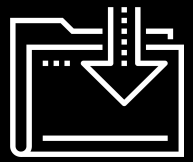
Python APIs: Day 3

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 APIs to obtain information about geographic areas

Use the Census API to get population counts, average income, and poverty rates of cities

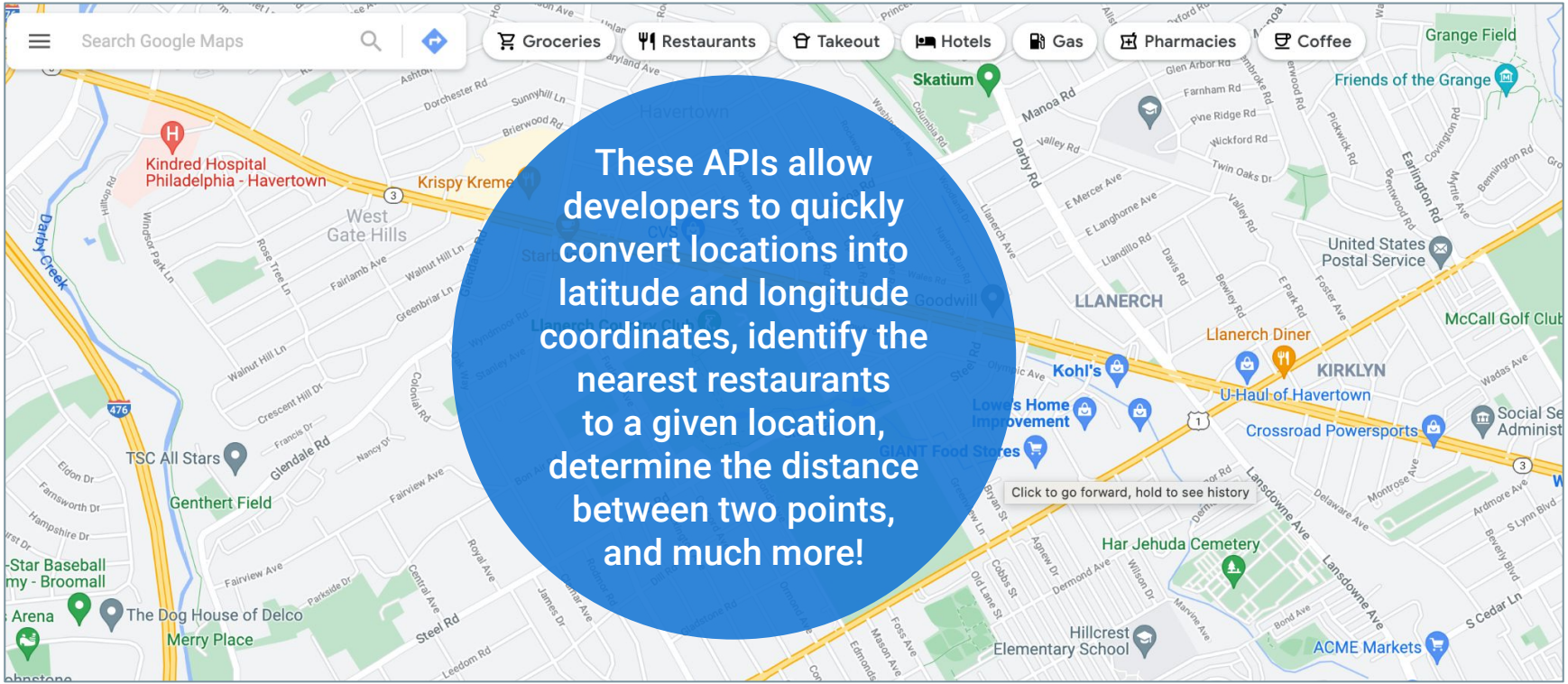
Visually represent banking and income data using Jupyter Gmaps

Instructor Demonstration

Google Maps

Google Maps and Google Places APIs

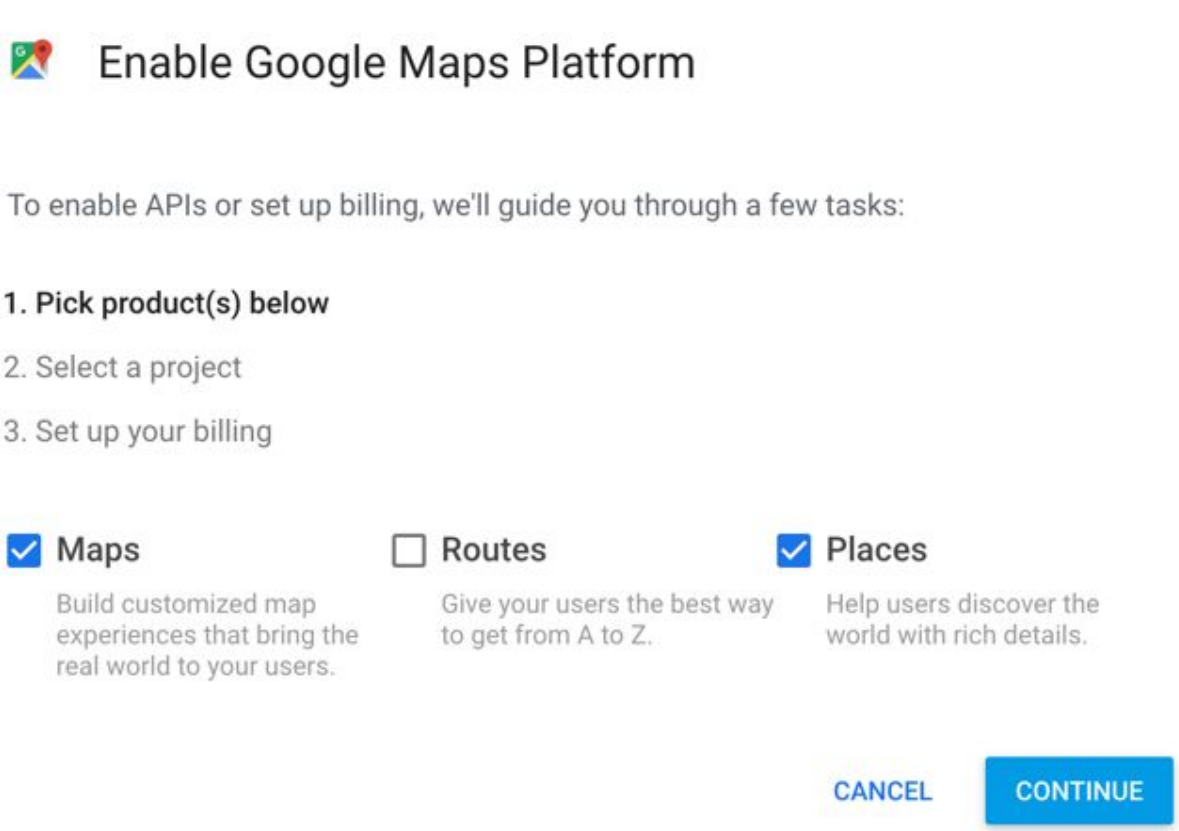
Today's class will cover the Google Maps and Google Places APIs.



Obtaining an API Key

Instructions	
	Visit https://cloud.google.com/maps-platform/, and click **Get Started**.
	Select the boxes for **Maps** and **Places**.
	Click **Create a New Project**, and assign a name.
	Click **Create Billing Account.**
	Retrieve the number of views for the video. • Google provides a $200 monthly credit, but charges for usage above $200 will be charged to your personal account. This class is designed to remain under the $200 monthly credit, and helpful alerts can be set up under the **Billing** menu.
	Consult `Capping_Queries.md` for more on setting query limits for API usage.

Obtaining an API Key



Instructor Demonstration

Google Geocode

Google Geocode

Google Maps' Geocoding API converts addresses into latitudinal and longitudinal coordinates.

This process is known as **geocoding**.

Many applications require locations to be formatted in terms of latitude and 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
      }
   }
```

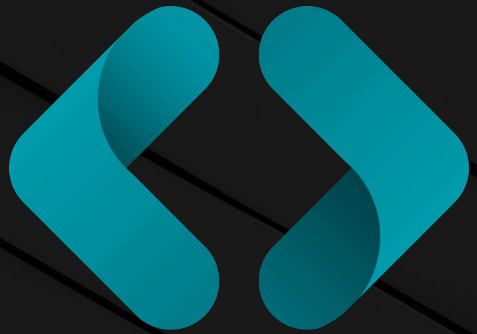
Google Geocode

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

```
# 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



Time to <code>

Instructor Demonstration

Google Places

Google Places

Nearby Search Searches for places within an area

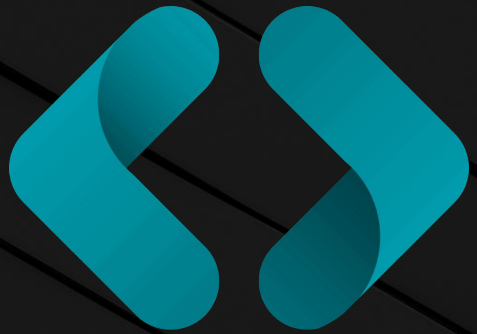
`https://maps.googleapis.com/maps/api/place/nearbysearch/outbput?parameters`

Text Search Returns info about a set of places based on a string

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

Place Search Searches for place information based on category

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

Activity: Google Drills

Instructions	Complete the six drills included in the provided code. You're encouraged to return to the previous examples, but know that you will have to consult the Google API documentation.
Hints	Make sure to read the **Google Geocoding Documentation** and the **Google Places Documentation**.
	Check the **Capping Queries** document to set usage limits on your API calls.

Time's Up! Let's Review.

Instructor Demonstration

Nearest Restaurants

Another Way to Traverse JSONs

The Pandas method `iterrows()` returns an index number and the contents of each row.

- Rows can be accessed using `row['column label']`.

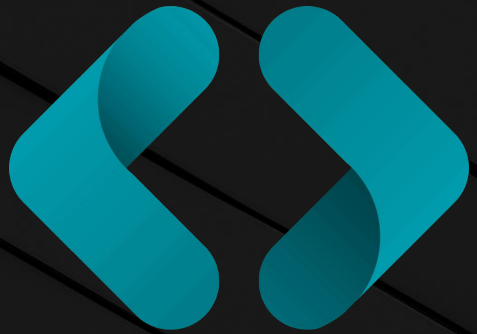
With each iteration, the keyword is overwritten.

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 Google user ratings for every airport in the top 100 metropolitan areas.

(Instructions sent via Slack.)

Suggested Time:

20 minutes

Activity: Google Complex (Airport)

Instructions	With `Airport_Ratings.ipynb` as your starting point, use 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 notebook 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 latitude `(lat)` and longitude `(lng)` of each airport prior to sending it through the Google Places API to obtain the rating.
	When using the Google Places API, make 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 ratings.

Time's Up! Let's Review.


Break

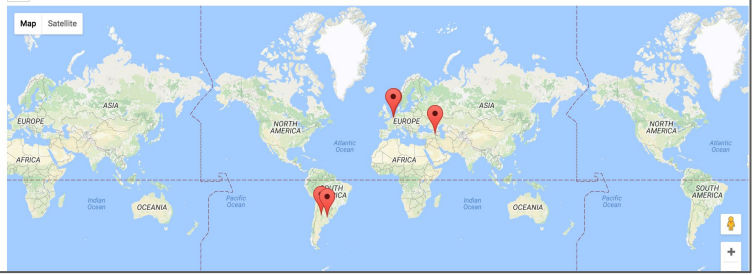
Instructor Demonstration

Jupyter Gmaps

Gmaps

- A Jupyter plugin
- Allows embedding of Google Maps into notebooks
- Enables visualization of multiple map layers
- Allows for map customization

```
In [27]: marker_locations = [
             (-34.0, -59.166672),
             (-32.23333, -64.433327),
             (40.166672, 44.133331),
             (51.216671, 5.0833302),
             (51.333328, 4.25)
         ]

         fig = gmaps.figure()
         markers = gmaps.marker_layer(marker_locations)
         fig.add_layer(markers)
         fig
```



Gmaps Installation

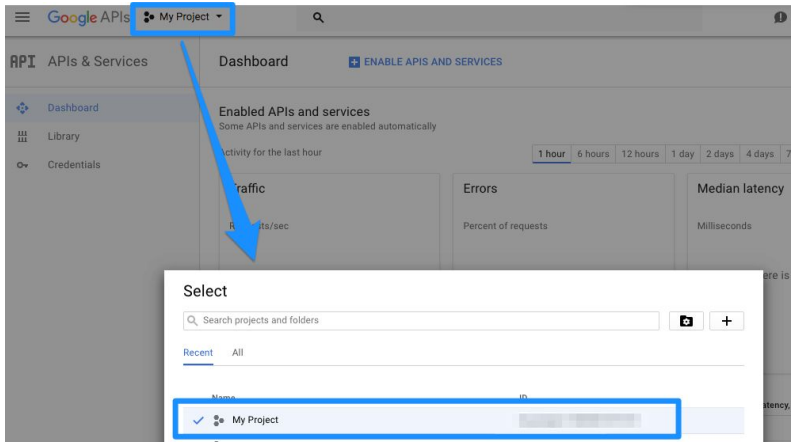
Log in to the console, and select the project created earlier.

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` that consists of a list of tuples.

The maps will automatically adjust the view
as data is added.

With each iteration, the keyword is overwritten.

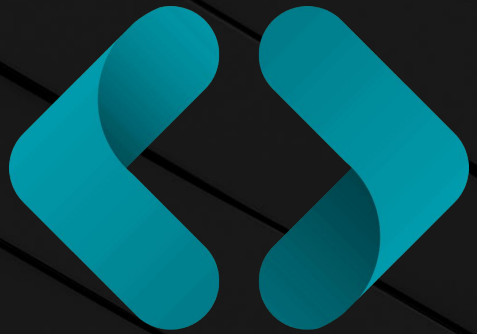
Layout can be preconfigured by storing any
of the values in a dictionary:

- 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 heatmap based on airport ratings.

(Instructions sent via Slack.)

Suggested Time:

15 minutes

Gmap Installation

Instructions	
	If you haven't already, navigate to 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.`

Activity: 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 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 using two other map types.

Time's Up! Let's Review.

Instructor Demonstration

Census Demo

Census API

Instructions:

- Obtain an API key from http://www.census.gov/developers/.
- Run `pip install` census in your environment.
- The wrapper provides an easy way to retrieve data from the 2013 Census based on zip code, state, district, or county.
- Each census field (for example, 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.

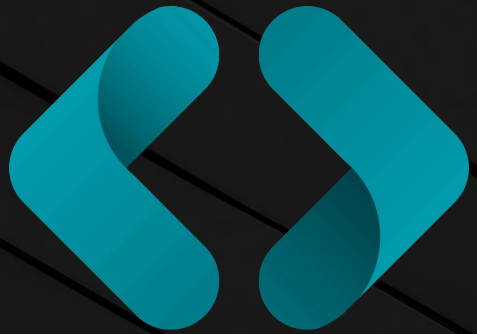
How We'll Use the Census API

The `c.asc5.get` method grabs data on each field.

`Poverty Rate` is divided by `Total Population` to evaluate `Poverty Rate`.

The U.S. Census does not explicitly calculate `Poverty Rate`.

```
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"]]
```



Time to <code>

Activity: Census Activity

In this activity, you will use the Census API to obtain state-level census data and visualize it with Gmaps.

(Instructions sent via Slack.)

Suggested Time:

20 minutes

Activity: 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.

Bonus

With the coordinates now appended to the DataFrame, you can 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

Activity: 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 in 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_2 value.

Write a sentence describing your findings. Were they what you expected? What other factors could be at play?

Time's Up! Let's Review.