Data, Metadata and APIs

Part 5: The Google Maps API and Open Data

Now that you've extracted GPS coordinates from JPEG metadata and mapped it using the Google Maps API, you might be wondering what else you can do with the Google Maps API. The short answer is... a lot.

In this notebook, you'll see how to combine your knowledge of the Google Maps API with your knowledge of data analysis with Pandas.

Find an Open Data Set that contains Location Data

Here's a data set that tracks the location of all potholes filled by the City of Chicago for the past 7 days. Chicago is known for its potholes (https://www.wbez.org/shows/curious-city/city-of-big-potholes-is-asphalt-the-best-choice-for-chicagos-streets/8bbd9e7a-b27e-4e00-a868-aa0b826b53b2), so this should be good.

We will load this *.csv* file in from a URL so that it is guaranteed to be the most up-to-date as possible:

In [2]:
```
# Note: the spike in traffic from Fremd may get us IP-banned by Chicago's Open Da
#       If this happens, your teacher will share a static copy of Potholes_Patche
#       and you'll need to run the code "potholes_DF = pd.read_csv('Potholes_Patc

import pandas as pd

potholes_DF = pd.read_csv("Potholes_Patched.csv")

# display the 3 most recent potholes that were filled
potholes_DF[-3:]
```

Out[2]:

	ADDRESS	REQUEST DATE	COMPLETION DATE	NUMBER OF POTHOLES FILLED ON BLOCK	LATITUDE	LONGITUDE	LOCA
60305	5900 S KEELER AVE	03/11/2020 03:19:15 PM	03/17/2020 06:29:21 AM	8	41.785804	-87.727891	P((-87.72789140 41.785803772
60306	4339 S WASHTENAW AVE	03/17/2020 11:06:21 AM	03/17/2020 11:08:01 AM	3	41.814439	-87.691686	P((-87.69168586 41.81443892
60307	6310 N GREENVIEW AVE	03/13/2020 03:02:56 PM	03/17/2020 01:07:39 PM	12	41.996762	-87.667839	P((-87.66783858 41.996761979

Check how many potholes were filled in the last week:

In [3]:
```
print(len(potholes_DF))
```

```
60308
```

That's a lot of potholes. Now extract the location data, clean out the "nan" values, and store it as a list of tuples:

In [4]:
```
import numpy as np

lat = list(potholes_DF["LATITUDE"])

lon = list(potholes_DF["LONGITUDE"])

tuple_list = []

for i in range(len(lat)):
    coord = (lat[i],lon[i])
    tuple_list.append(coord)

tuple_list = [x for x in tuple_list if not np.isnan(x[1])]
```

Let's compare the length of *potholes_DF* to *tuple_list* to see how many "nan" values we cleaned out:

In [5]:
```
print(len(potholes_DF),len(tuple_list))
```

```
60308 52061
```

Depending on the week, there may be a handful of "nan" values to clean out. If you were lucky, there were none.

Now let's look at a few of the tuples in the list:

In [6]:
```
tuple_list[-10:]
```

Out[6]:
```
[(41.654151783, -87.538938104),
 (41.93726585, -87.680959323),
 (41.866800963, -87.671623008),
 (41.851282923, -87.693164695),
 (41.903007756, -87.721701289),
 (41.991803565, -87.741005558),
 (41.810436175999996, -87.685549079),
 (41.785803772, -87.72789140100001),
 (41.814438925, -87.691685867),
 (41.996761979, -87.667838581)]
```

Google Maps API with Markers

Let's put a marker every place we found a pothole.

WARNING: Adding more than 500 marker points could potentially crash your kernel! To combat this, we are creating a list of 500 random entries from the original tuple_list.

In [23]:
```
import numpy as np

tuple_list_500 = []
indicies_used = []
for i in range(500):                                    # Loop 500 times
    random = np.random.randint(0,500)                   # Generate random index numb
    if random not in indicies_used:                     # Check if number has alread
        indicies_used.append(random)                    # Add new number to list of
        tuple_list_500.append(tuple_list[random])       # Add the tuple from that in
print(tuple_list_500[:10])
```

```
[(41.802136294, -87.72804566299999), (41.993531, -87.655668), (41.969670165, -8
7.78166036200001), (41.896815000000004, -87.68715), (41.953682, -87.712053), (4
1.894507198, -87.65618182), (41.953920000000004, -87.72498399999999), (41.84265
1000000004, -87.64429399999999), (41.730125753, -87.68002877200001), (41.954361
999999996, -87.685873)]
```

In [24]:
```
# Import the gmaps python module and load in your API Key:
import gmaps
gmaps.configure(api_key="AIzaSyCLla6Q7krE9xNg6SnNMoGNIzjCLddE9EU")
```

In [25]:
```
from ipywidgets.embed import embed_minimal_html # Allows us to create a separte
markers = gmaps.marker_layer(tuple_list_500)    # Create markers for each tuple/
markermap = gmaps.Map()                         # Create a GMap variable
markermap.add_layer(markers)                    # Add the layer of markers to GM

embed_minimal_html('MarkerMap1.html', views=[markermap])
print("*** Check your 'Metadata Part 5' folder to find the new HTML file name \"M
```

```
*** Check your 'Metadata Part 5' folder to find the new HTML file name "MarkerM
ap1.html". ***
```

Question 1: Look at the marker map at various zoom levels. What do you notice above the graph? Comment on anything interesting you see and try to summarize "the good" and "the bad" in this visualization.

Your Answer:

Good: Very specific Markers: Pins it to not only street, but which side of the street. It also is very easy to use, and I am able to click on the area that I want to see to zoom in.

Bad: The map only covers half of my browser tab, and everything is convered by the watertag " For Developmental purposes only".

Google Maps API to Create a Heatmap

Instead of markers, let's make a heat map:

WARNING: Adding more than 500 marker points could potentially crash your kernel! To combat this, we are again using the list of 500 random entries from the original tuple_list.

In [10]:
```
from ipywidgets.embed import embed_minimal_html # Allows us to create a separte j

heatm = gmaps.Map()
heatm.add_layer(gmaps.heatmap_layer(tuple_list_500))

embed_minimal_html('MarkerMap2.html', views=[heatm])
print("*** Check your 'Metadata Part 5' folder to find the new HTML file. ***")
```

```
*** Check your 'Metadata Part 5' folder to find the new HTML file. ***
```

Question 2: Look at the heatmap at various zoom levels. What do you notice above the graph? Comment on anything interesting you see and try to summarize "the good" and "the bad" in this visualization.

Your Answer:

Good: Very specific Markers: Pins it to not only street, but which side of the street. It also is hellpful to see which area has the most potholes, as the heat signature is the most red in those areas.

Bad: The map only covers half of my browser tab, and everything is convered by the watertag " For Developmental purposes only".

Task 1: Find your own dataset!

You are going to create a marker map **and** a heatmap from a dataset you have found. For Task 1, find a dataset with location data (GPS coordinates!). Fill in the following:

Name: Higher Education School Locations - Data Table.csv

Date: 4/20/20

Source for Data Set: data.gov

URL for Data Set: https://catalog.data.gov/dataset/higher-education-school-locations

Description of Data Set: All higher education location in the USA

File Format for Data Set: csv

Age of Data Set: Feb 27 2019

Task 2: Show some entries fom your dataset

Import your data set as a Pandas Data Frame, then show the last 10 entries:

In [14]:
```
# Your code here
facilities_DF = pd.read_csv("School Location.csv")

# display the 3 most recent potholes that were filled
potholes_DF[-10:]
```

Out[14]:

	ADDRESS	REQUEST DATE	COMPLETION DATE	NUMBER OF POTHOLES FILLED ON BLOCK	LATITUDE	LONGITUDE	LO
60298	13244 S AVENUE N	03/16/2020 02:54:45 PM	03/17/2020 11:50:08 AM	19	41.654152	-87.538938	(-87.53893 41.654151
60299	3046 N HOYNE AVE	03/10/2020 07:29:30 PM	03/17/2020 01:56:18 PM	28	41.937266	-87.680959	(-87.68095 41.93726
60300	1805 W ROOSEVELT RD	03/17/2020 09:13:38 AM	03/17/2020 09:14:54 AM	1	41.866801	-87.671623	(-87.67162 41.866800
60301	2216 S WASHTENAW AVE	03/16/2020 03:13:57 PM	03/17/2020 01:02:14 PM	7	41.851283	-87.693165	(-87.69316 41.851282
60302	3800 W DIVISION ST	03/17/2020 12:19:04 PM	03/17/2020 12:21:06 PM	6	41.903008	-87.721701	(-87.72170 41.903007
60303	6069 N FOREST GLEN AVE	03/12/2020 08:34:39 AM	03/17/2020 01:49:38 PM	18	41.991804	-87.741006	(-87.74100 41.991803
60304	2422 W 46TH ST	03/17/2020 10:44:14 AM	03/17/2020 10:46:10 AM	7	41.810436	-87.685549	(-87.68554 41.810436
60305	5900 S KEELER AVE	03/11/2020 03:19:15 PM	03/17/2020 06:29:21 AM	8	41.785804	-87.727891	(-87.72789 41.785803
60306	4339 S WASHTENAW AVE	03/17/2020 11:06:21 AM	03/17/2020 11:08:01 AM	3	41.814439	-87.691686	(-87.69168 41.814438
60307	6310 N GREENVIEW AVE	03/13/2020 03:02:56 PM	03/17/2020 01:07:39 PM	12	41.996762	-87.667839	(-87.66783 41.996761

Task 3: Create a list of tuples

Use your dataset to create a list of tuples (a list of DD coordinates) representing the locations in your dataset:

WARNING: Adding more than 500 marker points could potentially crash your kernel! To combat this, create a list of 500 random entries from the original list of tuples.

In [19]:
```
# Your code here
latitude = list(potholes_DF["LATITUDE"])

longitude = list(potholes_DF["LONGITUDE"])

coordinate = []

for i in range(len(lat)):
    coord = (latitude[i],longitude[i])
    coordinate.append(coord)

coordinate = [x for x in tuple_list if not np.isnan(x[1])]


short_list = []

for i in range(500):
        short_list.append(coordinate[i])

print(short_list)
```

```
79), (41.954414, -87.682521), (41.955506, -87.677803), (41.735995, -87.668342
00000001), (41.842651000000004, -87.64429399999999), (41.750888, -87.60423),
(41.949498, -87.707739), (41.837042, -87.64538399999999), (41.954011, -87.716
87299999999), (41.954095, -87.707237), (41.776668, -87.601202), (41.954404, -
87.675758), (41.953564, -87.711965), (41.954483, -87.678191), (41.784624, -8
7.59259300000001), (41.896534, -87.687141), (41.954097999999995, -87.709902),
(41.880037, -87.742482), (41.931139, -87.71681600000001), (41.788051, -87.698
072), (41.900738, -87.687345), (41.716097, -87.556901), (41.749632, -87.60170
3), (41.939155, -87.69324300000001), (41.954346, -87.677157), (41.830201, -8
7.642108), (41.953694, -87.711137), (41.877215, -87.742965), (41.914485, -87.
687538), (41.896405, -87.686696), (41.780459, -87.591689), (41.79010506, -87.
696179552), (41.920745944000004, -87.71814981899999), (41.953571000000004, -8
7.71145899999999), (41.90503, -87.687462), (41.915638, -87.68770699999999),
(41.963089000000004, -87.721889), (41.735995, -87.66834200000001), (41.748841
999999996, -87.651648), (41.954432000000004, -87.676535), (41.920952, -87.728
226), (41.961386, -87.717721), (41.968201, -87.720945), (41.962752, -87.7196
6), (41.790575, -87.69511999999999), (41.830662, -87.64264), (41.836129, -87.
645364), (41.968146999999995, -87.724183), (41.790717623, -87.695213105), (4
1.736032, -87.658407), (41.954209000000006, -87.70985), (41.793688, -87.65384
9), (41.91225, -87.687659), (41.96308, -87.722628), (41.954370000000004, -87.
```

Task 4: Create a marker map from your data

Use the Google Maps API to create a marker map using your list of tuples from above.

In [21]:
```
# Your code here
import gmaps
gmaps.configure(api_key="AIzaSyCLla6Q7krE9xNg6SnNMoGNIzjCLddE9EU")
```

In [26]:
```
from ipywidgets.embed import embed_minimal_html # Allows us to create a separte j

marker = gmaps.marker_layer(short_list)    # Create markers for each tuple/coord
markermaps = gmaps.Map()                        # Create a GMap variable
markermaps.add_layer(marker)                   # Add the layer of markers to GM

embed_minimal_html('MarkerMap3.html', views=[markermaps])
print("*** Check your 'Metadata Part 5' folder to find the new HTML file name \"M
```

```
*** Check your 'Metadata Part 5' folder to find the new HTML file name "MarkerM
ap3.html". ***
```

Task 5: Create a heatmap from your data

Use the Google Maps API to create a **heatmap** using your list of tuples from above.

Note: The Google Maps API can struggle with heatmaps that have more than 1000 datapoints. If your map is not working, try reducing your list to fewer tuples (try creating a list with just the most recent 100 entries in the dataset). Once this works, you can always add in a few more tuples!

In [27]:
```
# Your code here
from ipywidgets.embed import embed_minimal_html # Allows us to create a separte j

heatma = gmaps.Map()
heatma.add_layer(gmaps.heatmap_layer(short_list))

embed_minimal_html('MarkerMap4.html', views=[heatma])
print("*** Check your 'Metadata Part 5' folder to find the new HTML file. ***")
```

```
*** Check your 'Metadata Part 5' folder to find the new HTML file. ***
```

Task 6: Comment on what you see

Look at your marker map and your heatmap at various zoom levels. Comment on anything interesting or notable that you see.

Your Answer: Far away, heatmaps are more helpful because you can see a physical respresentation of where the most schools are/where the most students are. Close up, gmaps is more helpful at pinpointing specific things.

Task 7: Brainstorm further study

If you had more time and resources, what else would you like to explore using the GPS data in this dataset?

Your Answer: I would like to see where are the schools are, and compare thre schools with the student count. I am wondering if there are more schools in an area, do the students in the school decreae commpared to other shcools. Also, of the population of a specific aresa was more, does that mean they havr more shcools.

In [ ]: