

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

Tuesday, October 17, 2023 6:11 PM

Class Objectives

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

 Load JSON from API responses into Pandas DataFrames

 Use `'try'` and `'except'` to resolve missing key values without terminating the code

 Use linear regression to predict the 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

Activity: JSON Traversal

Instructions	Load the provided JSON.
	Retrieve the video's title.
	Retrieve the video's rating.
	Retrieve the link to the video's thumbnail.
	Retrieve the number of views for the video.

```
{
  "apiVersion": "2.0",
  "data": {
    "updated": "2010-01-07T19:58:42.949Z",
    "totalItems": 800,
    "startIndex": 1,
    "itemsPerPage": 1,
    "items": [
      {
        "id": "hYB0mn5zh2c",
        "uploaded": "2007-06-05T22:07:03.000Z",
        "updated": "2010-01-07T13:26:50.000Z",
        "uploader": "GoogleDeveloperDay",
        "category": "News",
        "title": "Google Developers Day US - Maps API Introduction",
        "description": "Google Maps API Introduction ...",
        "tags": [
          "GDD07", "GDD07US", "Maps"
        ],
        "thumbnail": {
          "default": "http://i.ytimg.com/vi/hYB0mn5zh2c/default.jpg",
          "hqDefault": "http://i.ytimg.com/vi/hYB0mn5zh2c/hqdefault.jpg"
        },
        "player": {
          "default": "http://www.youtube.com/watch?v=u003dhYB0mn5zh2c"
        }
      }
    ]
  }
}
```

Activity 1

Json traveleres

0:16:00

We import dependensin

what is "os"

We import and load json

We separtate by file types

We access the list

what is a dictioany

what is a list

ngResponses | [① README.md](#) ~/.../04-Stu_FarFarAway-APIData | [① README.md](#) ~/.../12-Stu_RetrieveArticles | [① README.md](#) ~/.../11-Ins_NYTAPI | [① README](#)

Users > sigmanationn > Desktop > Analysis Project > UC Berkley Class > Moduel 6 Python API's > 06-Python-APIs > 2 > Activities > 01-Stu_JSONTraversReview > [① README](#)

```
1 # JSON Traversal Review
2
3 This activity is an opportunity to practice loading and parsing JSON in Python.
4
5 ## Instructions
6
7 * Load the provided JSON.
8
9 * Retrieve the video's title.
10
11 * Retrieve the video's rating.
12
13 * Retrieve the link to the video's first tag.
14
15 * Retrieve the number of views for the video.
16
17 ## References
18
19 Data Source:
20 Data for this dataset was generated by edX Boot Camps LLC, and is intended for educational purposes only.
21
22 ---
23
24 © 2022 edX Boot Camps LLC. Confidential and Proprietary. All Rights Reserved.
25
```

Activity 2

00:41:00

Request review

Import request dependence

We create a JSON response and pretty print the response

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

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

Users > sigmanationn > Desktop > Analysis Project > UC Berkley Class > Module 6 Python API's > 06-Python-API's > 2 > Activities > 02-Stu_RequestReview > [README.md](#) # Requests & Responses

1 # Requests & Responses
2
3 This activity provides practice making requests, converting the response to JSON, and then manipulating the result with Python.
4
5 ## Instructions
6
7 * 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.
8
9 * Print the JSON representations of the first and last posts.
10
11 * Print number of posts received.
12
13 ## References
14
15 Data Source: Mockaroo, LLC. (2021). Realistic Data Generator. [<https://www.mockaroo.com/>] (<https://www.mockaroo.com/>)
16
17 ---
18
19 © 2022 edX Boot Camps LLC. Confidential and Proprietary. All Rights Reserved.
20

Activity 3

1:14:00

03-Ins_OpenWeatherRequest

Import 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

In this activity, you will work with the OpenWeatherMap API to create an application that provides the user with the current temperature in Burundi's largest city.

(Instructions sent via Slack.)

Suggested Time:

15 minutes

12

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 call it as a dependency) (to referecen later)

```
① README.md ~/.../01-Stu_JSONTraversalReview ② README.md ~/.../02-Stu_RequestReview ③ README.md ~/.../03-Ins_OpenWeatherRequest ④ README.md ~/.../04-  
Users > sigmanationn > Desktop > Analysis Project > REPOS > UCB-VIRT-DATA-PT-05-2023-U-LOLC > 01-Lesson-Plans > 06-Python-APIs > 2 > Activities > 04-Stu_BurundiWeatherA  
1 # Weather in Bujumbura  
2  
3 This activity gives students practice with making API calls and handling responses.  
4  
5 ## Instructions  
6  
7 * Save all of your "config" information—i.e., your API key; the base URL; etc.—before moving on.  
8  
9 * Build your query URL.  
10  
11 * **Hint:** Check the documentation to figure out how to request temperatures in Celsius.  
12  
13 * Make your request, and save the API response.  
14  
15 * Retrieve the current temperature in Bujumbura from the JSON response.  
16  
17 * Print the temperature to the console.  
18  
19 ## Bonus  
20  
21 * Augment your code to report the temperature in both Fahrenheit and Celsius.  
22  
23 * **Note:** Don't forget to change the API key in config.py!  
24  
25 ---  
26  
27 © 2022 edX Boot Camps LLC. Confidential and Proprietary. All Rights Reserved.  
28
```

Activity 05

05-Ins_OpenWeatherDataFrame

1:24:00

We use json and urls to make tables and data frames



Instructor Demonstration

OpenWeatherMap DataFrame

15

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?

16

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]

17

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()
```

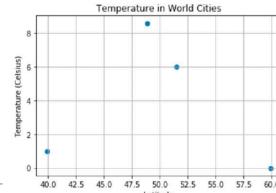
	city	lat	temp
0	Paris	48.86	8.59
1	London	51.51	6.00
2	Oslo	59.91	0.00
3	Beijing	39.91	1.00

```
# 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()
```



18



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

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 sent via Slack.)

Suggested Time:

15 minutes

19

Activity 6

06-Stu_TVRatingsDataFrame

We import json url into a dataframe and out put list

```
Users > sigmamonn > Desktop > Analysis Project > REPOS > UCB-VIRT-DATA-PT-05-2023-U-LOLC > 01-Lesson-Plans > 06-Python-APIs > 2 > Activities > 06-Stu_TVRatingsDataFrame > README.md
1 # TV Ratings
2
3 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.
4
5 ## Instructions
6
7 * You may use the list of TV shows provided in the starter file or create your own.
8
9 * Request information on each TV show from the [TVmaze API's Show Search endpoint](https://www.tvmaze.com/api#show-search)
10
11 * Store the name and rating information into lists.
12
13 * Store this data in a dictionary, and use it to create a Pandas DataFrame.
14
15 * Use Pandas to create a bar chart comparing the ratings of each show.
16
17 ---
18
19 © 2022 edX Boot Camps LLC. Confidential and Proprietary. All Rights Reserved.
20
```

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 retrieve rating information, creates a Pandas Dataframe, and then visually displays the data.

(Instructions sent via Slack.)

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.

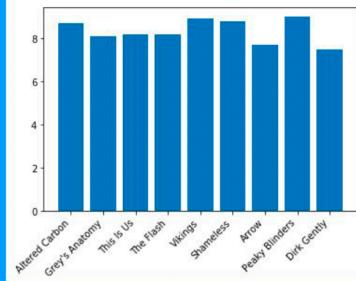
Request information on each TV show from <https://www.tvmaze.com/api#show-search>.

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 Statistics
1  # Weather Statistics
2
3  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.
4
5  ## Instructions
6
7  * Using the starter file as a guide, complete the following steps:
8
9  * Create a scatter plot of Temperature vs. Latitude.
10
11 * Perform linear regression.
12
13 * Create a line equation for the regression.
14
15 * Create a scatter plot with the linear regression line.
16
17 * Predict the temperature of Florence, which is at latitude 43.77&deg;.
18
19 * Use the API to determine the actual temperature in Florence.
20
21  ## Bonus
22
23 If you finish early, feel free to try to predict the temperatures of other cities.
24
25  ## Hint
26
27 If you need help, revisit the material on statistics from Unit 5.
28
29 ---
30
31 © 2022 edX Boot Camps LLC. Confidential and Proprietary. All Rights Reserved.
32
```

Activity 08

08-Ins_ExceptionHandling





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

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.")
```



```
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:  
    students["Mary"]
```


like `try` and `except`.

 Python will "try" to run the code.

 If the code throws an error or exception, the code in the `except` block is executed.

```
student[s] may ]  
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!



Activity: Making Exceptions

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

(Instructions sent via Slack.)

Suggested Time:

5 minutes

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

```
pynb README.md ~.../06-Stu_TVRatingDataFrames README.md ~.../07-Stu_Weather_Stats README.md ~.../08-Ins_ExceptionHandling README.md ~.../10-Stu_API_Exceptions x
Johnn > Desktop > Analysis Project > REPOS > UCB-VIRT-DATA-PT-05-2023-U-LOLC > 01-Lesson-Plans > 06-Python-APIs > 2 > Activities > 10-Stu_API_Exceptions > README.md > # API Exceptions #
```



```
1  ## API Exceptions
2
3  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.
4
5  ## Instructions
6
7  * Loop through the characters in the list, and send a request to the Star Wars API.
8
9  * 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
10 respective lists. If the character is not available in the Star Wars API, use the `except` clause to print a message and `pass`.
11
12
13
14
15  © 2022 edX Boot Camps LLC. Confidential and Proprietary. All Rights Reserved.
16
```

Activity 11

2:33:00

11-Ins_WorldBankAPI

<https://datahelpdesk.worldbank.org/>



36

World Bank API

Up until now, we have been working with

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()
```

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 > https://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/SPANUSIFP9?date=200901:2010M9>
> <http://api.worldbank.org/v2/countries/chn/krs/indicators/SPANUSIZF8?date=2010Q1:2010Q3>

additionaly supports, year to date values (YTD). Useful for querying high frequency data

> <http://api.worldbank.org/v2/countries/chn/krs/indicators/SPANUSIZF8?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>

37

Activity: Two Calls

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

(Instructions sent via Slack.)

Suggested Time:

10 minutes

38

Activity 12

12-Stu_TwoCalls

```
Users > sigmanationn > Desktop > Analysis Project > REPOS > UCB-VIRT-DATA-PT-05-2023-U-LOLC > 01-Lesson-Plans > 06-Python-APIs > 2 > Activities > 12-Stu_Tw  
1   # Lending Types  
2  
3   In this activity, you'll practice making two API calls in sequence. The second API call will depend on the response from the first.
```



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

Activity: Two Calls



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

(Instructions sent via Slack.)

Suggested Time:

10 minutes

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.bcedx.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/>(<https://www.mockaroo.com/>)

Activity 3

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 response 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.

