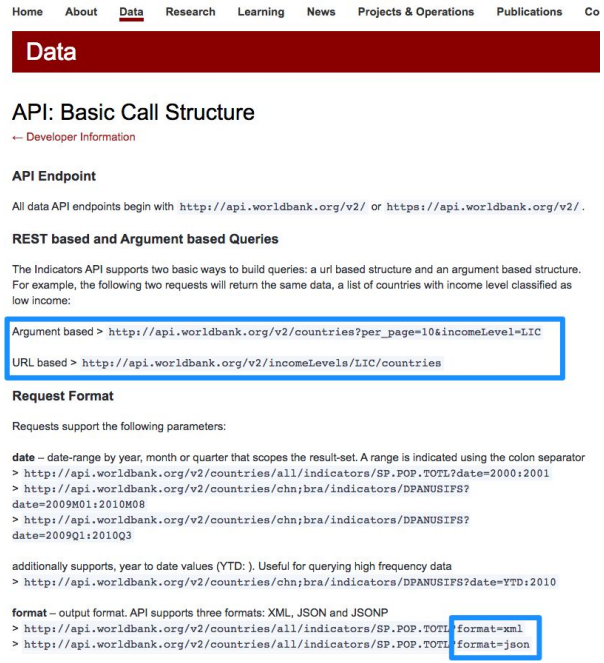


Instructor Demonstration

World Bank API

World Bank API

- Up to now we have been working with fairly straightforward API queries
- There are more complicated API frameworks that exist
- For the remainder of class we will practice working with more complicated APIs



Home About Data Research Learning News Projects & Operations Publications Co

Data

API: Basic Call Structure

← Developer Information

API Endpoint

All data API endpoints begin with `http://api.worldbank.org/v2/` or `https://api.worldbank.org/v2/`.

REST based and Argument based Queries

The Indicators API supports two basic ways to build queries: a url based structure and an argument based structure. For example, the following two requests will return the same data, a list of countries with income level classified as low income:

Argument based > `http://api.worldbank.org/v2/countries?per_page=10&incomeLevel=LIC`

URL based > `http://api.worldbank.org/v2/incomeLevels/LIC/countries`

Request Format

Requests support the following parameters:

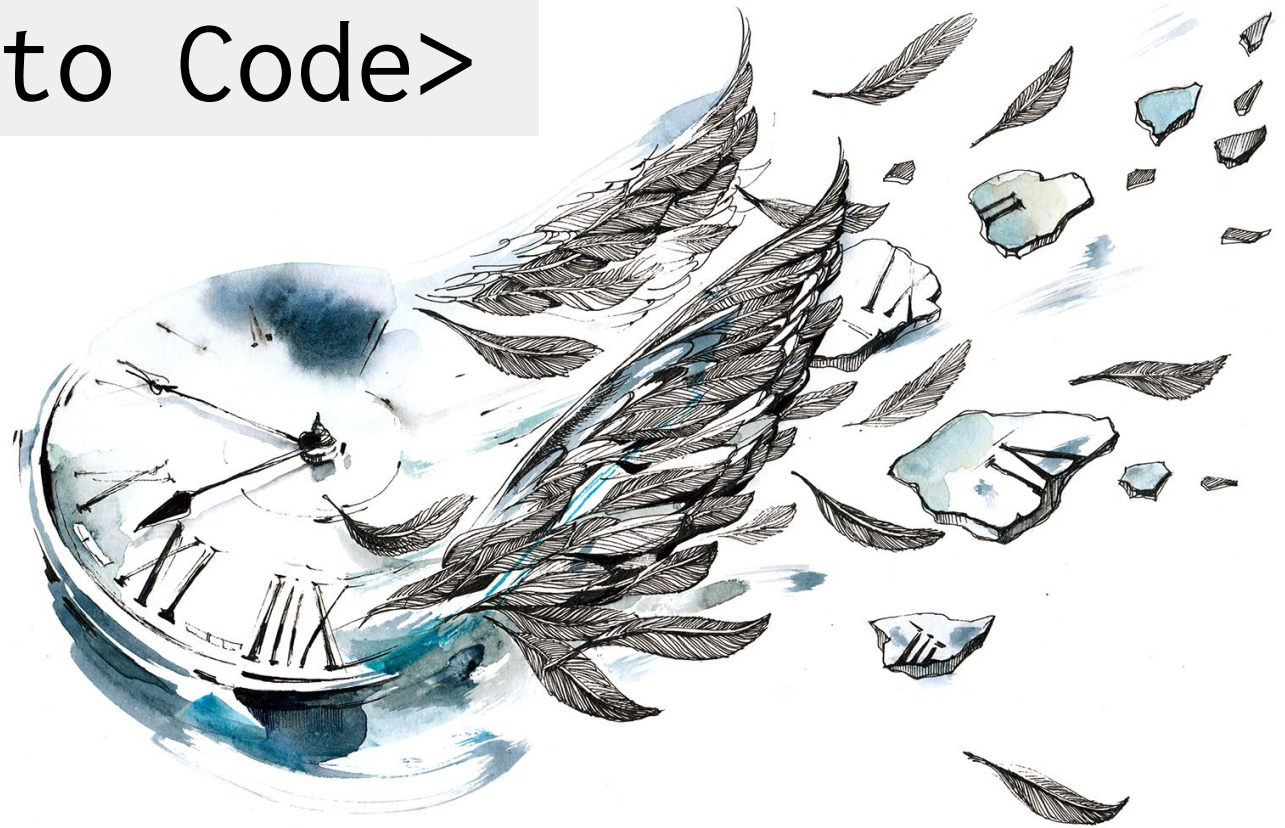
date – date-range by year, month or quarter that scopes the result-set. A range is indicated using the colon separator
> `http://api.worldbank.org/v2/countries/all/indicators/SP.POP.TOTL?date=2000:2001`
> `http://api.worldbank.org/v2/countries/chn;bra/indicators/DPANUSIFS?date=2009M01:2010M08`
> `http://api.worldbank.org/v2/countries/chn;bra/indicators/DPANUSIFS?date=2009Q1:2010Q3`

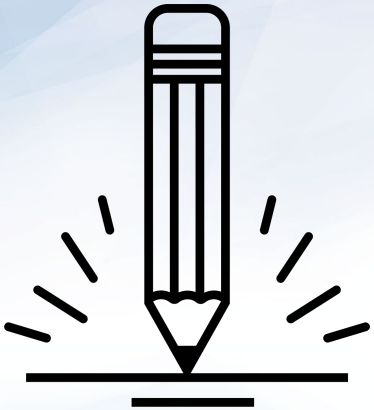
additionally supports, year to date values (YTD:). Useful for querying high frequency data
> `http://api.worldbank.org/v2/countries/chn;bra/indicators/DPANUSIFS?date=YTD:2010`

format – output format. API supports three formats: XML, JSON and JSONP
> `http://api.worldbank.org/v2/countries/all/indicators/SP.POP.TOTL?format=xml`
> `http://api.worldbank.org/v2/countries/all/indicators/SP.POP.TOTL?format=json`

```
url = "http://api.worldbank.org/v2/"
format = "json"

# Get country information in JSON format
countries_response = requests.get(f"{url}countries?format={format}").json()
```

<Time to Code>





Activity: Two Calls

In this activity, you will be utilizing the World Bank API to make two API calls in a sequence. The second API call depends on the response of the first.

(Instructions sent via Slack.)

Suggested Time:
10 Minutes

Two Calls Instructions

- Retrieve a list of the lending types the world bank keeps track of, and extract the ID key from each of them.
- Next, determine how many countries are categorized under each lending type. Use a dict to store this information.
    - This data is stored as the first element of the response array.
- Finally, print the number of countries of each lending type.

```
The number of countries with lending type IBD is 140.
The number of countries with lending type IDB is 30.
The number of countries with lending type IDX is 118.
The number of countries with lending type LNX is 74.
```

Time’s Up! Let’s Review.