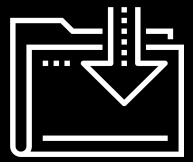
Python APIs: Day 3

Data Boot Camp

Lesson 6.3



Class Objectives

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

Use the Geoapify API 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 by using GeoViews.

Instructor Demonstration

Geoapify API

Geoapify API

Today’s class will cover the Geoapify API.

https://apidocs.geoapify.com/playground/places

Obtaining an API Key

Instructions	
	Visit https://www.geoapify.com/, and click **"LOG IN / SIGN UP"**.
	In the "Sign in" page, register a new account or sign in with Google or Facebook.
	Set up a project and assign a name.
	Copy and safely store you API key**.**
	You can can use this API free of charge to make up to 3,000 API calls.
	Review the API documentation page: https://apidocs.geoapify.com/.

Instructor Demonstration

Geoapify Geocode

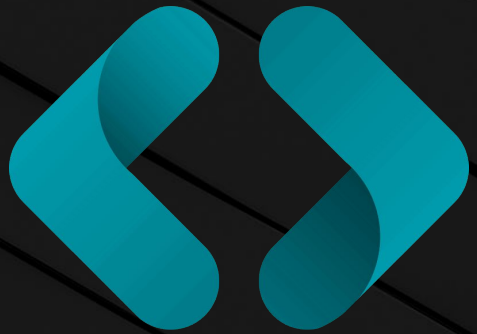
Geoapify Geocode

Geoapify API geocoding feature 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.

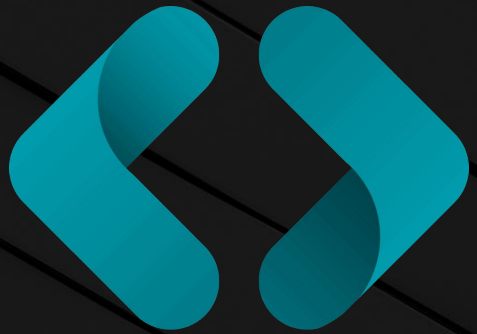

```
"distance": 0,
"formatted": "Sydney, NSW, Australia",
"lat": -33.8698439,
"lon": 151.2082848,
"place_id": "512088e244aae662405903907f0b57ef40c0f00101f901f5bc570000000000c00208",
"rank": {
    "confidence": 1,
    "confidence_city_level": 1,
    "importance": 0.8245908962989684,
    "match_type": "full_match",
    "popularity": 7.332617625087506
```



Time to <code>

Instructor Demonstration

Geoapify Places



Time to <code>

Activity: Geoapify Drills

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

Suggested Time:

15 minutes

Time's Up! Let's Review.

Instructor Demonstration

Exploring Nearest Restaurants in Madrid

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']`.

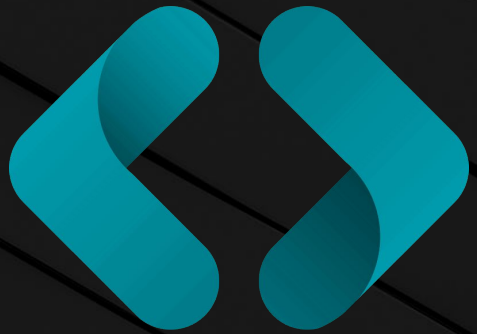
With each iteration, the keyword is overwritten.

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.

```python
# Iterate through the types_df DataFrame
for index, row in types_df.iterrows():

    # Get the ethnicity type from the current DataFrame's row
    ethnicity = types_df.loc[index, "ethnicity"]
    # Add the current ethnicity type to the parameters
    params["categories"] = f"catering.restaurant.{ethnicity}"

    # Make an API request using the params dictionary
    restaurant = requests.get(base_url, params=params)

    # Convert the API response to JSON format
    restaurant = restaurant.json()

    # Grab the first restaurant from the results and store the details in the DataFrame
    try:
        types_df.loc[index, "name"] = restaurant["features"][0]["properties"]["name"]
        types_df.loc[index, "address"] = restaurant["features"][0]["properties"]["address_line2"]
        types_df.loc[index, "distance"] = int(restaurant["features"][0]["properties"]["distance"])
    except (KeyError, IndexError):
        # If no restaurant is found, set the restaurant name as "No restaurant found".
        types_df.loc[index, "name"] = "No restaurant found"
        # Set the distance column value to np.nan to allow sorting values
        types_df.loc[index, "distance"] = np.nan

    # Log the search results
    print(f"nearest {types_df.loc[index, 'ethnicity']} restaurant: {types_df.loc[index, 'name']}")
```



Time to <code>


Break

Activity: Exploring Airports in Australia

In this activity, you will obtain information for the airports in major Australian cities.

Suggested Time:

20 minutes

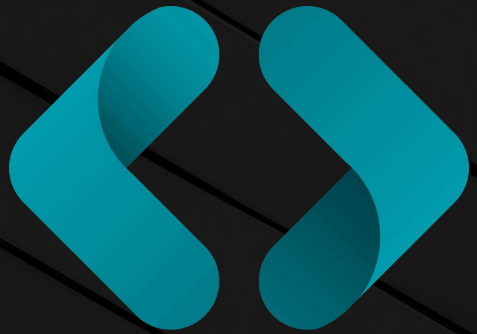
Activity: Exploring Airports in Australia

Instructions	With `airports.ipynb` as your starting point, use the Geoapify Geocoding API, the Geoapify Places API, and Python to create a script that retrieves information of some Australian airports in each of the cities found in `Cities.csv`.
	Your final notebook file should contain each of the following headers: `Lat, Lon, Airport Name, IATA Name, Airport Address, Distance, Website`
Hints	You will need to obtain the latitude `(lat)` and longitude `(lon)` of each airport prior to sending it through the Geoapify Places API to obtain the information.
	When using the Geoapify Places API, make sure to use the `"airport"` category to ensure that the data received is for an airport in the city.
	Use a `try-except` to identify airports for which there are missing data..

Time's Up! Let's Review.

Instructor Demonstration

GeoViews Maps



Time to <code>

Activity: Australian Airports Map

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

Suggested Time:

15 minutes

Time's Up! Let's Review.

Instructor Demonstration

U.S. Census Demo

U.S. 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) 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.

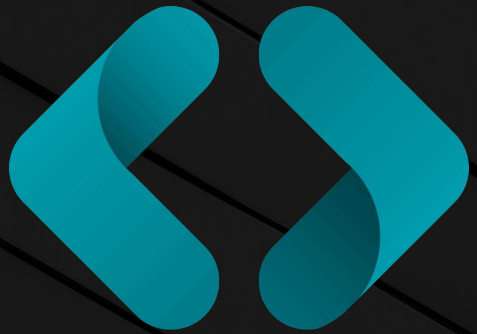
How We'll Use the U.S. Census API

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: Banking Deserts

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

Suggested Time:

20 minutes

Activity: Banking Deserts Heatmap

Instructions

Using `GeoViews`, create a poverty rate map:

- Use the "Poverty Rate" column to set the point's size. Recall using the `scale` parameter to modify the size appearance.
- Use the "Zipcode" column to set the point's colour.
- Read the HoloViews documentation and learn how you can use the `hover_cols` parameter to add additional information to the tooltip of a point. Add the "Address" and the "Bank Count" columns.

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.