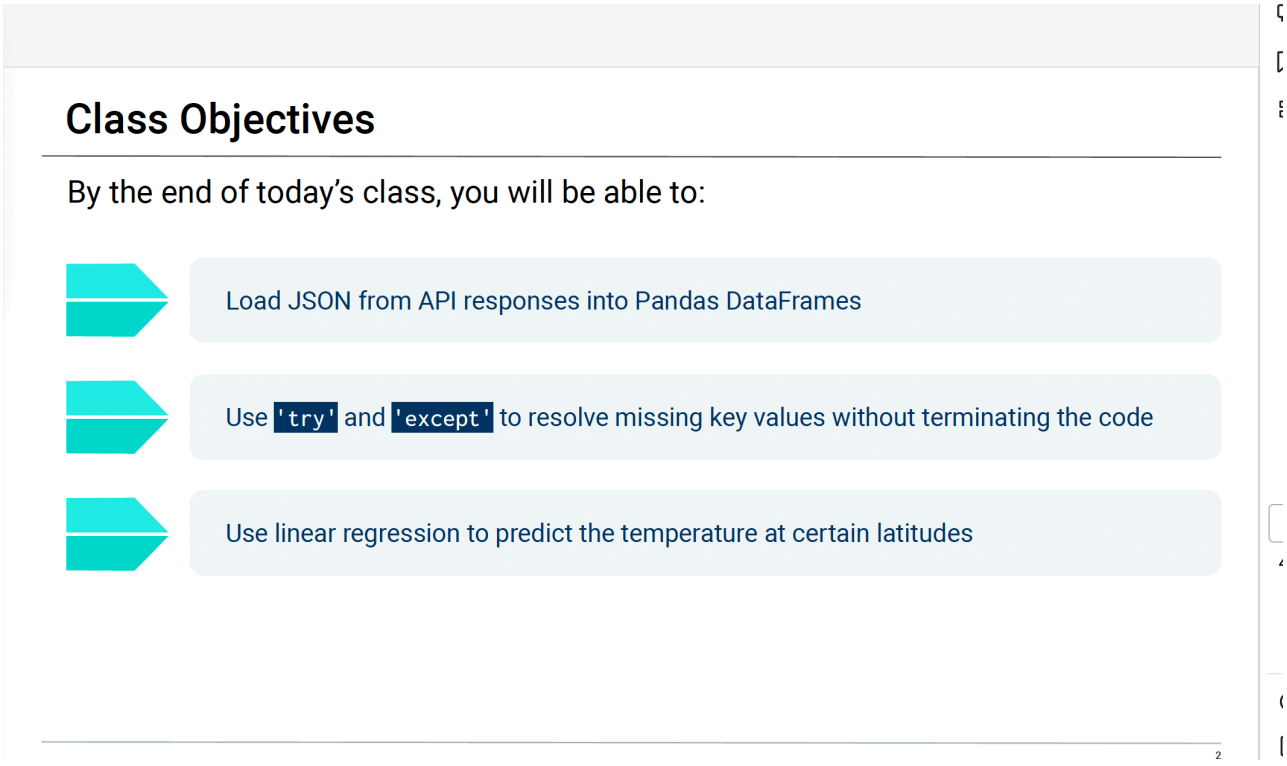
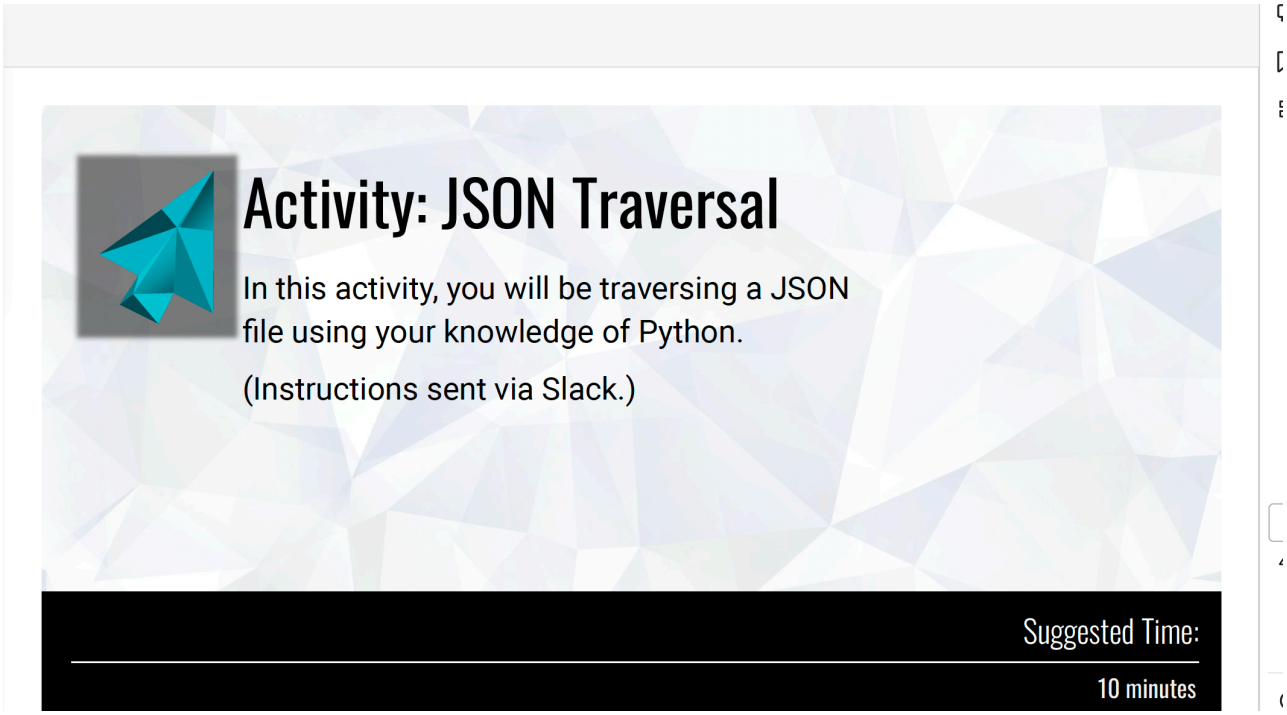
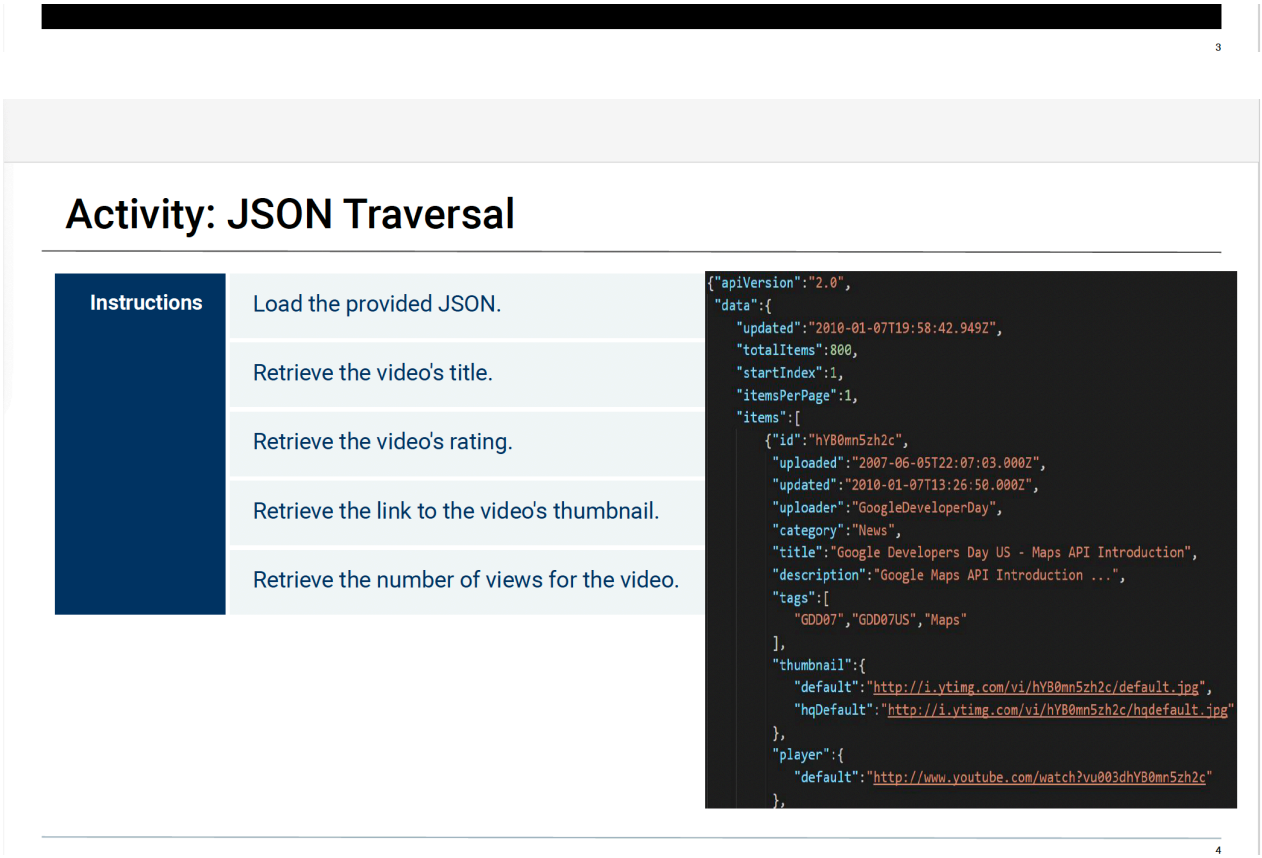
MOD 6 DAY 2 **Working with Weather and City APIs WITH CODE**

Tuesday, October 17, 2023 6:11 PM







Activity 1

Json traveleres

0:16:00

We import dependensin

what is "os"

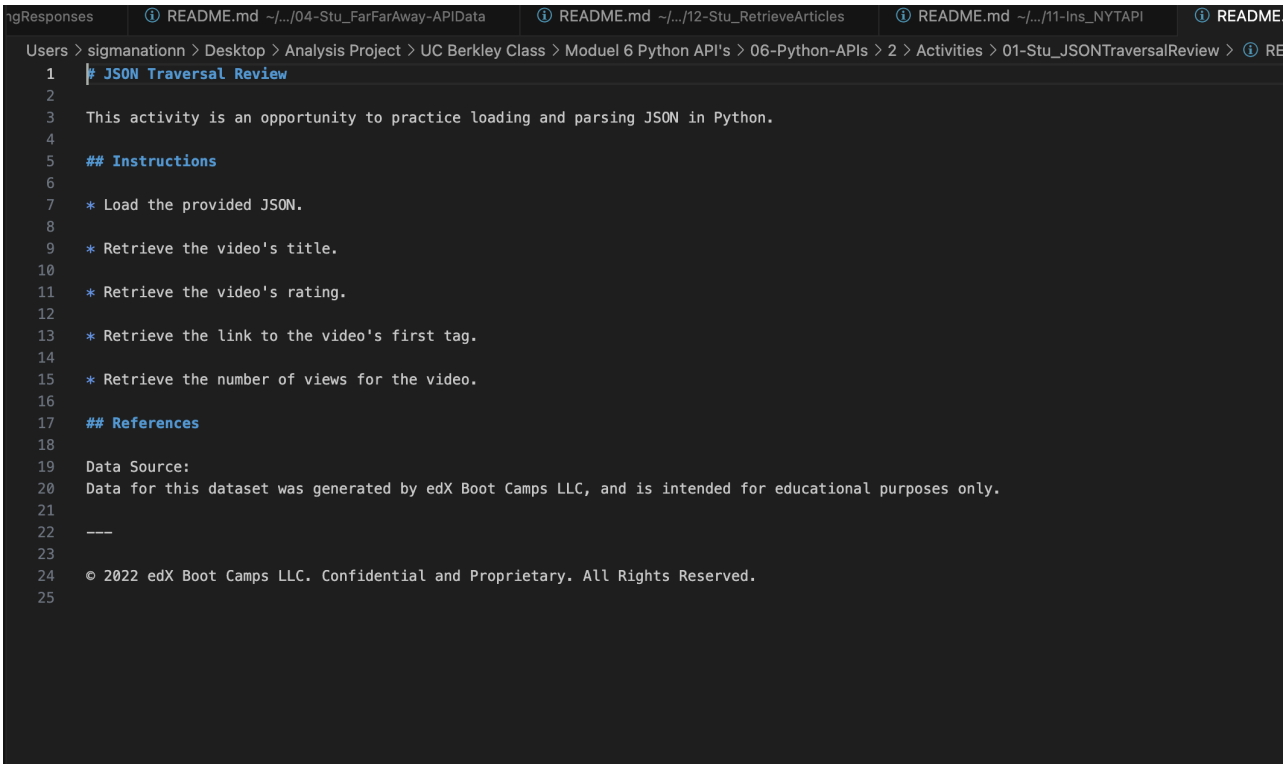
We import and load json

We sepertate by file types

We access the list

what is a dictioary

what is a list



Actiicuty 2

00:41:00

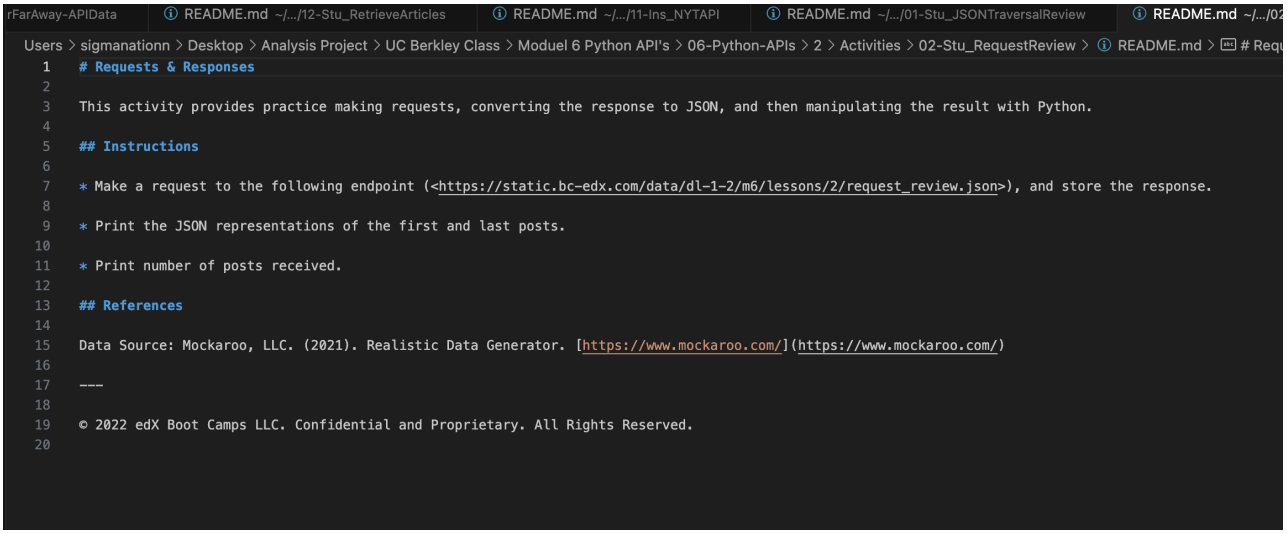
Request review

Import request dependence

We creat a json reponse and pretty print the resposnse

How to search and navigate an api (00:52:00)

(reference the Day 1 on how to access api and search them)



Activity 3

1:14:00

03-Ins_OpenWeatherRequest

Impoert request and config api

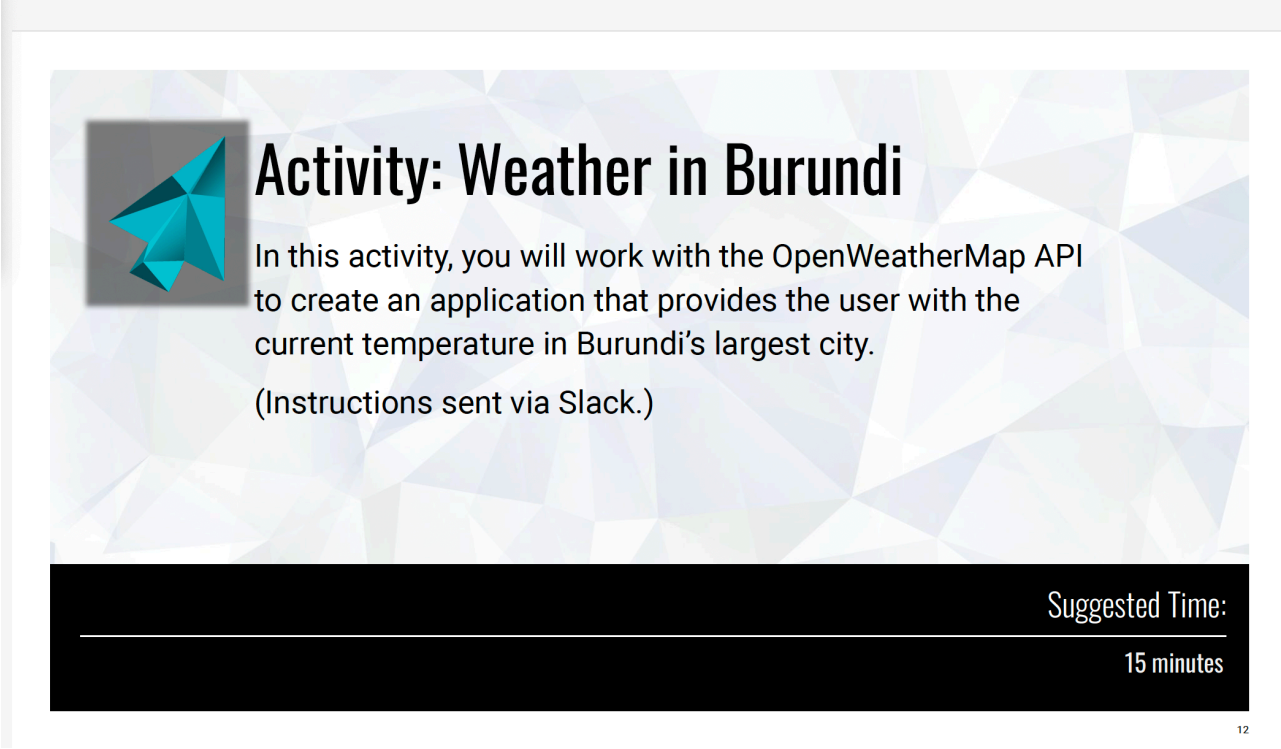
(use the metrics of the website of api to get that I want)

Get temp of a location

Lon and lattirude

Main

Metric



Activity: Weather in Burundi

Instructions	Save all of your `"config"` information—i.e., your API key, the base URL, etc.—before getting started.
	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.
Hints	Check the documentation to figure out how to request temperatures in Celsius.
	Don't forget to change the API key in `config.py`!

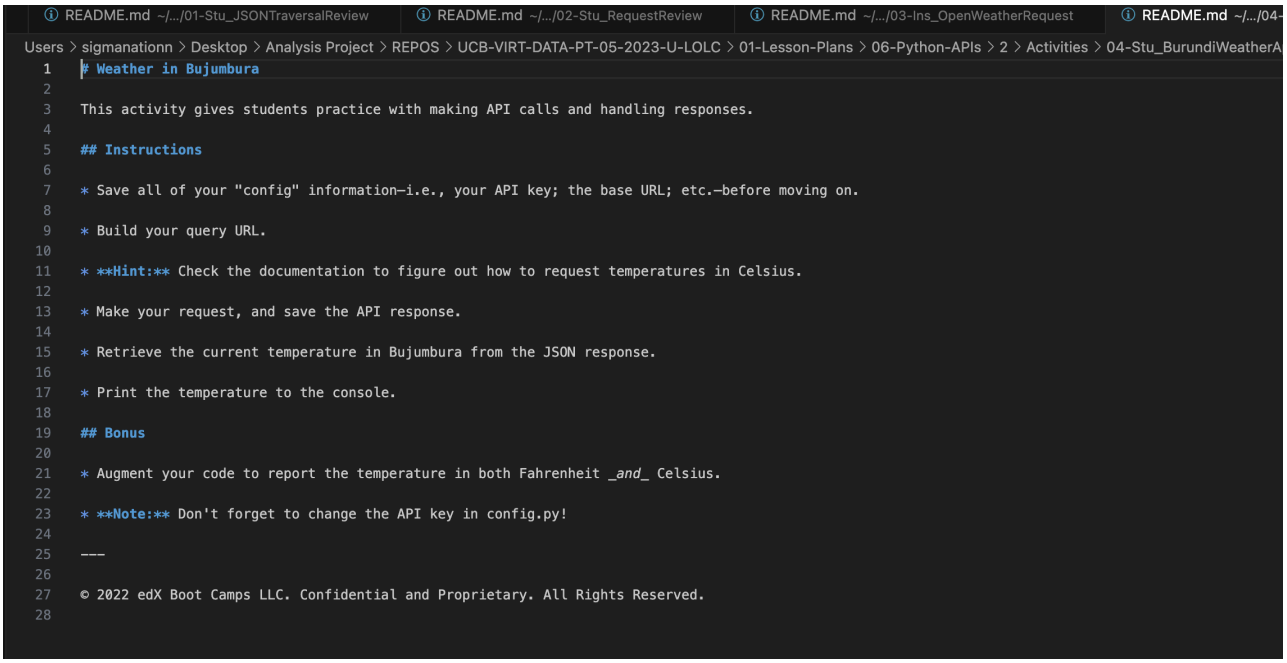
```
The temperature in Bujumbura is 75.2 C.
```

13

Activity 4

Str bnrundi

1:20:00 (put api in a config file and and call it as a depencency) (to referecen later)

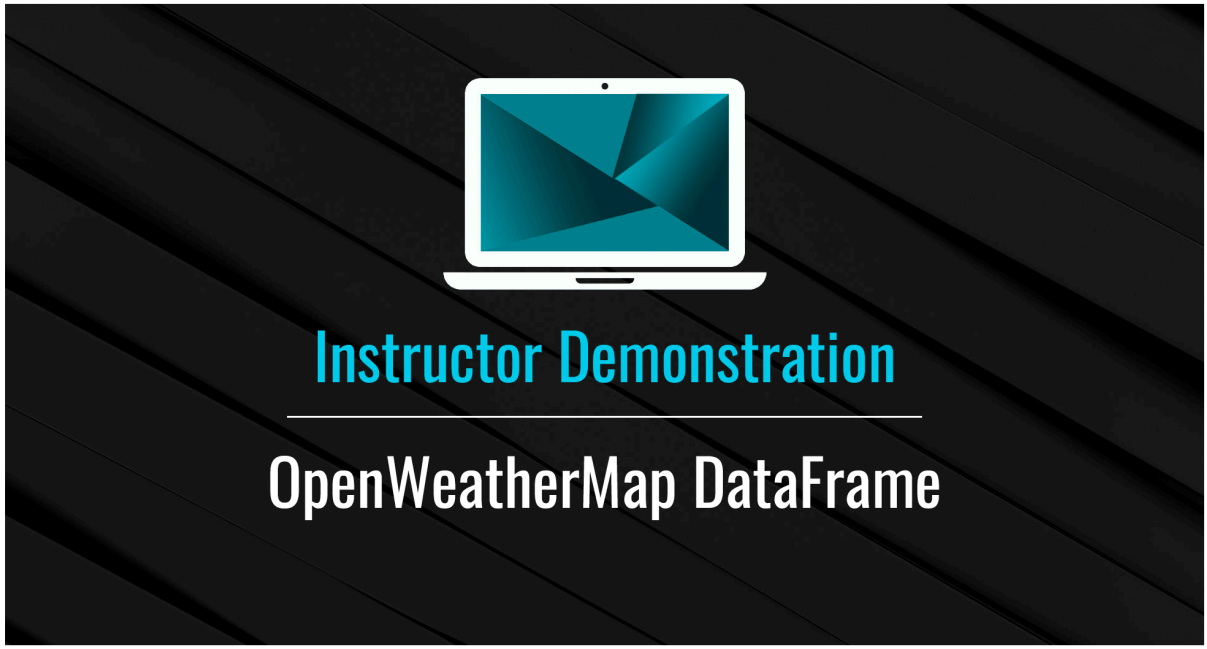


Activity 05

05-Ins_OpenWeatherDataFrame

1:24:00

We use json and urls to make tables and data frames



OpenWeatherMap DataFrame

- We'll use 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?

OpenWeatherMap DataFrame

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

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

# Loop through a 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, continued

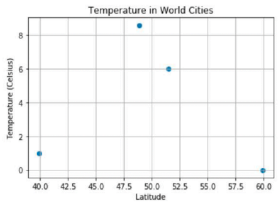
Once all the data has been collected, the list can be stored in a dictionary and then in a DataFrame.

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()
```

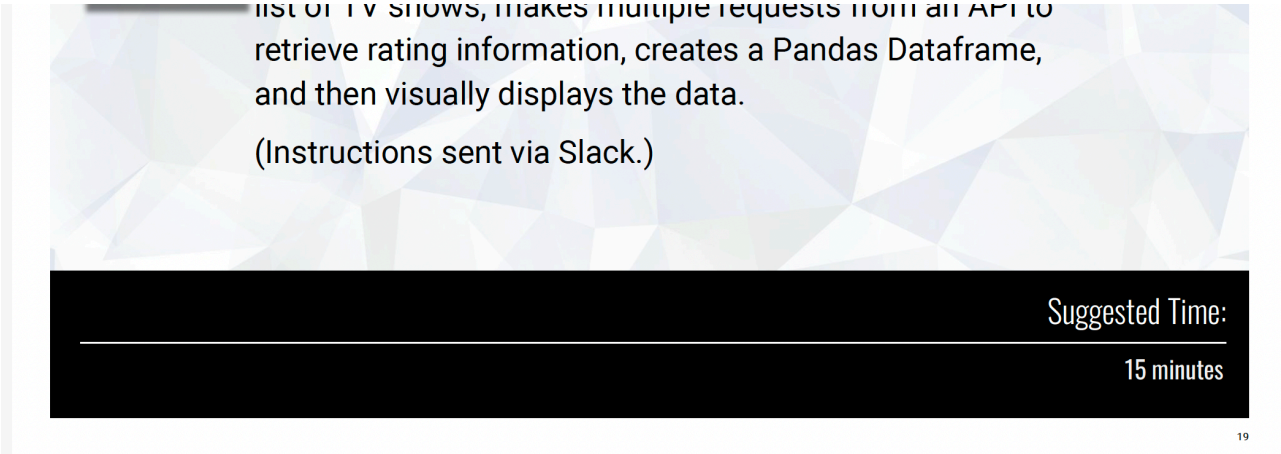
	city	lat	temp
0	Paris	48.86	8.59
1	London	51.51	6.00
2	Oslo	59.91	0.00
3	Beijing	39.91	1.00

```
# 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()
```



Activity: TV Ratings

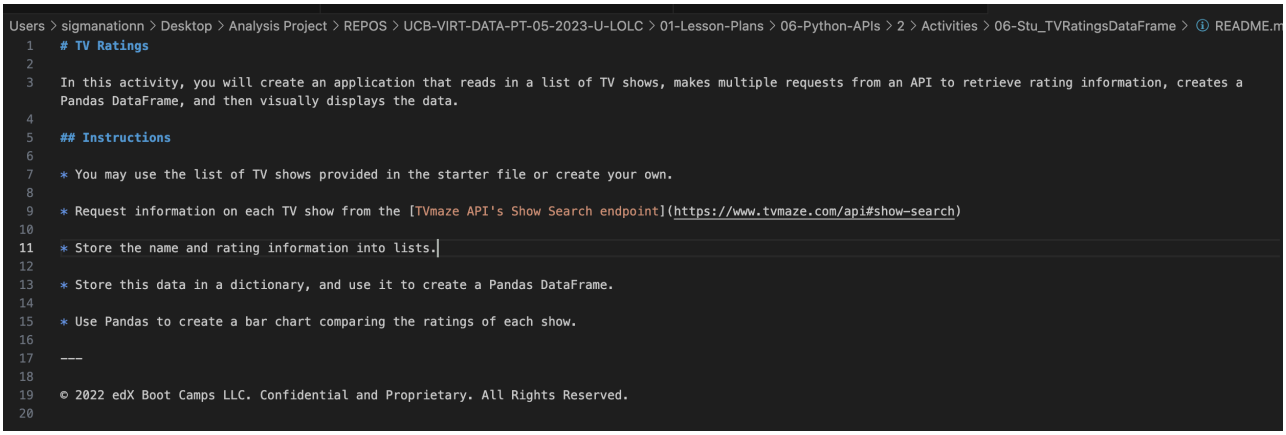
In this activity, you'll create an application that reads in a list of TV shows, makes multiple requests from an API to

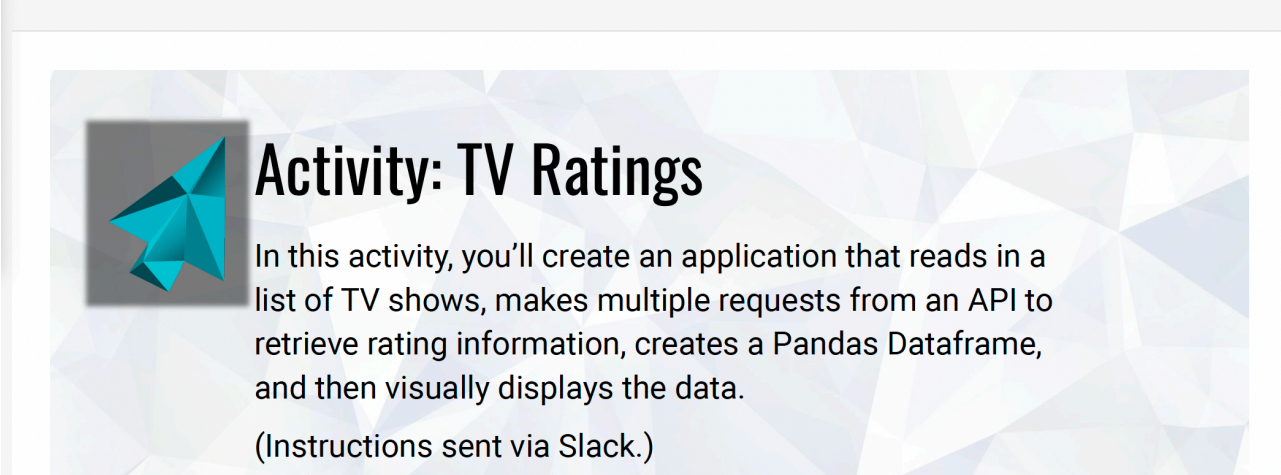


Activity 6

06-Stu_TVRatingsDataFrame

We import json url into a dataframe and out put list





Suggested Time:

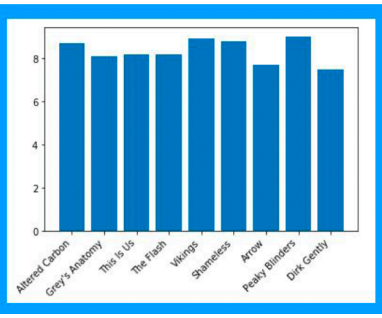
15 minutes

19

Activity: 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 in 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



20

ACTIVITY 07

2:03:00

07-Stu_Weather_Stats

HOW TO CREATE LIST AND WHAT ARE THEY

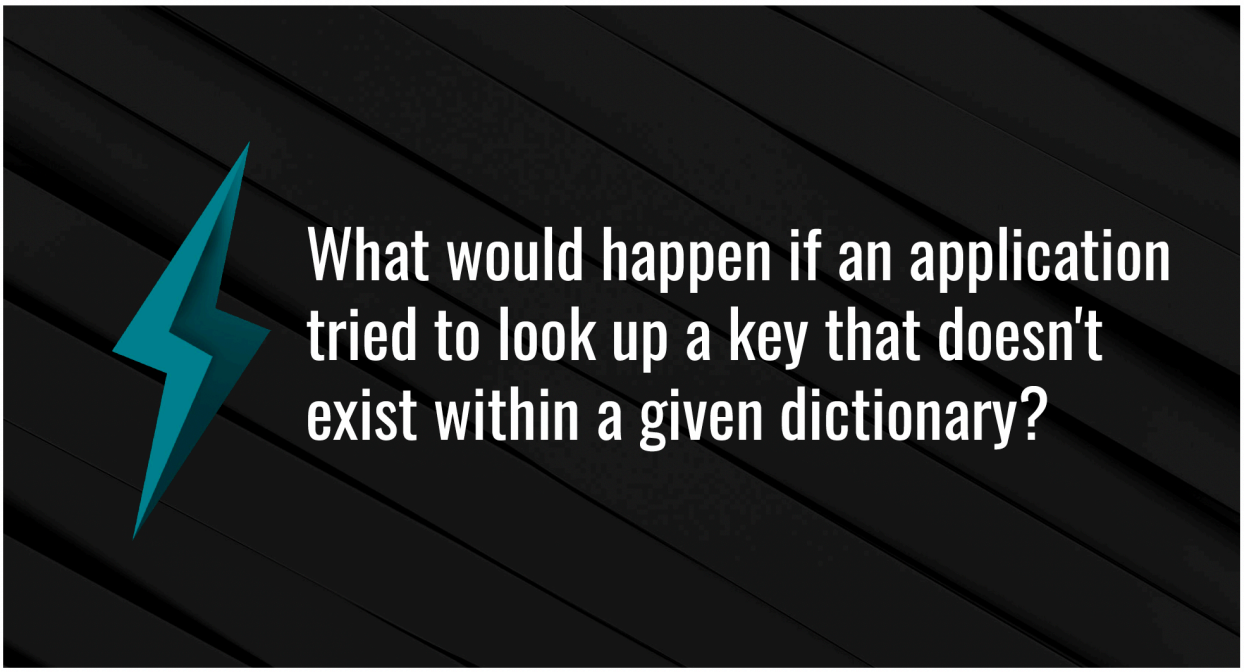
We create a line equation

Users > sigmanationn > Desktop > Analysis Project > REPOS > UCB-VIRT-DATA-PT-05-2023-U-LOLC > 01-Lesson-Plans > 06-Python-APIs > 2 > Activities > 07-Stu_Weather_Stats > README.md > # Weather

```
# Weather Statistics

In this activity, you will perform a linear regression on weather data from select cities in the Northern Hemisphere, and you will use the results to predict the temperature in Florence, Italy.

## Instructions

* Using the starter file as a guide, complete the following steps:

  * 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, which is at latitude 43.77&deg;.

  * Use the API to determine the actual temperature in Florence.

## Bonus

If you finish early, feel free to try to predict the temperatures of other cities.

## Hint

If you need help, revisit the material on statistics from Unit 5.

---

© 2022 edX Boot Camps LLC. Confidential and Proprietary. All Rights Reserved.
```

Activity 08

08-Ins_ExceptionHandling



Errors

So far, our API requests have had the values we're looking for.

When a value is not found, Python returns an error, as it does in our notebook when we look up `"Mary"`

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

print(students["Mary"])

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

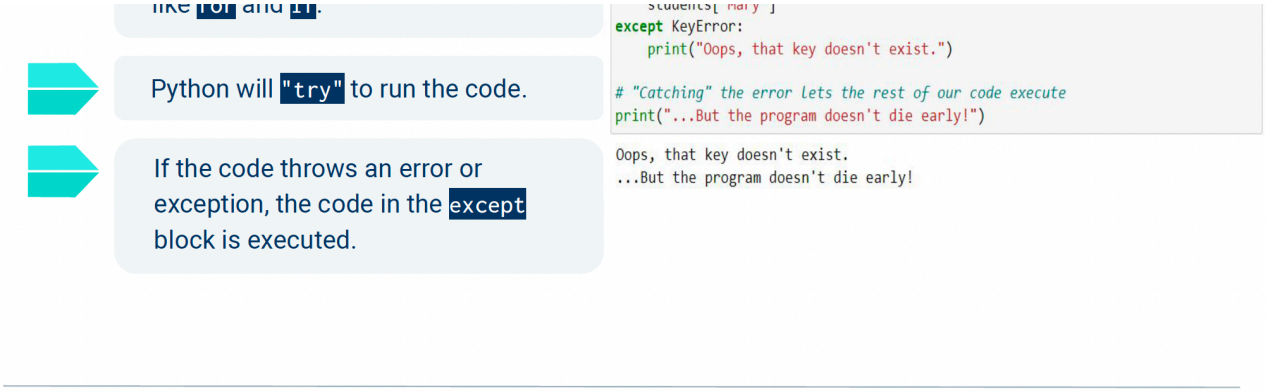
```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
/var/folders/1n/50wf_pbs5bsb8__bwdkxwvq86mm3js/T/ipykernel_69817/95543560.py in
<module>
      6 }
      7
----> 8 print(students["Mary"])
      9
     10 print("This line will never print.")

KeyError: 'Mary'
```

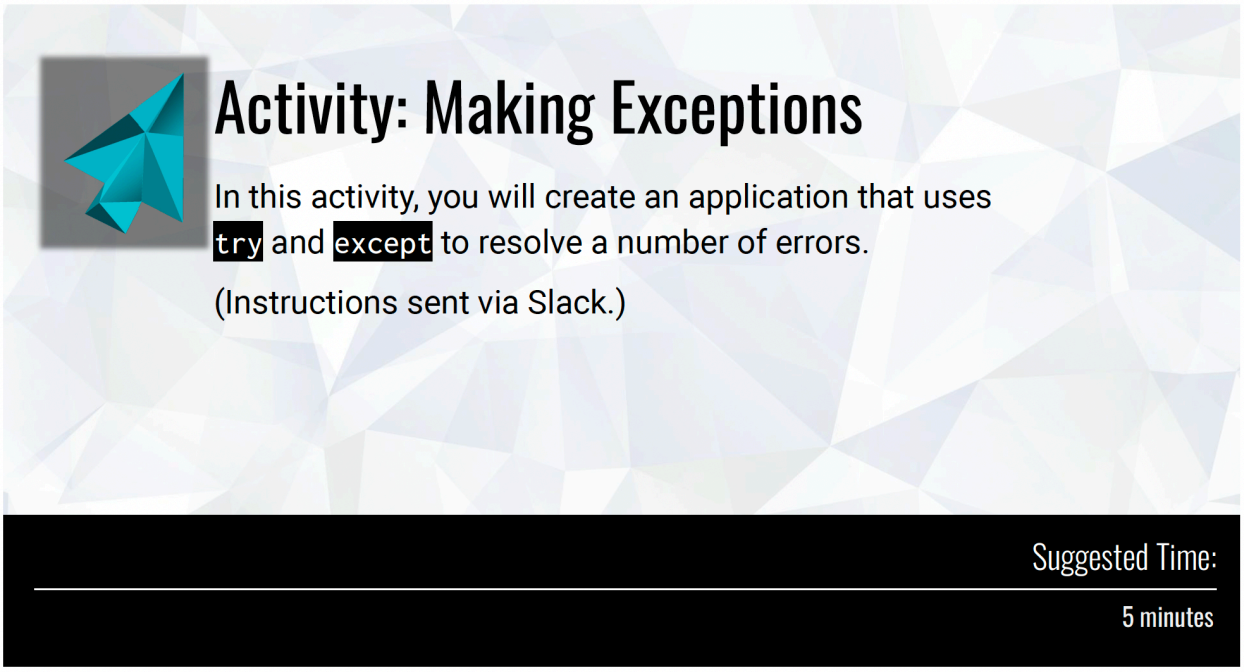
Try-Except

The `try-except` code will let an application recover from errors like our Mary example

`"try"` and `except` are statements like `for` and `if`

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

# Try to access key that doesn't exist
try:
```





Activity 9

2:27:00

09-Stu_MakingExceptions

Activity: API Call Exceptions

In this activity, you will implement `try-except` blocks 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

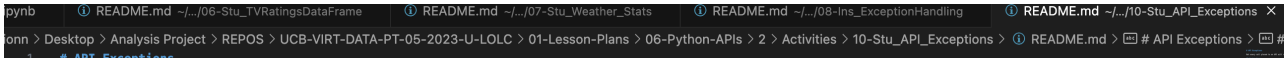
Activity: API Call Exceptions

Instructions	
	Loop through the characters in the list, and send a request to the Star Wars API.
	Create a `'try'` clause and an `'except'` clause. In the `'try'` clause, append the height, mass, and character name that is available in the Star Wars API to their respective lists.
	If the character is not available in the Star Wars API, create an `'except'` clause to print a message and `'pass'`.
	Create a DataFrame from the results.

Activity 10

2:29:00

10-Stu_API_Exceptions



```
# API Exceptions

Not every call placed to an API will return a result. In this activity, you will use `try` and `except` to handle errors from API calls.

## Instructions

* Loop through the characters in the list, and send a request to the Star Wars API.

* Create a `try` clause and an `except` clause. In the `try` clause, append the height, mass, and character name that is available in the Star Wars API to their respective lists. If the character is not available in the Star Wars API, use the `except` clause to print a message and `pass`.

* Create a DataFrame from the results.

---

© 2022 edX Boot Camps LLC. Confidential and Proprietary. All Rights Reserved.
```

Activity 11

2:33:00

11-Ins_WorldBankAPI

https://datahelpdesk.worldbank.org/

36

World Bank API

Up until now, we have been working with

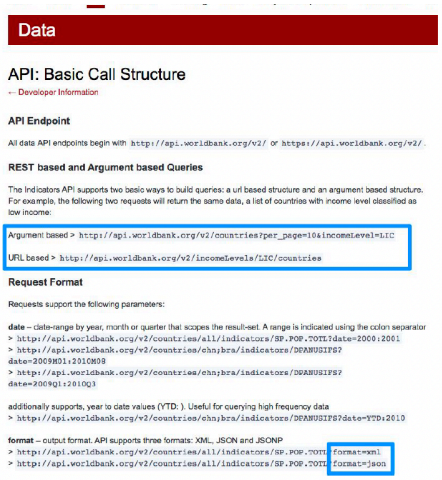
Home About Data Research Learning News Projects & Operations Publications Co

straightforward API queries.

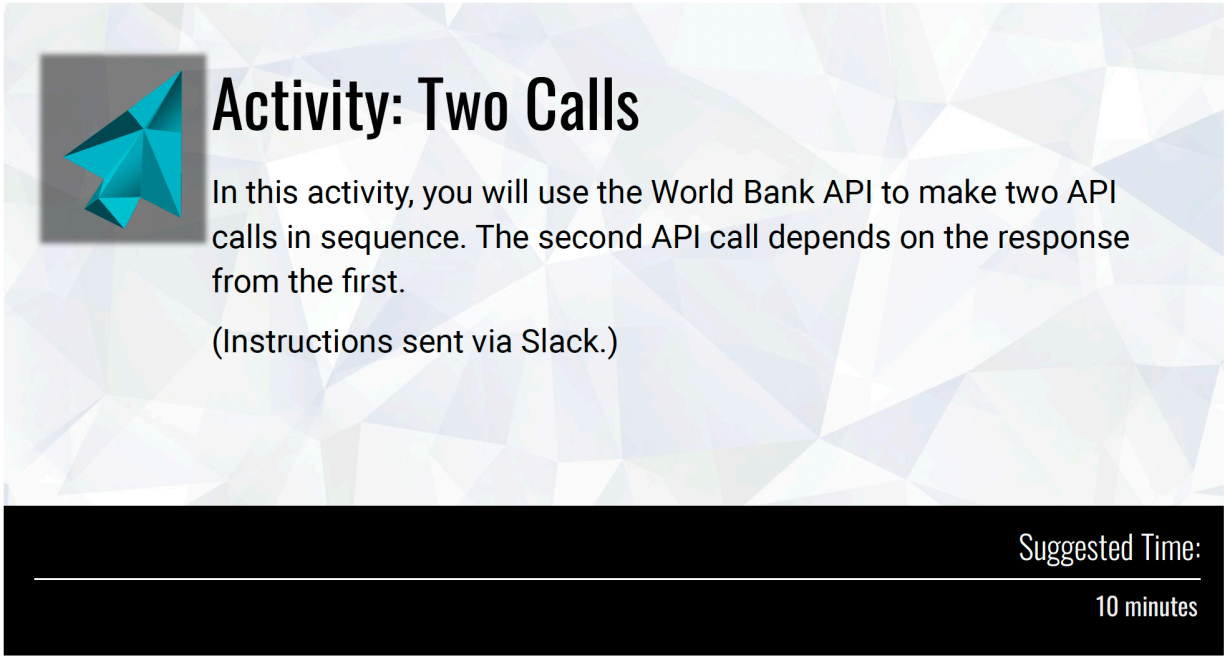
Of course, more complicated API frameworks exist.

For the remainder of class, we will practice working with more complicated APIs.

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

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



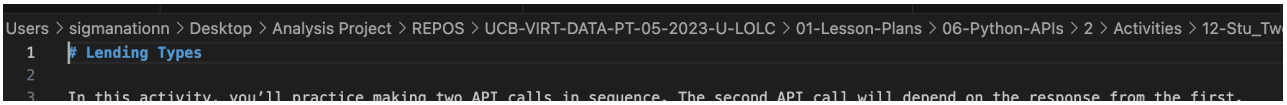
37



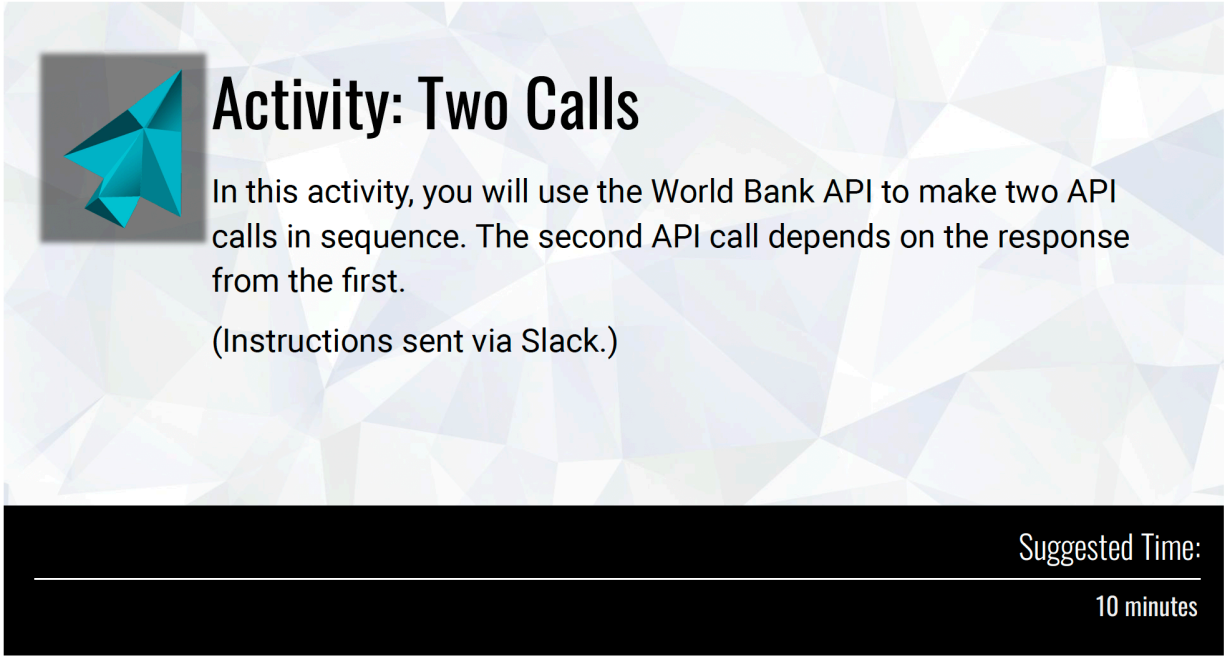
38

Activity 12

12-Stu_TwoCalls



```
4
5   ## Instructions
6
7   * Retrieve a list of the lending types that the world bank keeps track of, and extract the ID key for each one.
8
9   * Next, determine how many countries are categorized under each lending type. Use a dictionary to store this information.
10
11    * This data is stored as the first element of the response array.
12
13  * Finally, print the number of countries for each lending type.
14
15  ---
16
17  © 2022 edX Boot Camps LLC. Confidential and Proprietary. All Rights Reserved.
18
```



38

Activity: Two Calls

Instructions

Retrieve a list of the lending types that the World Bank keeps track of, and extract the ID key from each lending types or list.

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 for each lending type.

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

Activity 1

json_traversal_solution

Dependencies
Load JSON
Isolate "data items" for easy reading
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

JSON Traversal Review

This activity is an opportunity to practice loading and parsing JSON in Python.

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 for the video.

Activity 2

requests_review_solution

Dependencies
Specify the URL
Make request and store response
Verify status code
JSON-ify response
Print the number of responses

Requests & Responses

This activity provides practice making requests, converting the response to JSON, and then manipulating the result with Python.

Instructions

* Make a request to the following endpoint (<https://static.bc-edx.com/data/dl-1-2/m6/lessons/2/request_review.json>), and store the response.

* Print the JSON representations of the first and last posts.

* Print number of posts received.

References

Data Source: Mockaroo, LLC. (2021). Realistic Data Generator. https://www.mockaroo.com/

Activity 3

open_weather_request_solution

open_weather_request_solution

Dependencies
Save config information
Build query URL
Get weather data
Get the temperature from the response

Activity 4

Burundi_solution

Dependencies
Save config information.
Build query URL and request your results in Celsius
Get weather data
Get temperature from JSON response
Report temperature
Weather in Bujumbura
BONUS

use list of units
set up list to hold two different temperatures
loop through the list of units and append them to temperatures list
    # Build query URL based on current element in units
    # Get weather data
    # Get temperature from JSON response
    # Report temperatures by accessing each element in the list

This activity gives students practice with making API calls and handling responses.

Instructions

* Save all of your "config" information—i.e., your API key; the base URL; etc.—before moving on.

* Build your query URL.

* **Hint:** 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.

Activity 5

open_weather_dataframe_solution

```
# Dependencies
# Save config information.
# Build partial query URL
# List of cities
# set up lists to hold response info
# Loop through the list of cities and perform a request for data on each
# create a DataFrame from cities, lat, and temp
# Build a scatter plot for each data type
# Incorporate the other graph properties
# Save the figure
# Show plot
```

Activity 6

TV_ratings_solution

Dependencies
list of TV show titles to query
TV Maze show search base URL
set up lists to hold response data for name and rating
loop through TV show titles, make requests and parse
create DataFrame
Plot the data

TV Ratings

In this activity, you will 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 then visually displays the data.

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 the [TVmaze API's Show Search endpoint](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 Pandas to create a bar chart comparing the ratings of each show.

Activity 7

weather_stats_solution

Dependencies
Save config information.
Build partial query URL
List of cities
set up lists to hold reponse info
Loop through the list of cities and perform a request for data on each
create a data frame from cities, lat, and temp
Create a Scatter Plot for temperature vs latitude
Perform a linear regression on temperature vs. latitude
Get regression values
Create line equation string
Create Plot
Label plot and annotate the line equation
Print r value
Show plot
Calculate the temperature for Florence at 43.77 degrees
Use API to determine actual temperature

Weather Statistics

In this activity, you will perform a linear regression on weather data from select cities in the Northern Hemisphere, and you will use the results to predict the temperature in Florence, Italy.

Instructions

* Using the starter file as a guide, complete the following steps:

* 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, which is at latitude 43.77°.

* Use the API to determine the actual temperature in Florence.

Bonus

If you finish early, feel free to try to predict the temperatures of other cities.

Activity 8

exception_example

Instructor Demo

```
# Try to access key that doesn't exist
# "Catching" the error lets the rest of our code execute
```

Activity 9

making_exceptions_solution

Your assignment is to get the last line to print without changing any
of the code below. Instead, wrap each line that throws an error in a
try/except block.

Making Exceptions

In this activity, you will create an application that uses `try` and `except` to resolve a number of errors.

Instructions

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

* Each `except` block should handle the specific error that will occur.

* Add a `print` statement under the `except` block to log the error.

Activity 10

api_exceptions_solution

#Dependencies
List of character
Set url for API
Set empty lists to hold characters height and mass
Loop through each character
    # Create search query, make request and store in json

```
    # Try to grab the height and mass of characters if they are available in the Star Wars API
    # Handle exceptions for a character that is not available in the Star Wars API
    # Append null values
# Create DataFrame
```

```
# API Exceptions

Not every call placed to an API will return a result. In this activity, you will use `try` and `except` to handle errors from API calls.

## Instructions

* Loop through the characters in the list, and send a request to the Star Wars API.

* Create a `try` clause and an `except` clause. In the `try` clause, append the height, mass, and character name that is available in the Star Wars API to their respective lists. If the character is not available in the Star Wars API, use the `except` clause to print a message and `pass`.

* Create a DataFrame from the results.
```

Activity 11

world_bank_api_solution

```
# Dependencies
# Get country information in JSON format
# First element is general information, second is countries themselves
# Report the names
```

Instructor Demo

Activity 12

two_calls_solution

Dependencies
Get the list of lending types the world bank has
Next, determine how many countries fall into each lending type.
Hint: Look at the first element of the response array.
Print the number of countries of each lending type

Lending Types

In this activity, you'll practice making two API calls in sequence. The second API call will depend on the response from the first.

Instructions

* Retrieve a list of the lending types that the world bank keeps track of, and extract the ID key for each one.

* Next, determine how many countries are categorized under each lending type. Use a dictionary to store this information.

  * This data is stored as the first element of the response array.