Unit 6 Day 3 - Python APIs

Data Boot Camp

Lesson 6.3

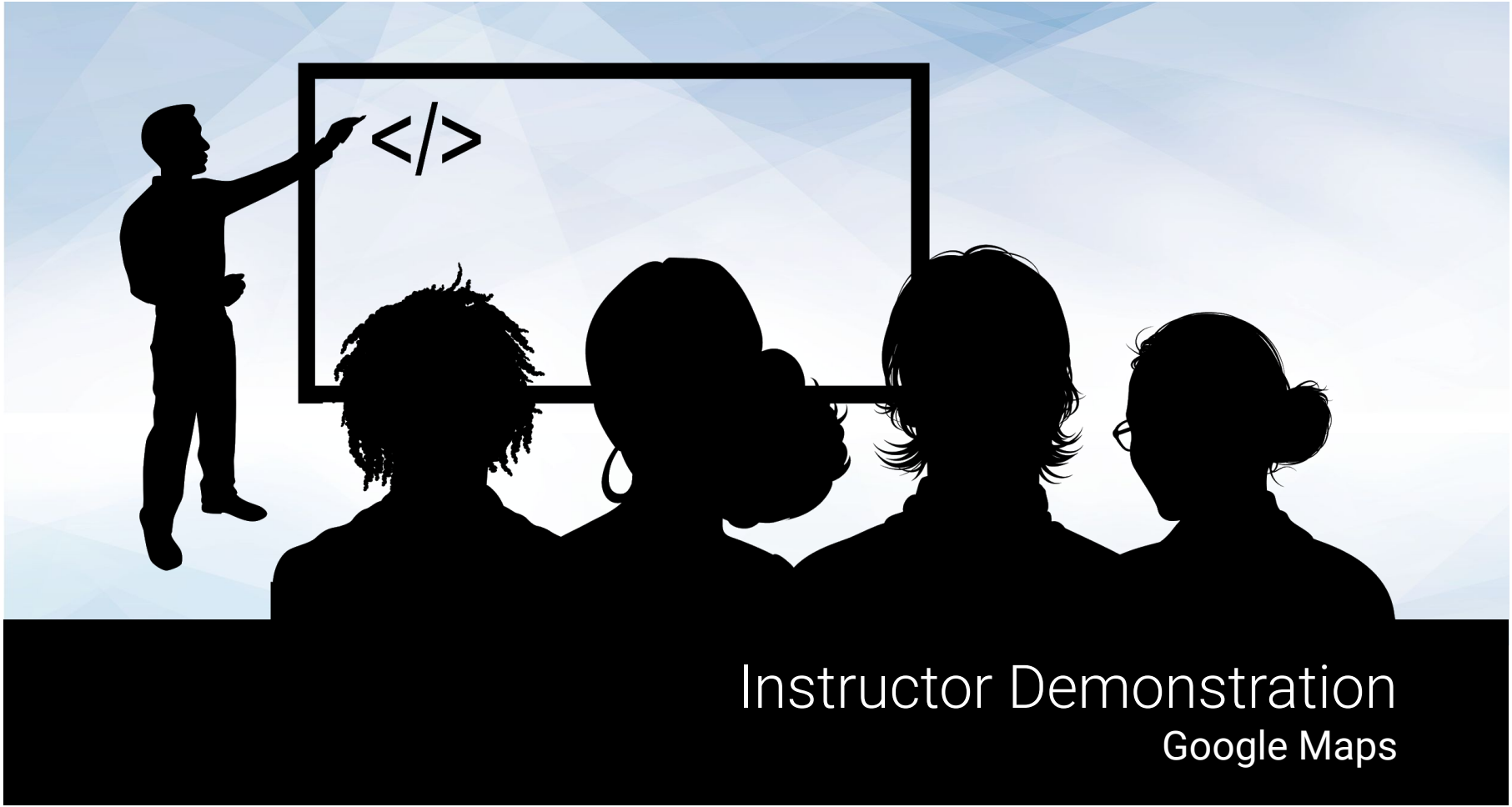
Class Objectives

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

Use the Google Maps and places API to obtain information about geographic areas

Use the Census API to get populations, average income and poverties rates of cities

Visually represent banking and income data using Jupyter Gmaps



Instructor Demonstration

Google Maps

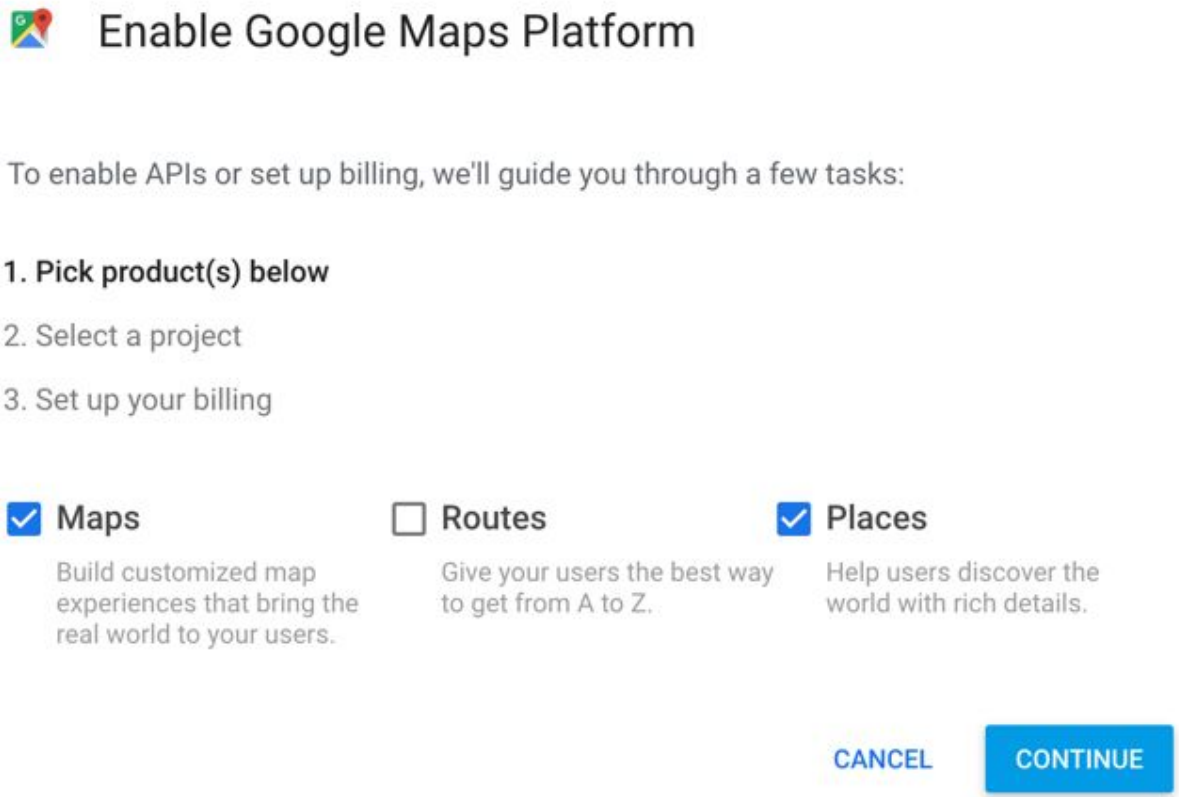
JSON Traversal Instructions

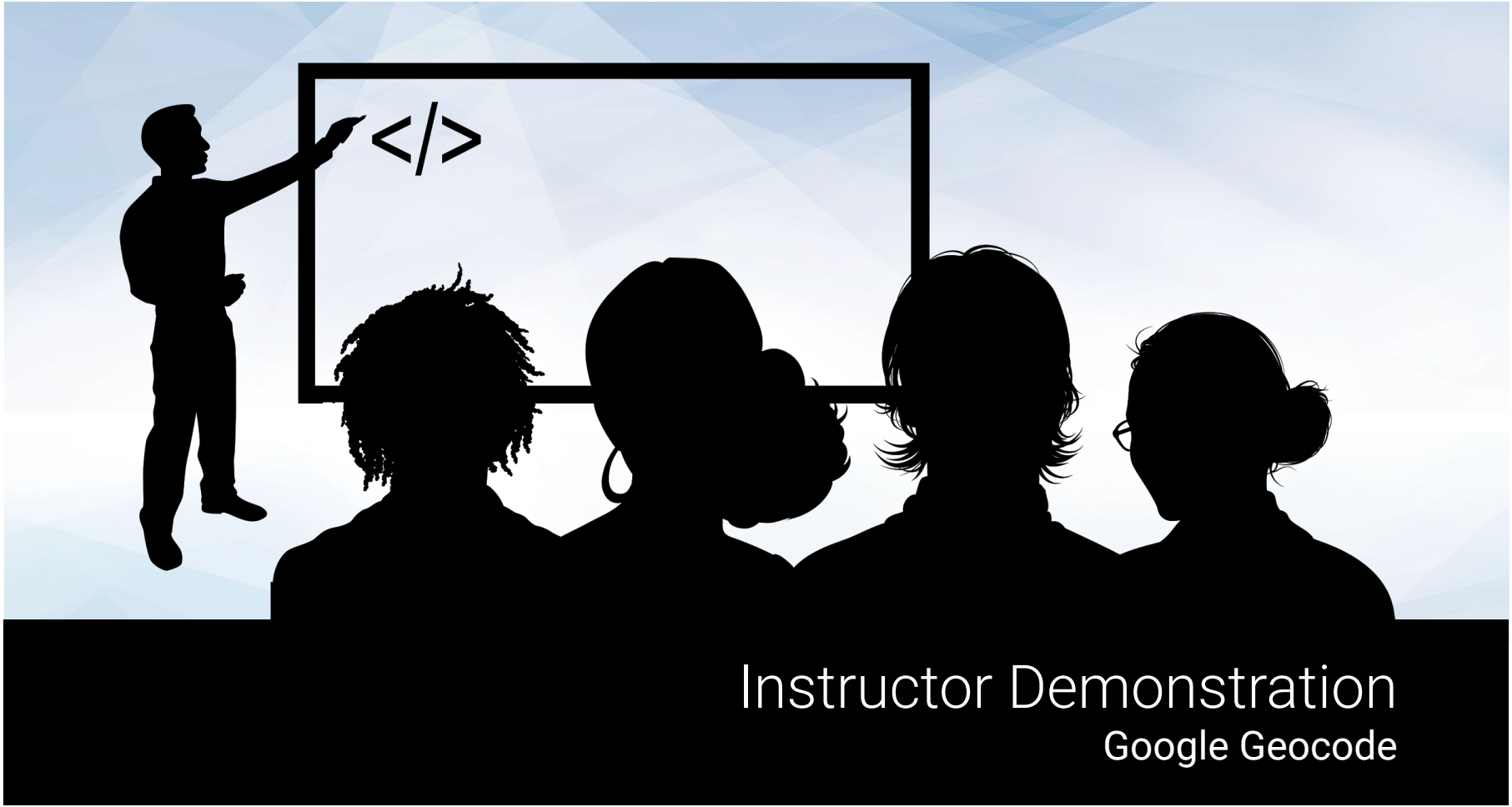
- Today’s class will cover Google Maps and Google Places API.
- These APIs allow developers to quickly convert locations into latitudinal and longitudinal coordinates, identify nearest restaurants to a given location, determine the distance between two points, and much more!

Obtaining API key

- Visit https://cloud.google.com/maps-platform/ and click **Get Started**
- Select Boxes for `Maps` and `Places`.
- Click **Create a New Project** and give a name.
- Click **Create Billing Account**
    - Google charges for services but a $200 credit is provided for the these services.
    - NOTE: charges above $200 will be charged to your personal account. Nothing in this class will go over that credit. Alerts can be setup under the billing menu.
- Consult the **Capping_Queries.md** to set query limits for API usage

Obtaining API key





Instructor Demonstration

Google Geocode

Google Geocode

- Google Maps Geocoding API turns addresses into latitudinal and longitudinal coordinates.
- This process is referred to **geocoding.**
- Many applications only understand locations formatted in terms of latitude/longitude.


```
"formatted_address" : "1600 Amphitheatre Parkway, Mountain View, CA 94043, USA",
"geometry" : {
   "location" : {
      "lat" : 37.4224764,
      "lng" : -122.0842499
   },
   "location_type" : "ROOFTOP",
   "viewport" : {
      "northeast" : {
         "lat" : 37.4238253802915,
         "lng" : -122.0829009197085
      },
      "southwest" : {
         "lat" : 37.4211274197085,
         "lng" : -122.0855988802915
      }
   }
```

Google Geocode

```
# Dependencies
import requests
import json

# Google developer API key
from config import gkey

# Target city
target_city = "Boise, Idaho"

# Build the endpoint URL
target_url = "https://maps.googleapis.com/maps/api/geocode/json?" \
    "address=%s&key=%s" % (target_city, gkey)

# Print the assembled URL, avoid pushing this print out to github for key security
print(target_url)
```

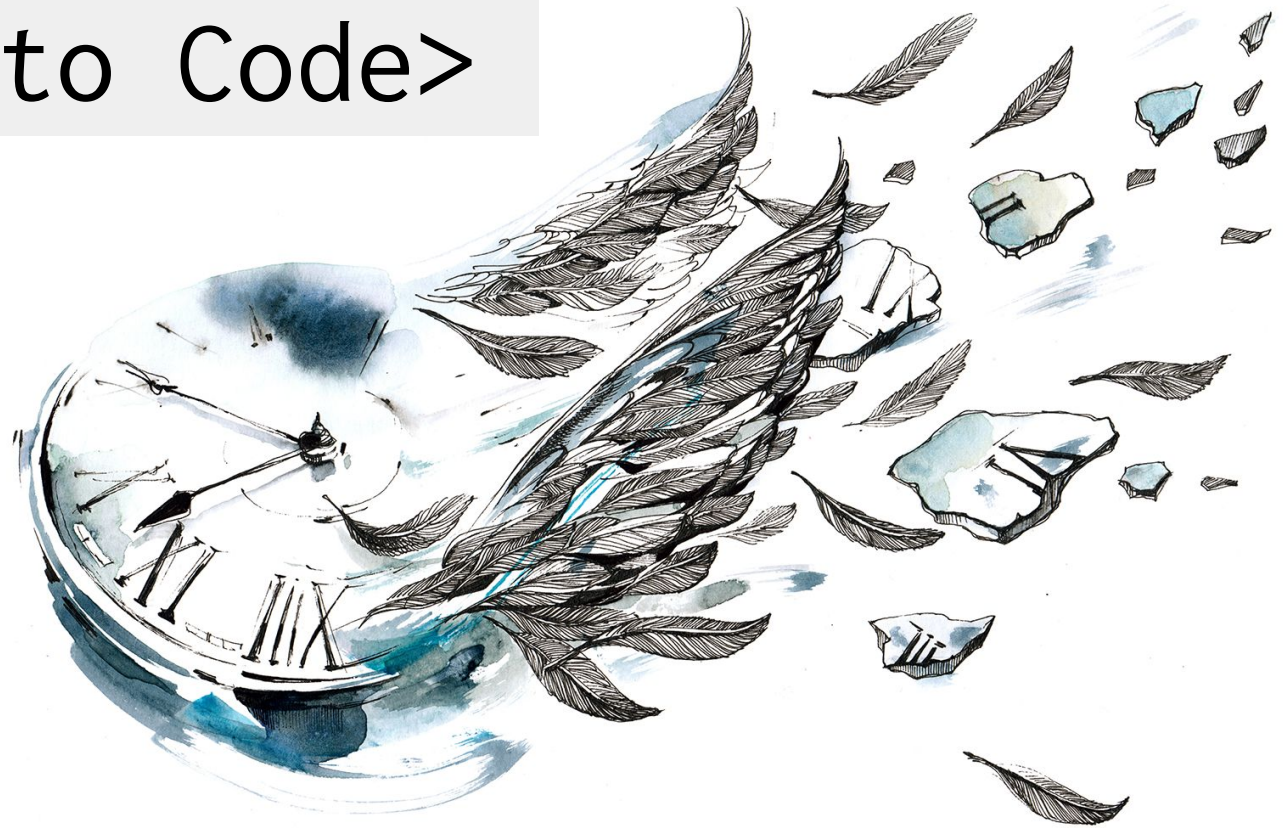
Google API Key saved in config.py

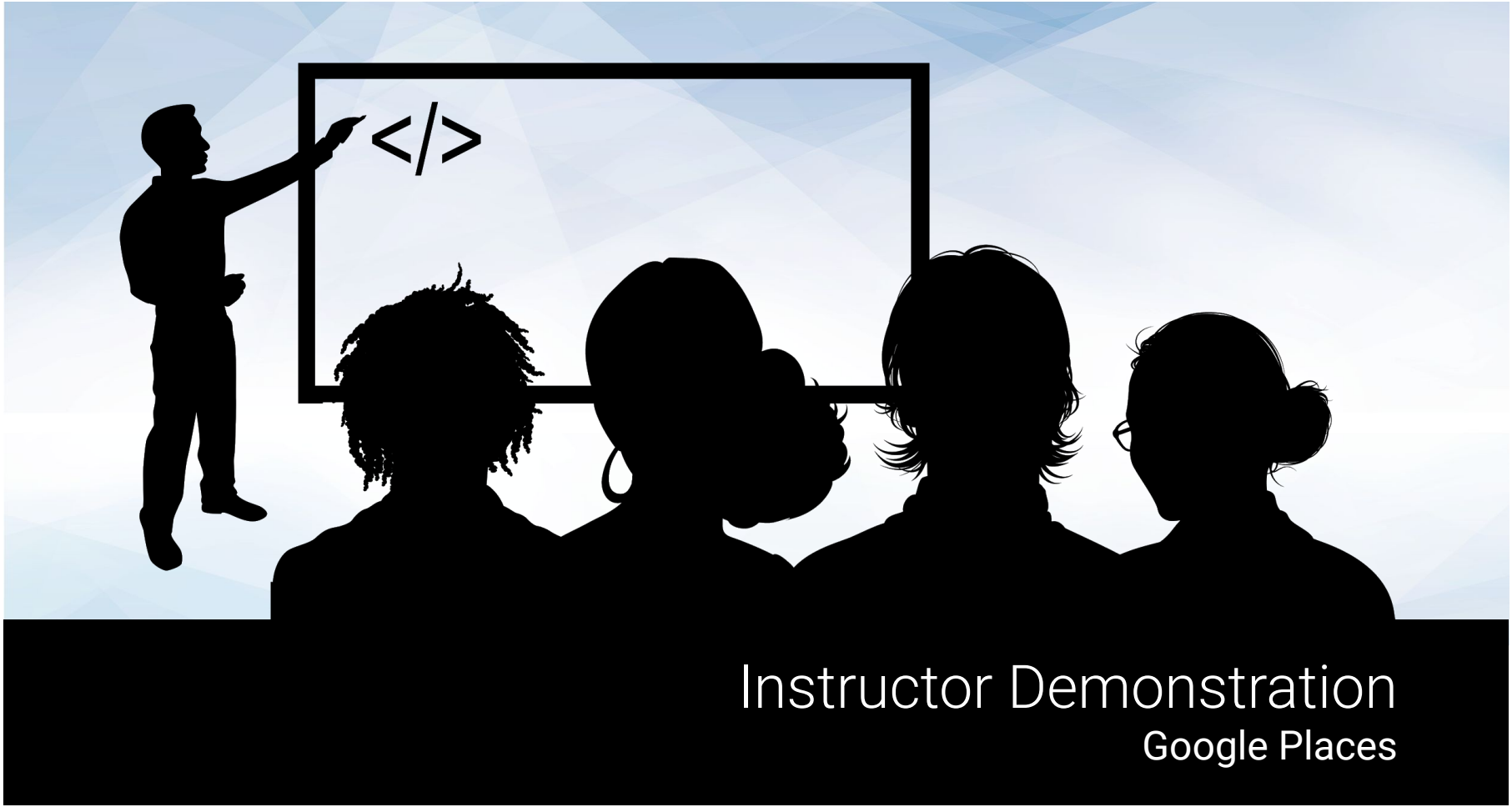
Endpoint URL

- View the documentation

  https://developers.google.com/maps/documentation/geocoding/start

<Time to Code>





Instructor Demonstration

Google Places

Google Places

Nearby Search

- Searches for places within an area.

```
https://maps.googleapis.com/maps/api/place/nearbysearch/output?parameters
```

Text Search

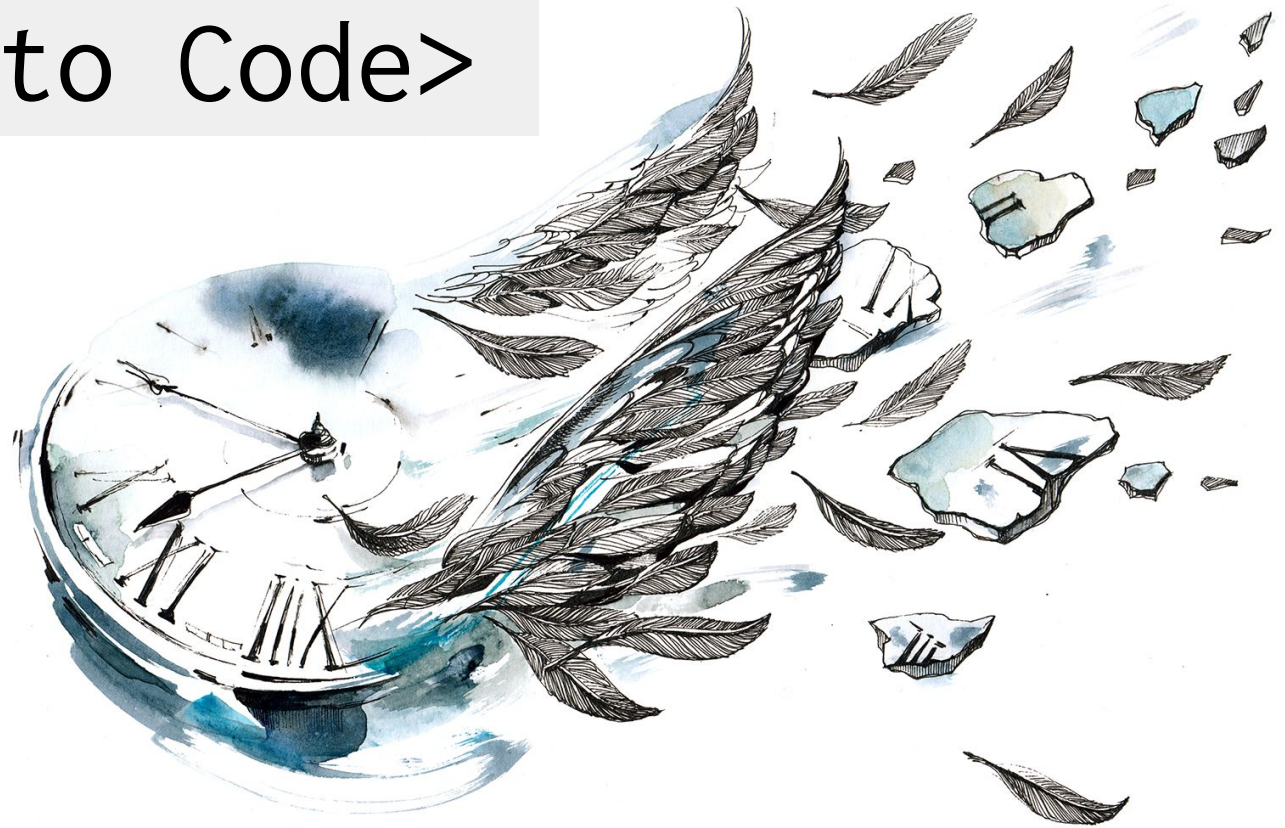
- Returns info about a set of places based on a string

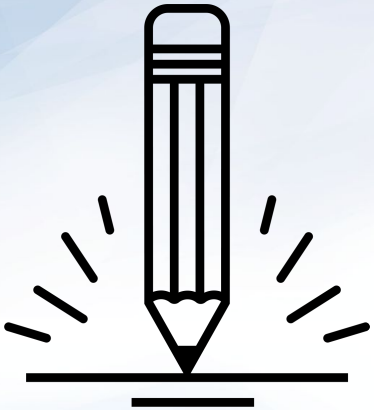
```
https://maps.googleapis.com/maps/api/place/textsearch/output?parameters
```

- Searches for place information based on category.

```
https://maps.googleapis.com/maps/api/place/findplacefromtext/output?parameters
```

<Time to Code>





Activity: Google Drills

In this activity, you will generate code that makes calls to both the Google Places and Google Geocoding APIs.

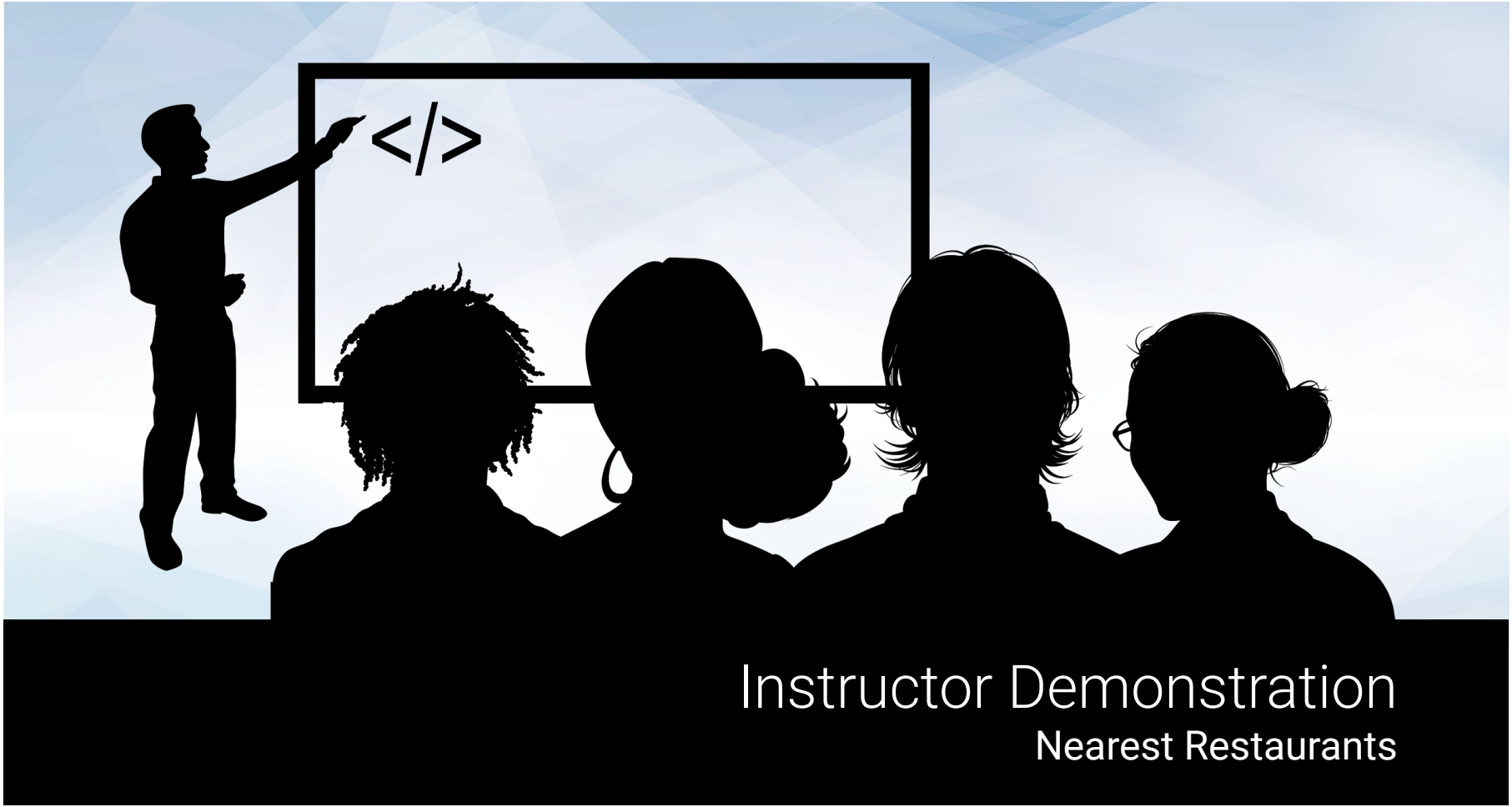
(Instructions sent via Slack.)

Suggested Time:
15 Minutes

Google Drills Instructions

- Complete each of the six drills articulated in the code provided. Feel encouraged to look back at the previous examples but know that you will have to consult the Google API documentation.
- **HINTS:**
  - Be sure to read the **Google Geocoding Documentation** and the **Google Places Documentation**
  - See the **Capping Queries** document to set usage limits on your API calls.

Time’s Up! Let’s Review.



Instructor Demonstration

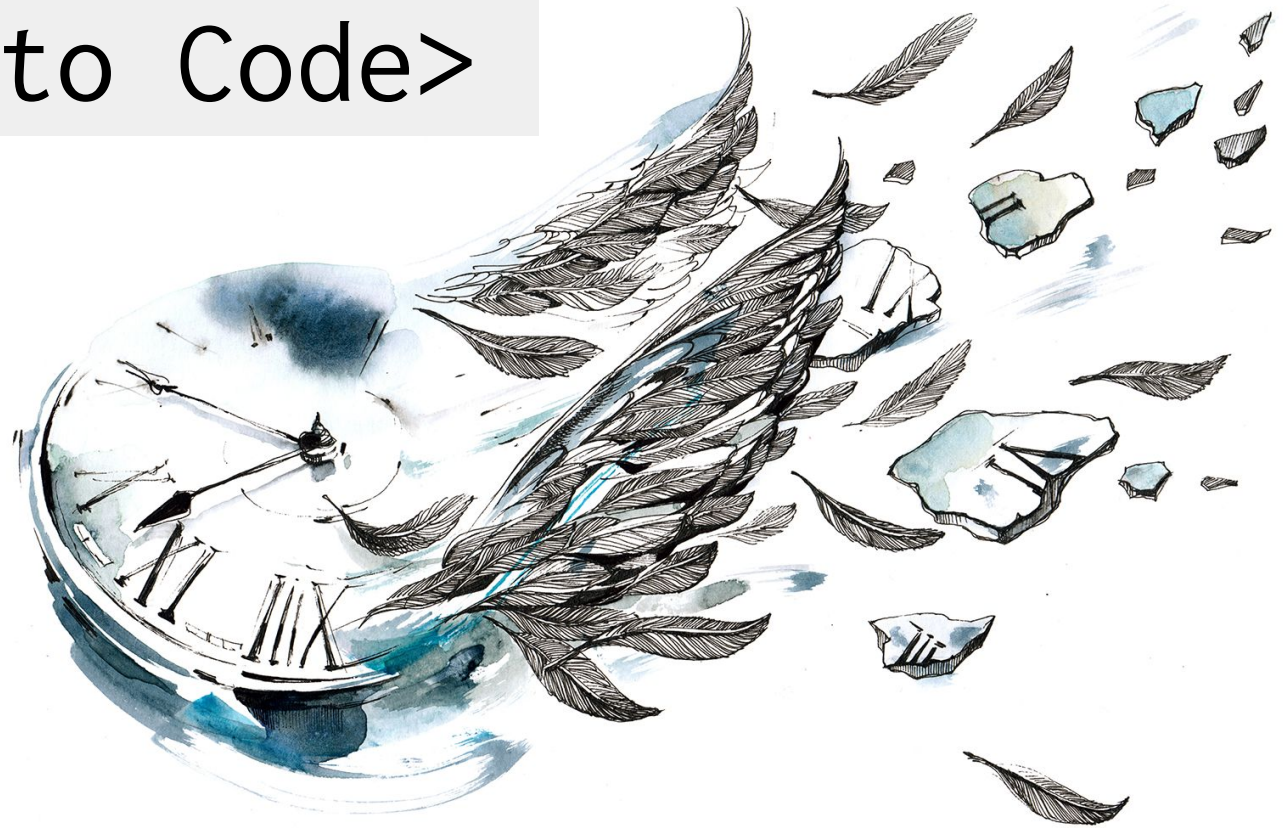
Nearest Restaurants

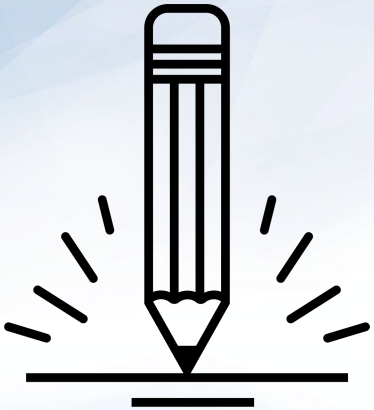
Another Means of Traversing JSONs

- The panda's method iterrows() returns an index number and the contents of each row.
  - Rows can be accessed row['column label'].
- With each iteration the keyword is overwritten
- The get() method retrieves results
  - If the result exists the value is retrieved.
  - If not, then None is stored.
  - Similar to try/except.
- The try/except clause is used with loc to store the responses.

```
for index, row in types_df.iterrows():

    # get restaurant type from df
    restr_type = row['ethnicity']

    # add keyword to params dict
    params['keyword'] = restr_type

    # assemble url and make API request
    print(f"Retrieving Results for Index {index}: {restr_type}.")
    response = requests.get(base_url, params=params).json()

    # extract results
    results = response['results']

    try:
        print(f"Closest {restr_type} restaurant is {results[0]['name']}.")

        types_df.loc[index, 'name'] = results[0]['name']
        types_df.loc[index, 'address'] = results[0]['vicinity']
        types_df.loc[index, 'price_level'] = results[0]['price_level']
        types_df.loc[index, 'rating'] = results[0]['rating']

    except (KeyError, IndexError):
        print("Missing field/result... skipping.")
```

<Time to Code>





Activity: Google Complex (Airport)

In this activity, you will obtain the rating of every airport in the top 100 metropolitan areas according to Google Users.

(Instructions sent via Slack.)

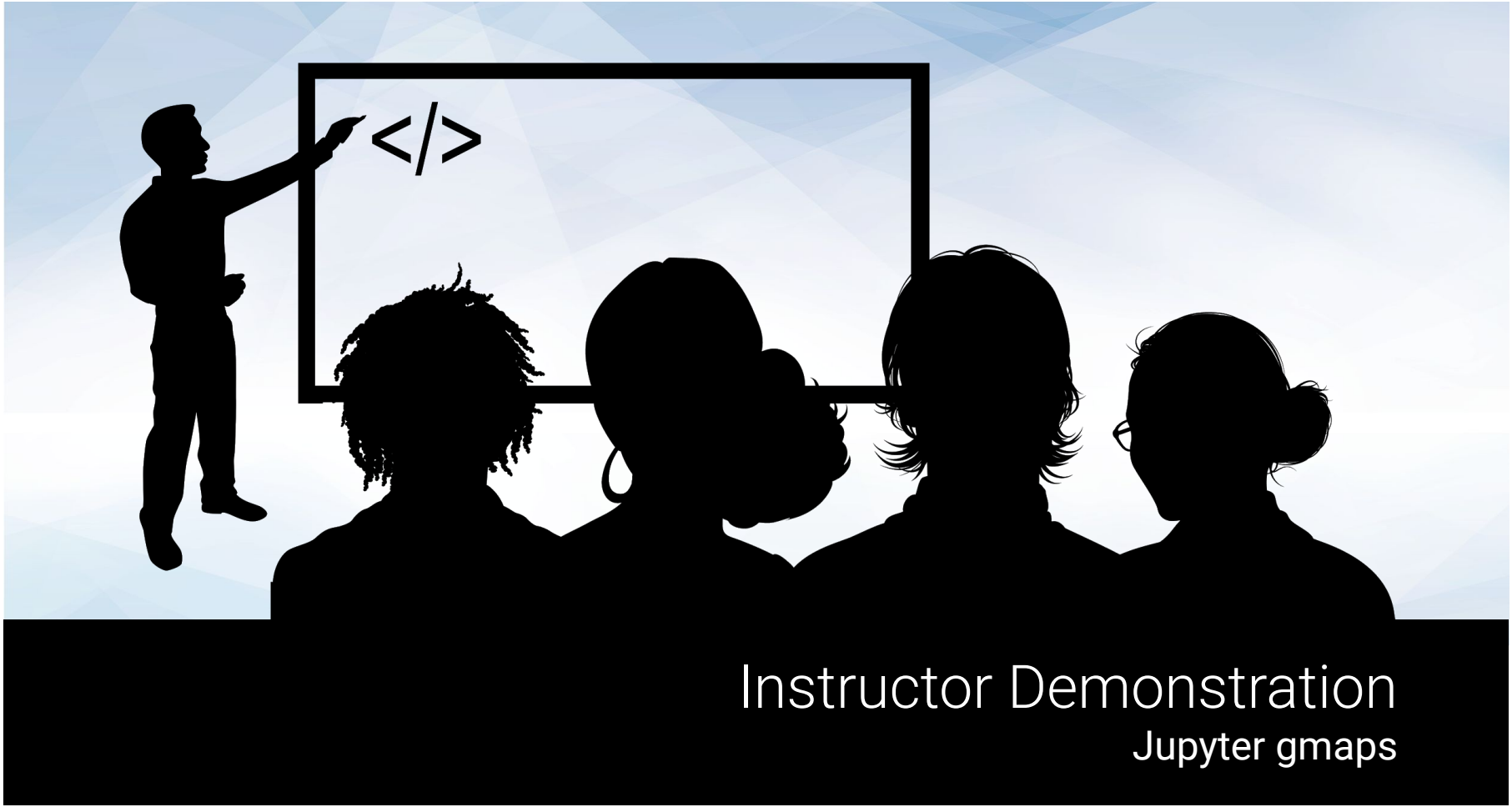
Suggested Time:
20 Minutes

Google Complex (Airport) Instructions

- Using Airport_Ratings.ipynb as a starting point, utilize the Google Geocoding API, the Google Places API, and Python/Jupyter to create a script that lists the "Airport Rating" of the major "International Airport" in each of the top 100 metropolitan areas found in Cities.csv.
- Your final `ipynb` file should contain each of the following headers:
  - `City`, `State`, `Lat`, Lng`, `Airport Name`, `Airport Address`, Airport Rating
- **Hints:**
  - You will need to obtain the lat/lng of each airport prior to sending it through the Google Places API to obtain the rating.
  - When using the Google Places API, be sure to use the term "International Airport" to ensure that the data received is for the major airport in the city and not a regional airport.
  - Use a try-except to skip airports for which there are no Google user's ratings.

Time’s Up! Let’s Review.

Take a Break!

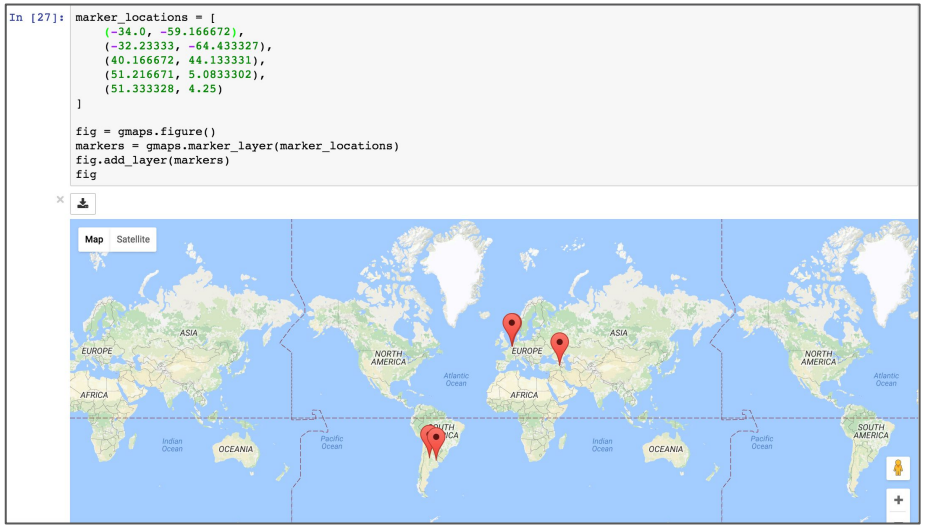


Instructor Demonstration

Jupyter gmaps

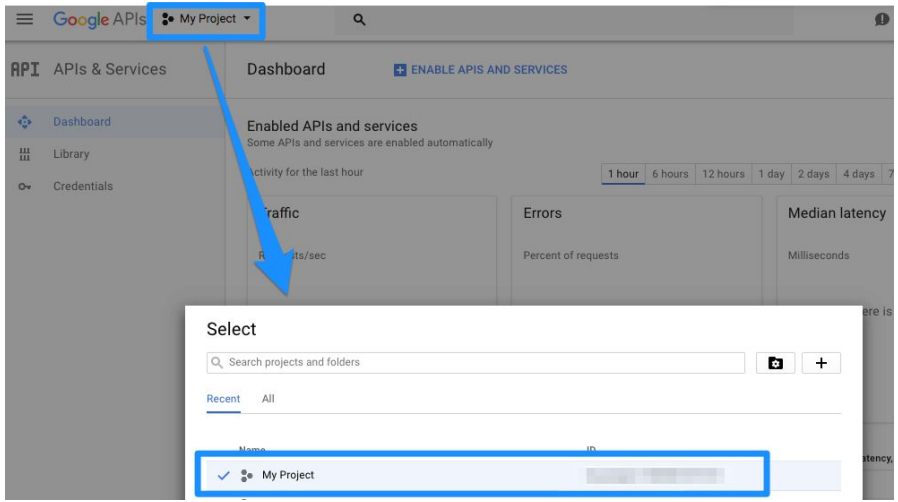
Gmaps

- A Jupyter plugin
- Allows embedding of Google maps into notebooks.
- Grants ability to visualize multiple map layers.
- Allows for customization of maps



Gmaps Installation

- Login in to the console and select project created earlier.
- Click library on side panel and search for Maps Javascript API.
- Enable API.
- Install the plugin: conda install -c conda-forge gmaps



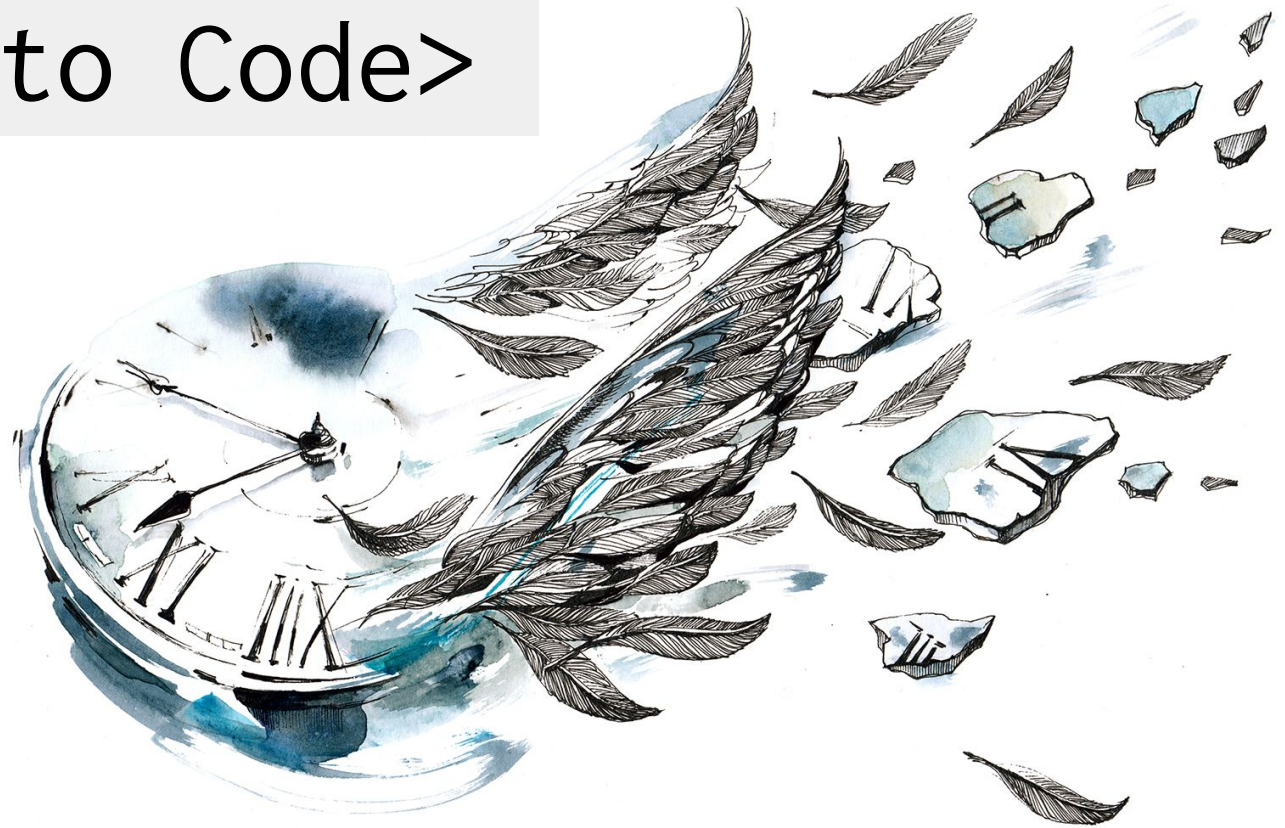
Running Gmaps

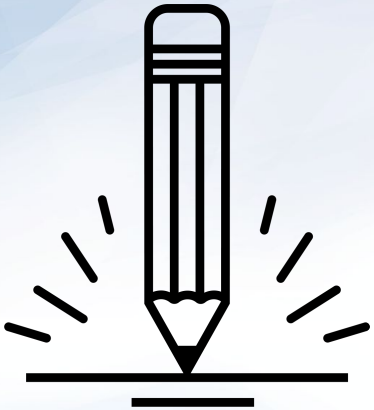
- Store API Key and configure gmaps.
- Add a marker_layer which consists of a list of tuples.
- The maps will automatically adjust the view as data is added.
- Layout can be pre-configured. By storing any of the values in a dictionary
  - Width
  - Border
  - Padding
- Pass the layout as a parameter to figure()

```
import gmaps
from config import gkey

gmaps.configure(api_key=gkey)

fig = gmaps.figure()
```

```
import gmaps
gmaps.configure(api_key="your_key")

figure_layout = {
  'width': '400px',
  'width': '300px',
  'border': '1px solid black',
  'padding': '1px'
}

fig = gmaps.figure(layout=figure_layout)
fig
```

<Time to Code>





Activity: Hot Airports

In this activity, you will create a heat map based on airport ratings.

(Instructions sent via Slack.)

Suggested Time:
15 Minutes

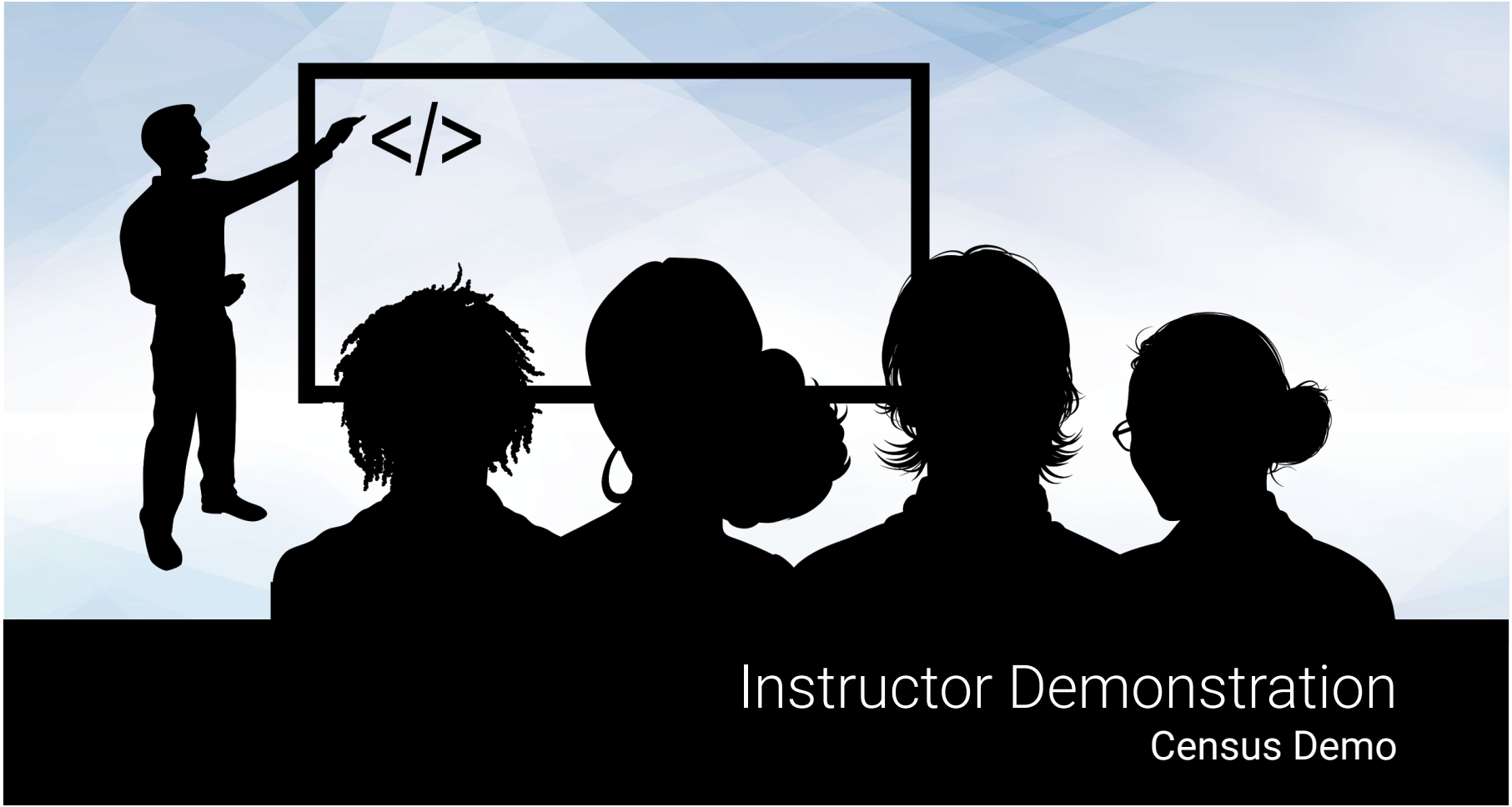
Gmap Installation

- If you haven't yet, navigate to your https://console.developers.google.com/ and enable the **Maps JavaScript API**.
- Install `gmaps`
- If it's not already running, start your virtual environment.
    - On Windows: `activate PythonData`
    - On Apple: `source activate PythonData`
- Then, from the command line run: `conda install -c conda-forge gmaps`

Hot Airports Instructions

- Using the starter notebook and the airport CSV in the Resources folder, create a heat map of the airports across the country.
    - Use the latitude and longitude to place the airport
    - Use the rating to weight the heat map.
    - Be sure to handle `NaN` values and data type in your Airport Ratings.
- Refer to the Jupyter Gmap documentation or instructions on how to implement heatmaps.
- **BONUS:** Explore Jupyter Gmap documentation to plot the map as two more types of maps.

Time’s Up! Let’s Review.



Instructor Demonstration

Census Demo

Census API

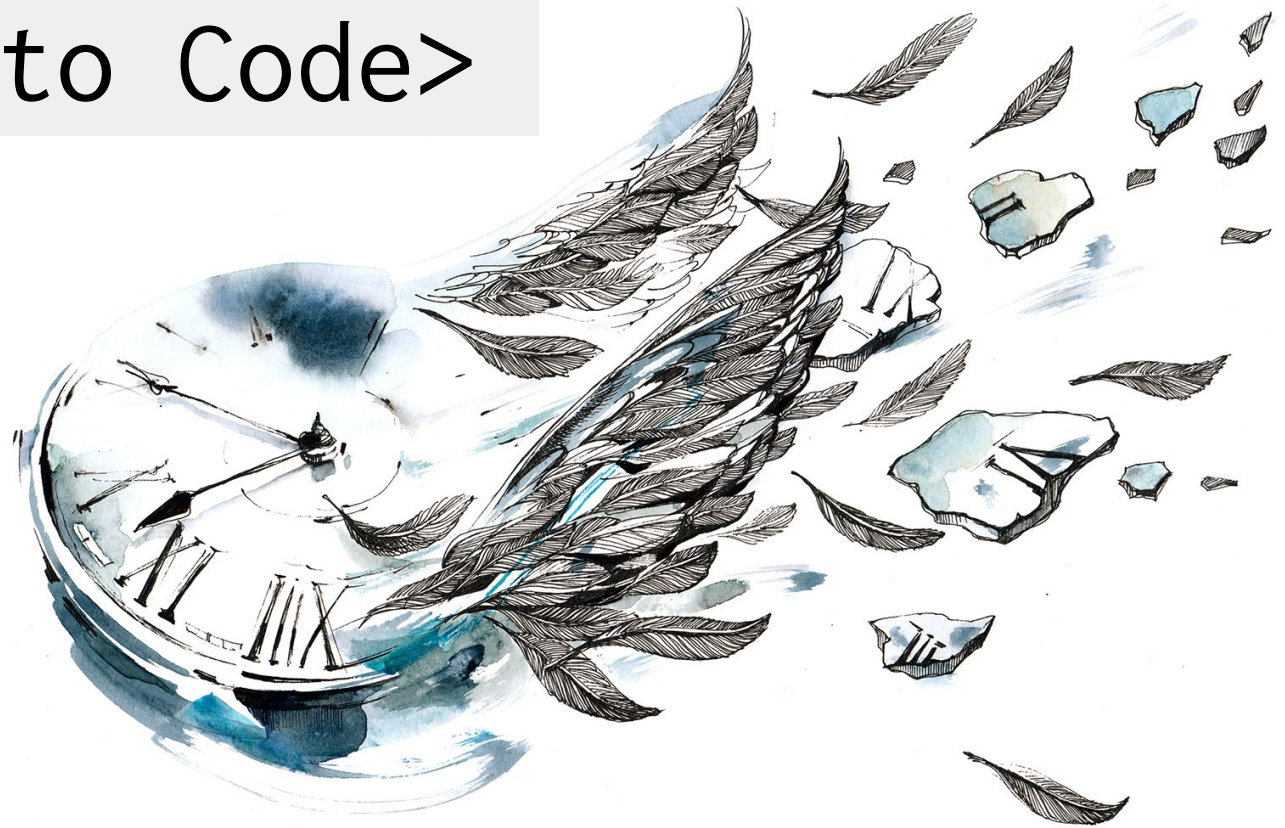
- Obtains an API key from http://www.census.gov/developers/
- Run pip install census in your environment
- the wrapper provides a fairly easy method of retrieving data from the 2013 census based on zip code, state, district, or county.
- Each census field (e.g. Poverty Count, Unemployment Count, Number of Asians, etc.) is denoted with a label like B201534_10E.
- The results are then returned as a list of dictionaries, which can be immediately converted into a DataFrame.

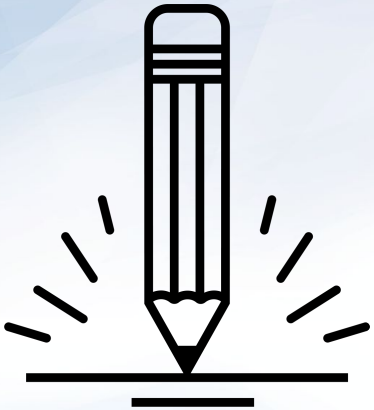
How We Will Use Census API

```
census_data = c.acs5.get(("NAME", "B19013_001E", "B01003_001E", "B01002_001E",
                          "B19301_001E",
                          "B17001_002E"), {'for': 'zip code tabulation area:*'})

# Convert to DataFrame
census_pd = pd.DataFrame(census_data)

# Column Reordering
census_pd = census_pd.rename(columns={"B01003_001E": "Population",
                                      "B01002_001E": "Median Age",
                                      "B19013_001E": "Household Income",
                                      "B19301_001E": "Per Capita Income",
                                      "B17001_002E": "Poverty Count",
                                      "NAME": "Name", "zip code tabulation area": "Zipcode"})

# Add in Poverty Rate (Poverty Count / Population)
census_pd["Poverty Rate"] = 100 * \
    census_pd["Poverty Count"].astype(
        int) / census_pd["Population"].astype(int)

# Final DataFrame
census_pd = census_pd[["Zipcode", "Population", "Median Age", "Household Income",
                       "Per Capita Income", "Poverty Count", "Poverty Rate"]]
```

- The c.asc5.get method grabs data on each field.
- Poverty Count is divided by Total Population to evaluate Poverty Count.
- US does not calculate Poverty Rate explicitly.

<Time to Code>





Activity: Census Activity

In this activity, you will utilize the Census API to obtain census data at a state level and visualize it with gmaps.

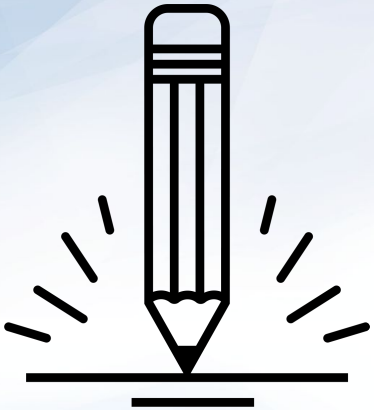
(Instructions sent via Slack.)

Suggested Time:
20 Minutes

Census Activity Instructions

- Using Census_States.ipynb as a reference, create a completely new script that calculates each of the following fields at the **state** level:
  - Population
  - Median Age
  - Household Income
  - Per Capita Income
  - Poverty Count
  - Poverty Rate
  - Unemployment Rate
- Next, read in the provided csv containing state centroid coordinates and merge this data with your original census data.
- With the coordinates now appended to the dataframe, you have the ability to add markers to the base map.
  - Use the 'Poverty Rate' column to create an `info_box` corresponding to each marker.

Time’s Up! Let’s Review.



Activity: Banking Deserts Heatmap

In this activity, you will create a data visualization to understand how prominent the "banking desert" phenomenon truly is.

(Instructions sent via Slack.)

Suggested Time:
20 Minutes

Banking Deserts Heatmap **Instructions**

- Using `gmap` create the following three figures:
  - A map with a `heatmap_layer` of the poverty rate for each city.
  - A map with a `symbol_layer` for the number of banks located at that city.
  - A map that includes both the poverty `heatmap_layer` and the bank `symbol_layer`.
- Print the summary statistics for Unemployment, Bank Count and Population.
- Create a scatter plot with linear regression for ****Bank Count**** vs. ****Unemployment Rate****.
- Plot the data points.
- Plot the linear regression line.
- Print the R square value.
- Write a sentences describing your finds. Were they what you expected? What other factors could be at play?

Time’s Up! Let’s Review.