Geospatial Analysis Home Page (https://www.kaggle.com/learn/geospatial-analysis)

Introduction

In this tutorial, you'll learn how to create interactive maps with the **folium** package. Along the way, you'll apply your new skills to visualize Boston crime data.

Code

In [2]:

```
import folium
from folium import Choropleth, Circle, Marker
from folium.plugins import HeatMap, MarkerCluster
```

Your first interactive map

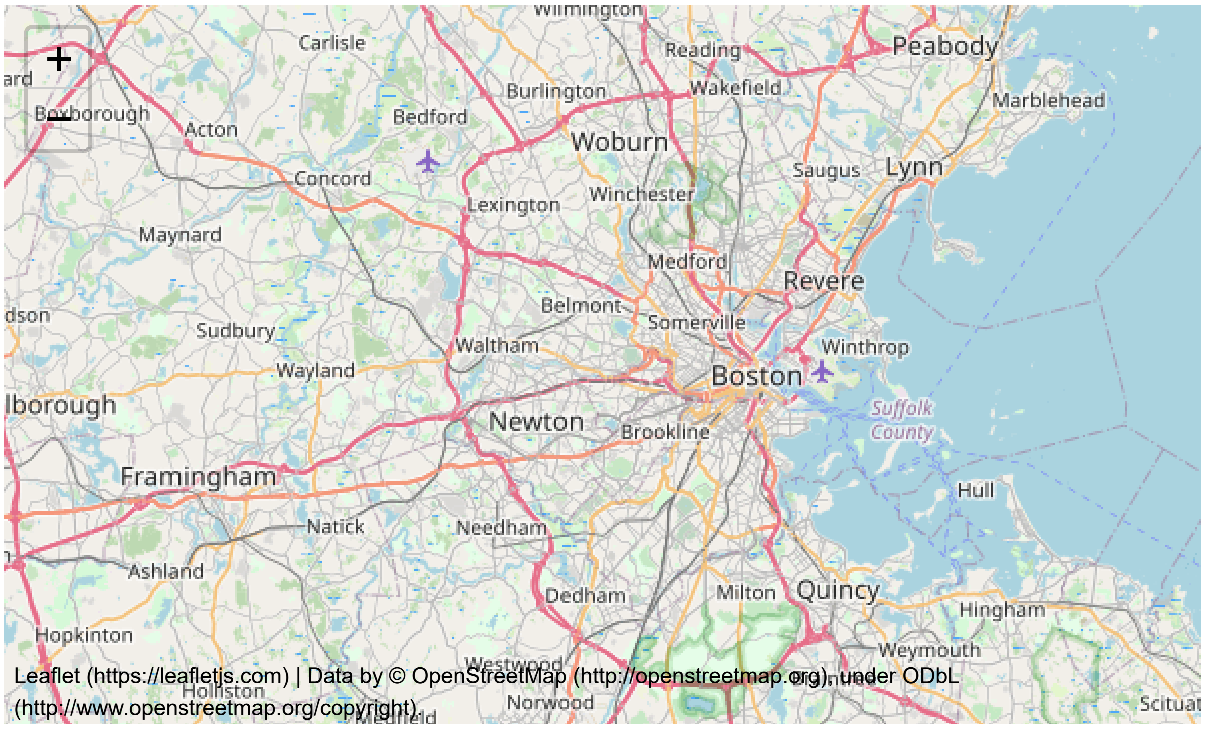
We begin by creating a relatively simple map with `folium.Map()`.

In [3]:

```python
# Create a map
m_1 = folium.Map(location=[42.32,-71.0589], tiles='openstreetmap', zoom_start=10
)

# Display the map
m_1
```

Out[3]:



Leaflet (https://leafletjs.com) | Data by © OpenStreetMap (http://openstreetmap.org), under ODbL (http://www.openstreetmap.org/copyright).

Several arguments customize the appearance of the map:

- `location` sets the initial center of the map. We use the latitude (42.32° N) and longitude (-71.0589° E) of the city of Boston.
- `tiles` changes the styling of the map; in this case, we choose the OpenStreetMap (https://www.openstreetmap.org/#map=10/42.32/-71.0589) style. If you're curious, you can find the other options listed here (https://github.com/python-visualization/folium/tree/master/folium/templates/tiles).
- `zoom_start` sets the initial level of zoom of the map, where higher values zoom in closer to the map.

Take the time now to explore by zooming in and out, or by dragging the map in different directions.

The data

Now, we'll add some crime data to the map!

We won't focus on the data loading step. Instead, you can imagine you are at a point where you already have the data in a pandas DataFrame `crimes`. The first five rows of the data are shown below.

Code

Out[4]:

	INCIDENT_NUMBER	OFFENSE_CODE	OFFENSE_CODE_GROUP	OFFENSE_DESCRIP
0	I182070945	619	Larceny	LARCENY ALL OTHI
6	I182070933	724	Auto Theft	AUTO THEFT
8	I182070931	301	Robbery	ROBBERY - STREET
19	I182070915	614	Larceny From Motor Vehicle	LARCENY THEFT FF MV - NON-ACCESS
24	I182070908	522	Residential Burglary	BURGLARY - RESIDENTIAL - NO FORCE

Plotting points

To reduce the amount of data we need to fit on the map, we'll (temporarily) confine our attention to daytime robberies.

In [5]:

```
daytime_robberies = crimes[((crimes.OFFENSE_CODE_GROUP == 'Robbery') & \
                            (crimes.HOUR.isin(range(9,18))))]
```

folium.Marker

We add markers to the map with `folium.Marker()`. Each marker below corresponds to a different robbery.

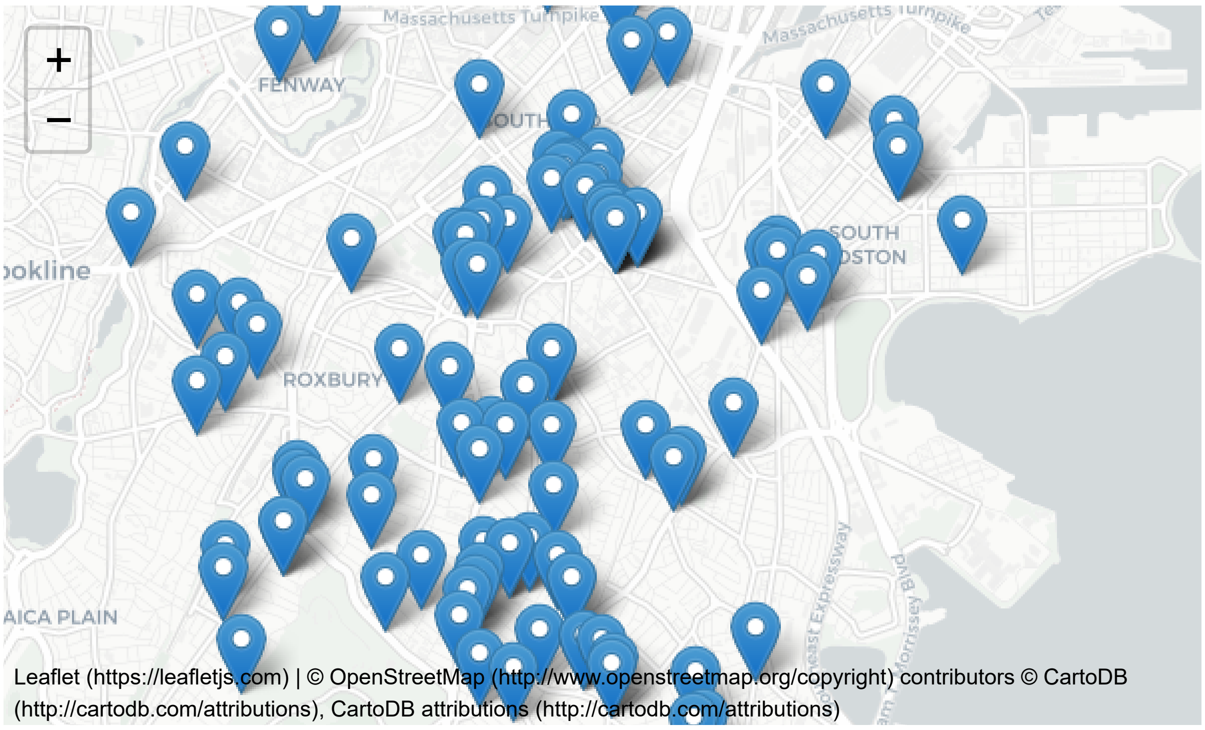
In [6]:

```python
# Create a map
m_2 = folium.Map(location=[42.32,-71.0589], tiles='cartodbpositron', zoom_start=
13)

# Add points to the map
for idx, row in daytime_robberies.iterrows():
    Marker([row['Lat'], row['Long']]).add_to(m_2)

# Display the map
m_2
```

Out[6]:


+
−
Massachusetts Turnpike
FENWAY
SOUTH
SOUTH BOSTON
ookline
ROXBURY
AICA PLAIN
Leaflet (https://leafletjs.com) | © OpenStreetMap (http://www.openstreetmap.org/copyright) contributors © CartoDB (http://cartodb.com/attributions), CartoDB attributions (http://cartodb.com/attributions)

folium.plugins.MarkerCluster

If we have a lot of markers to add, `folium.plugins.MarkerCluster()` can help to declutter the map. Each marker is added to a `MarkerCluster` object.

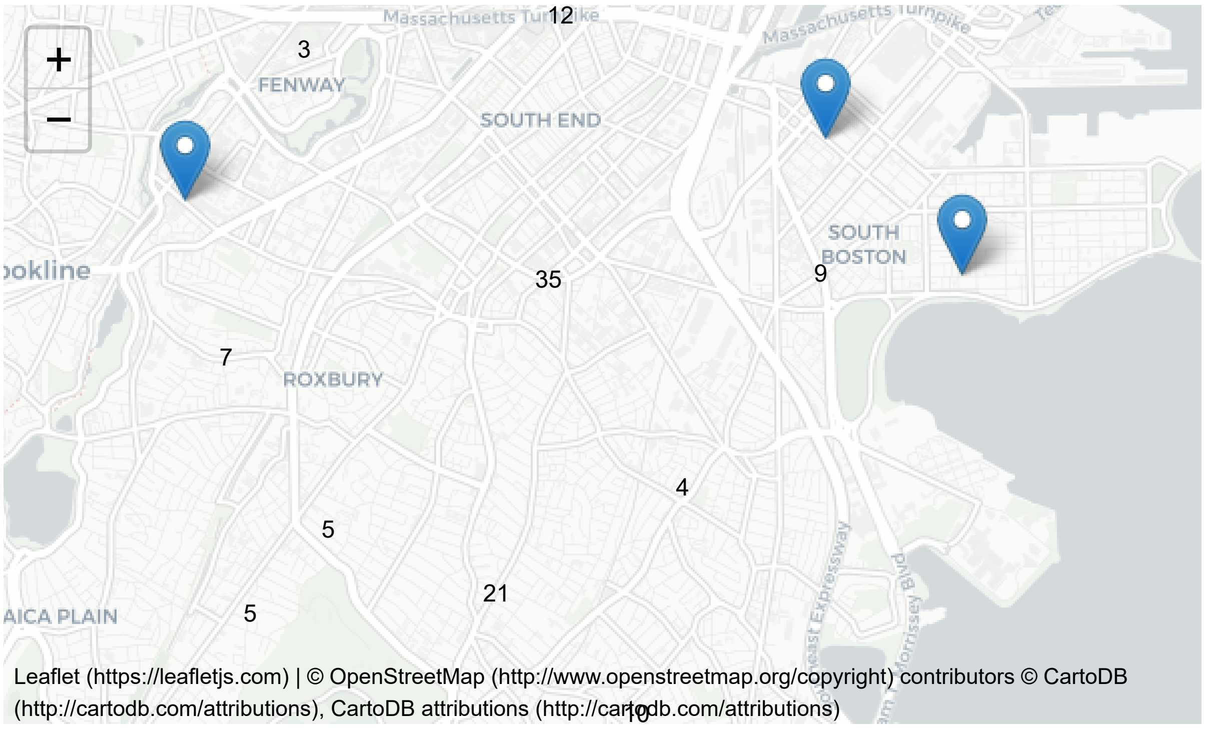
In [7]:

```
# Create the map
m_3 = folium.Map(location=[42.32,-71.0589], tiles='cartodbpositron', zoom_start=
13)

# Add points to the map
mc = MarkerCluster()
for idx, row in daytime_robberies.iterrows():
    if not math.isnan(row['Long']) and not math.isnan(row['Lat']):
        mc.add_child(Marker([row['Lat'], row['Long']]))
m_3.add_child(mc)

# Display the map
m_3
```

Out[7]:



Leaflet (https://leafletjs.com) | © OpenStreetMap (http://www.openstreetmap.org/copyright) contributors © CartoDB (http://cartodb.com/attributions), CartoDB attributions (http://cartodb.com/attributions)

Bubble maps

A **bubble map** uses circles instead of markers. By varying the size and color of each circle, we can also show the relationship between location and two other variables.

We create a bubble map by using `folium.Circle()` to iteratively add circles. In the code cell below, robberies that occurred in hours 9-12 are plotted in green, whereas robberies from hours 13-17 are plotted in red.

In [8]:

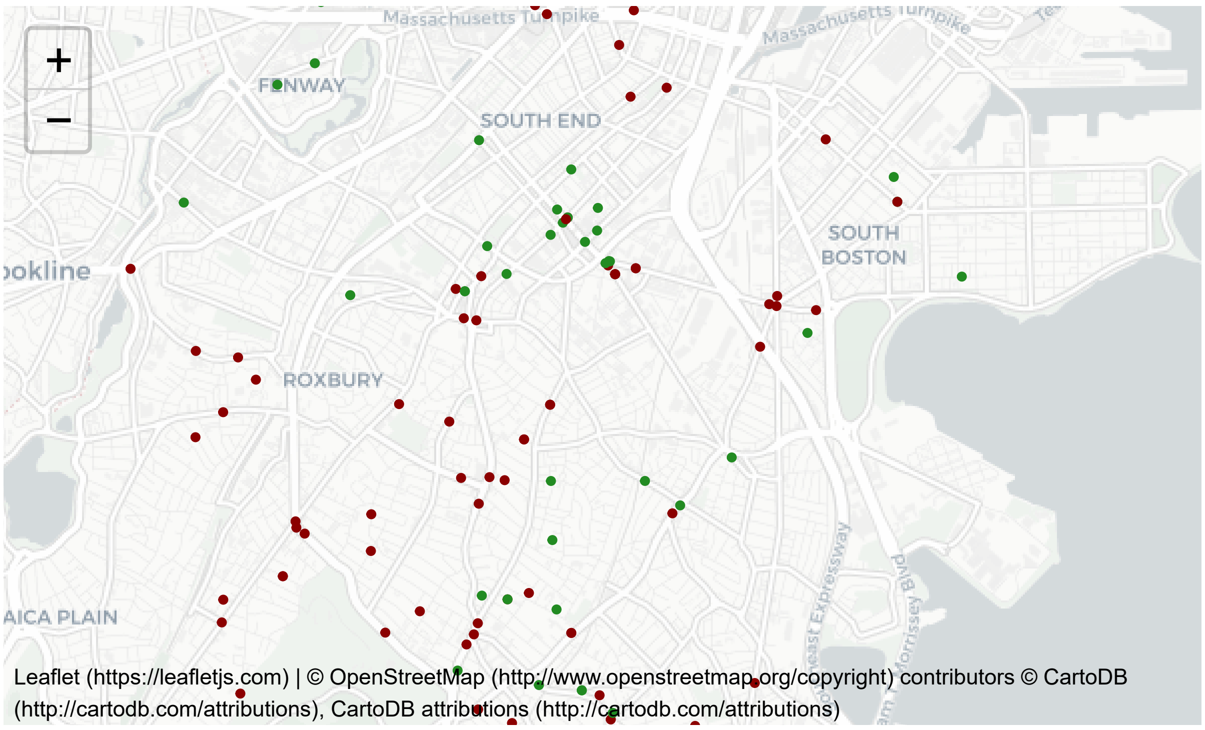
```
# Create a base map
m_4 = folium.Map(location=[42.32,-71.0589], tiles='cartodbpositron', zoom_start=
13)

def color_producer(val):
    if val <= 12:
        return 'forestgreen'
    else:
        return 'darkred'

# Add a bubble map to the base map
for i in range(0,len(daytime_robberies)):
    Circle(
        location=[daytime_robberies.iloc[i]['Lat'], daytime_robberies.iloc[i]['L
ong']],
        radius=20,
        color=color_producer(daytime_robberies.iloc[i]['HOUR'])).add_to(m_4)

# Display the map
m_4
```

Out[8]:


Massachusetts Turnpike
Massachusetts Turnpike
FENWAY
SOUTH END
SOUTH BOSTON
okline
ROXBURY
AICA PLAIN
+
−
Leaflet (https://leafletjs.com) | © OpenStreetMap (http://www.openstreetmap.org/copyright) contributors © CartoDB (http://cartodb.com/attributions), CartoDB attributions (http://cartodb.com/attributions)

Note that `folium.Circle()` takes several arguments:

- `location` is a list containing the center of the circle, in latitude and longitude.
- `radius` sets the radius of the circle.
  - Note that in a traditional bubble map, the radius of each circle is allowed to vary. We can implement this by defining a function similar to the `color_producer()` function that is used to vary the color of each circle.
- `color` sets the color of each circle.
  - The `color_producer()` function is used to visualize the effect of the hour on robbery location.

Heatmaps

To create a heatmap, we use `folium.plugins.HeatMap()` (https://python-visualization.github.io/folium/plugins.html#folium.plugins.HeatMap). This shows the density of crime in different areas of the city, where red areas have relatively more criminal incidents.

As we'd expect for a big city, most of the crime happens near the center.

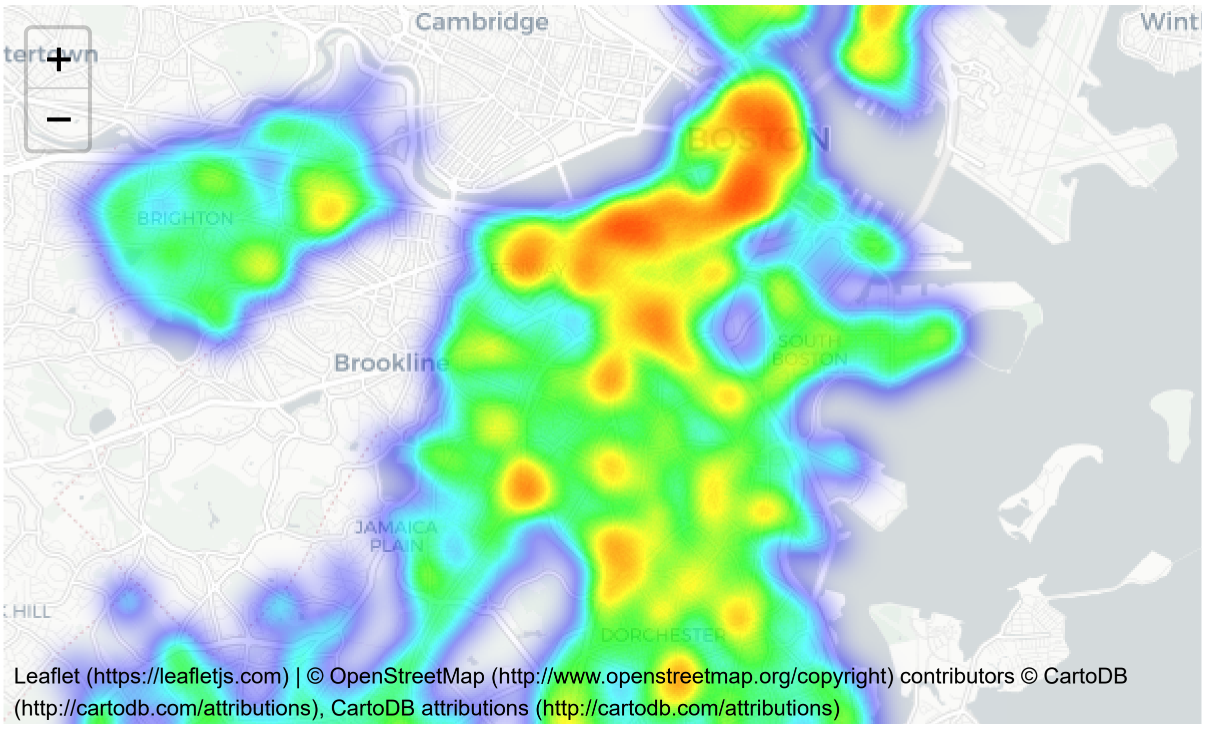
In [9]:

```python
# Create a base map
m_5 = folium.Map(location=[42.32,-71.0589], tiles='cartodbpositron', zoom_start=
12)

# Add a heatmap to the base map
HeatMap(data=crimes[['Lat', 'Long']], radius=10).add_to(m_5)

# Display the map
m_5
```

Out[9]:



Leaflet (https://leafletjs.com) | © OpenStreetMap (http://www.openstreetmap.org/copyright) contributors © CartoDB (http://cartodb.com/attributions), CartoDB attributions (http://cartodb.com/attributions)

As you can see in the code cell above, `folium.plugins.HeatMap()` takes a couple of arguments:

- `data` is a DataFrame containing the locations that we'd like to plot.
- `radius` controls the smoothness of the heatmap. Higher values make the heatmap look smoother (i.e., with fewer gaps).

Choropleth maps

To understand how crime varies by police district, we'll create a choropleth map.

As a first step, we create a GeoDataFrame where each district is assigned a different row, and the "geometry" column contains the geographical boundaries.

In [10]:

```
# GeoDataFrame with geographical boundaries of Boston police districts
districts_full = gpd.read_file('../input/geospatial-learn-course-data/Police_Districts/Police_Districts/Police_Districts.shp')
districts = districts_full[["DISTRICT", "geometry"]].set_index("DISTRICT")
districts.head()
```

Out[10]:

	geometry
DISTRICT	
A15	(POLYGON ((-71.07415718153364 42.3905076862483...
A7	(POLYGON ((-70.99644430907341 42.3955679826137...
A1	POLYGON ((-71.05199523849357 42.36883599550553...
C6	POLYGON ((-71.04405776717314 42.35403006334784...
D4	POLYGON ((-71.07416484856725 42.3572379188053,...

We also create a Pandas Series called `plot_dict` that shows the number of crimes in each district.

In [11]:

```
# Number of crimes in each police district
plot_dict = crimes.DISTRICT.value_counts()
plot_dict.head()
```

Out[11]:

```
D4     2885
B2     2231
A1     2130
C11    1899
B3     1421
Name: DISTRICT, dtype: int64
```

It's very important that `plot_dict` has the same index as `districts` - this is how the code knows how to match the geographical boundaries with appropriate colors.

Using the `folium.Choropleth()` class, we can create a choropleth map. If the map below does not render for you, try viewing the page in a different web browser (https://github.com/python-visualization/folium/issues/812).

In [12]:

```
# Create a base map
m_6 = folium.Map(location=[42.32,-71.0589], tiles='cartodbpositron', zoom_start=
12)

# Add a choropleth map to the base map
Choropleth(geo_data=districts.__geo_interface__,
           data=plot_dict,
           key_on="feature.id",
           fill_color='YlGnBu',
           legend_name='Major criminal incidents (Jan-Aug 2018)'
          ).add_to(m_6)

# Display the map
m_6
```

Out[12]:

Note that `folium.Choropleth()` takes several arguments:

- `geo_data` is a GeoJSON FeatureCollection containing the boundaries of each geographical area.
  - In the code above, we convert the `districts` GeoDataFrame to a GeoJSON FeatureCollection (https://en.wikipedia.org/wiki/GeoJSON) with the `__geo_interface__` attribute.
- `data` is a Pandas Series containing the values that will be used to color-code each geographical area.
- `key_on` will always be set to `feature.id`.
  - This refers to the fact that the GeoDataFrame used for `geo_data` and the Pandas Series provided in `data` have the same index. To understand the details, we'd have to look more closely at the structure of a GeoJSON Feature Collection (where the value corresponding to the "features" key is a list, wherein each entry is a dictionary containing an "id" key).
- `fill_color` sets the color scale.
- `legend_name` labels the legend in the top right corner of the map.

Your turn

Design your own maps (https://www.kaggle.com/kernels/fork/5832145) to determine which areas of Japan need extra earthquake reinforcement.

Geospatial Analysis Home Page (https://www.kaggle.com/learn/geospatial-analysis)

Have questions or comments? Visit the Learn Discussion forum (https://www.kaggle.com/learn-forum/161464) to chat with other Learners.