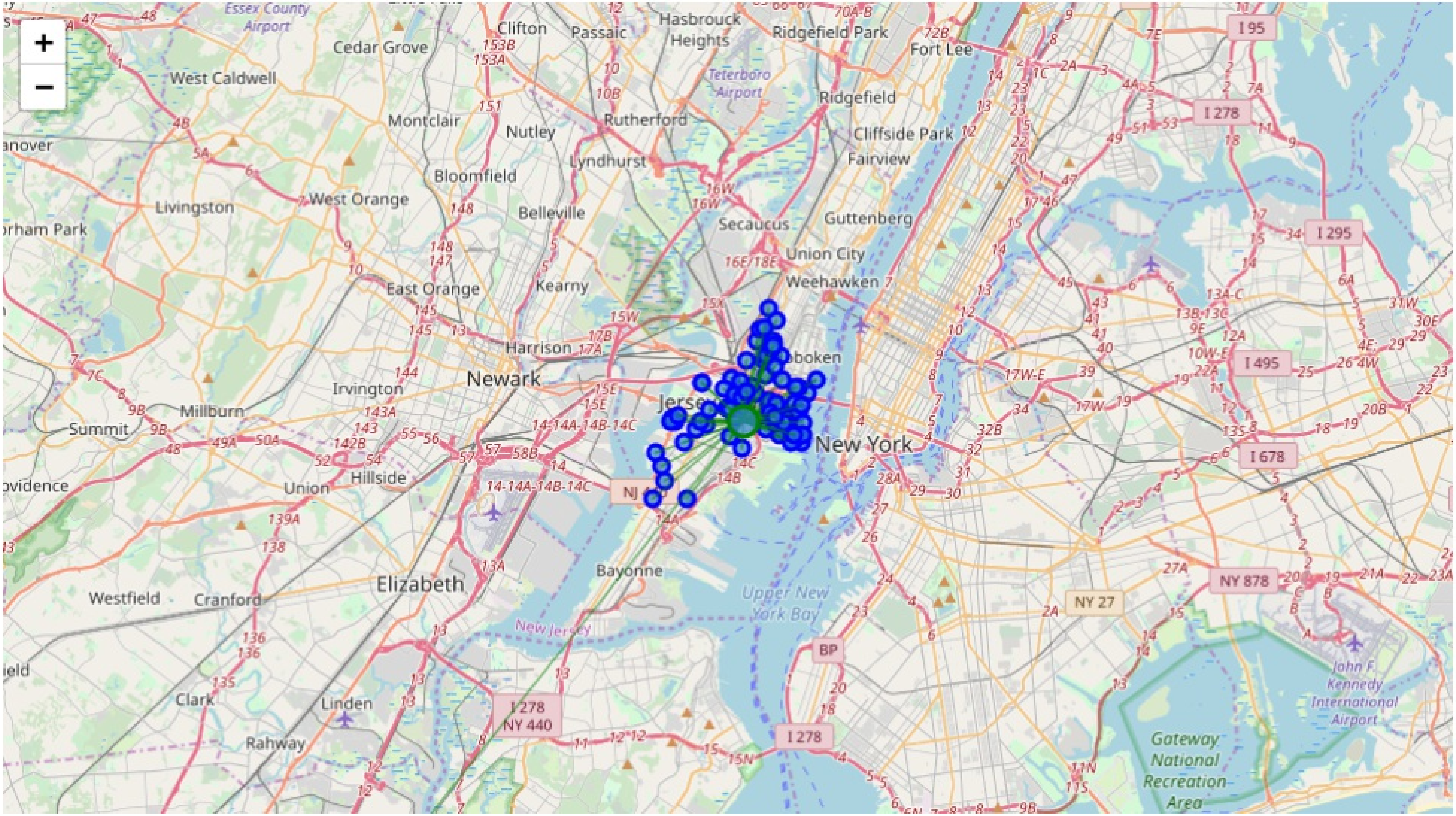
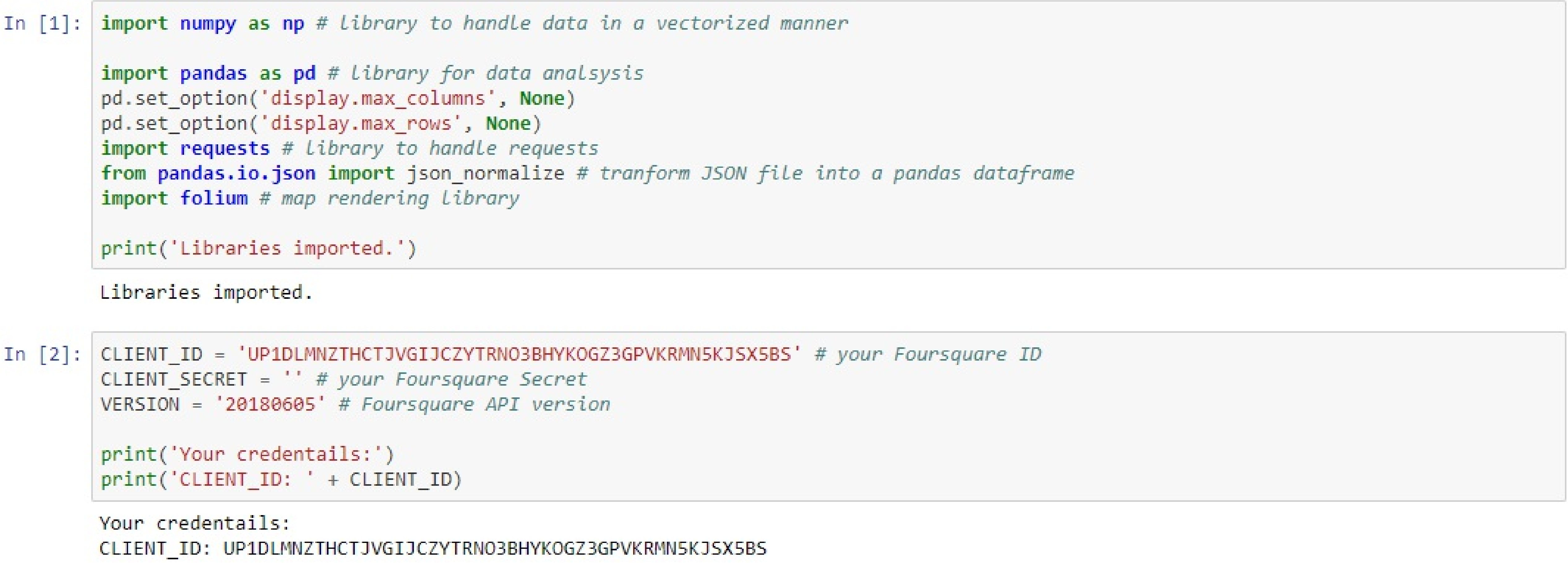
Jersey City:

MDMC: 0.029950



Rock's Bar

Muchas personas tienen preferencia por salir un fin de semana a disfrutar una buena cerveza acompa#ado de buen rock de fondo. Quiza escuchar bandas en vivo y compartir con personas con la misma afinidad musical



In [1]:

```
import numpy as np # library to handle data in a vectorized manner

import pandas as pd # library for data analsysis
pd.set_option('display.max_columns', None)
pd.set_option('display.max_rows', None)
import requests # library to handle requests
from pandas.io.json import json_normalize # tranform JSON file into a pandas dataframe
import folium # map rendering library

print('Libraries imported.')
```

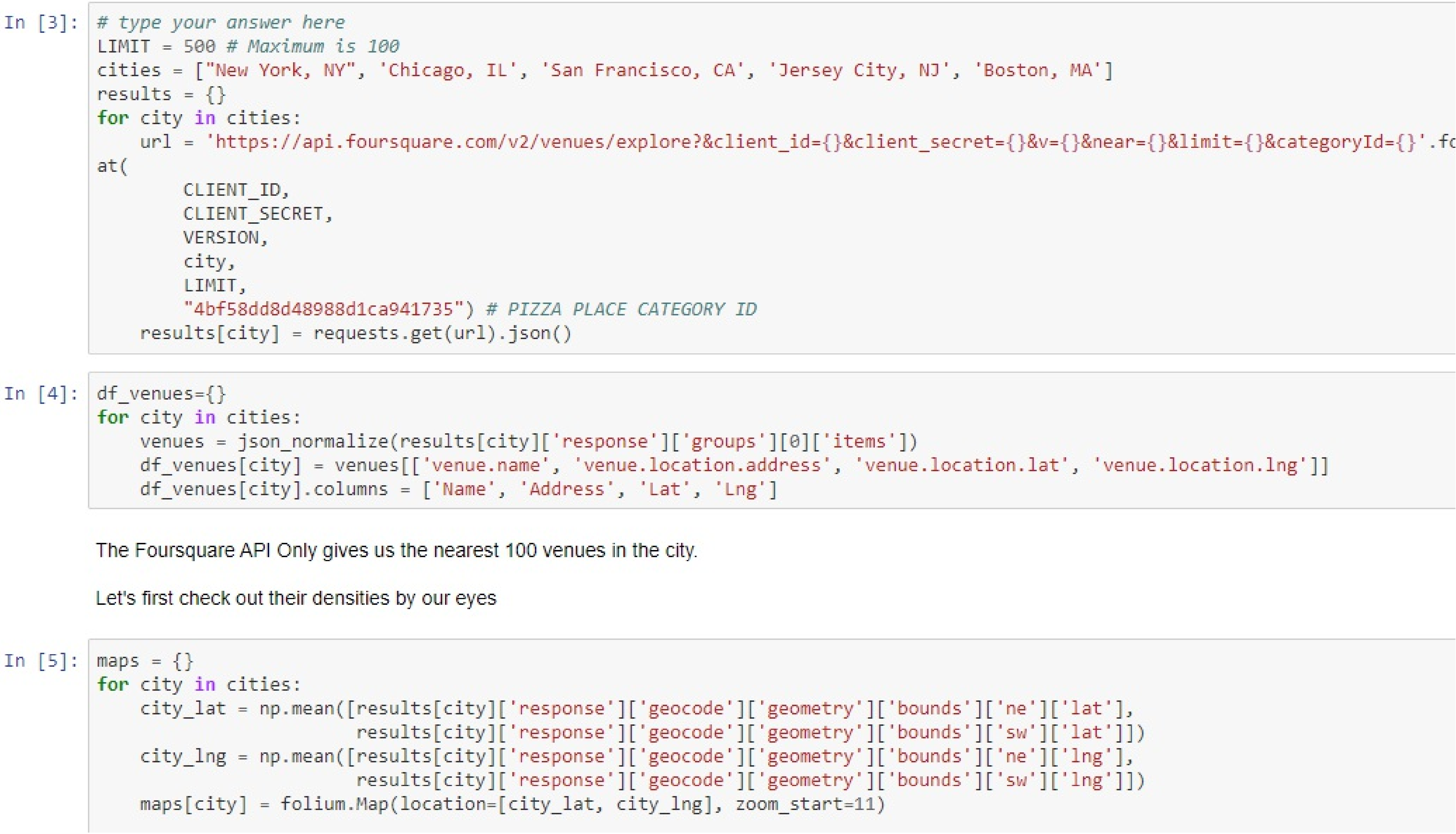
```
Libraries imported.
```

In [2]:

```
CLIENT_ID = 'UP1DLMNZTHCTJVGIJCZYTRNO3BHYKOGZ3GPVKRMN5KJSX5BS' # your Foursquare ID
CLIENT_SECRET = '' # your Foursquare Secret
VERSION = '20180605' # Foursquare API version

print('Your credentails:')
print('CLIENT_ID: ' + CLIENT_ID)
```

```
Your credentails:
CLIENT_ID: UP1DLMNZTHCTJVGIJCZYTRNO3BHYKOGZ3GPVKRMN5KJSX5BS
```



In [3]:
```
# type your answer here
LIMIT = 500 # Maximum is 100
cities = ["New York, NY", 'Chicago, IL', 'San Francisco, CA', 'Jersey City, NJ', 'Boston, MA']
results = {}
for city in cities:
    url = 'https://api.foursquare.com/v2/venues/explore?&client_id={}&client_secret={}&v={}&near={}&limit={}&categoryId={}'.fo
at(
        CLIENT_ID,
        CLIENT_SECRET,
        VERSION,
        city,
        LIMIT,
        "4bf58dd8d48988d1ca941735") # PIZZA PLACE CATEGORY ID
    results[city] = requests.get(url).json()
```

In [4]:
```
df_venues={}
for city in cities:
    venues = json_normalize(results[city]['response']['groups'][0]['items'])
    df_venues[city] = venues[['venue.name', 'venue.location.address', 'venue.location.lat', 'venue.location.lng']]
    df_venues[city].columns = ['Name', 'Address', 'Lat', 'Lng']
```

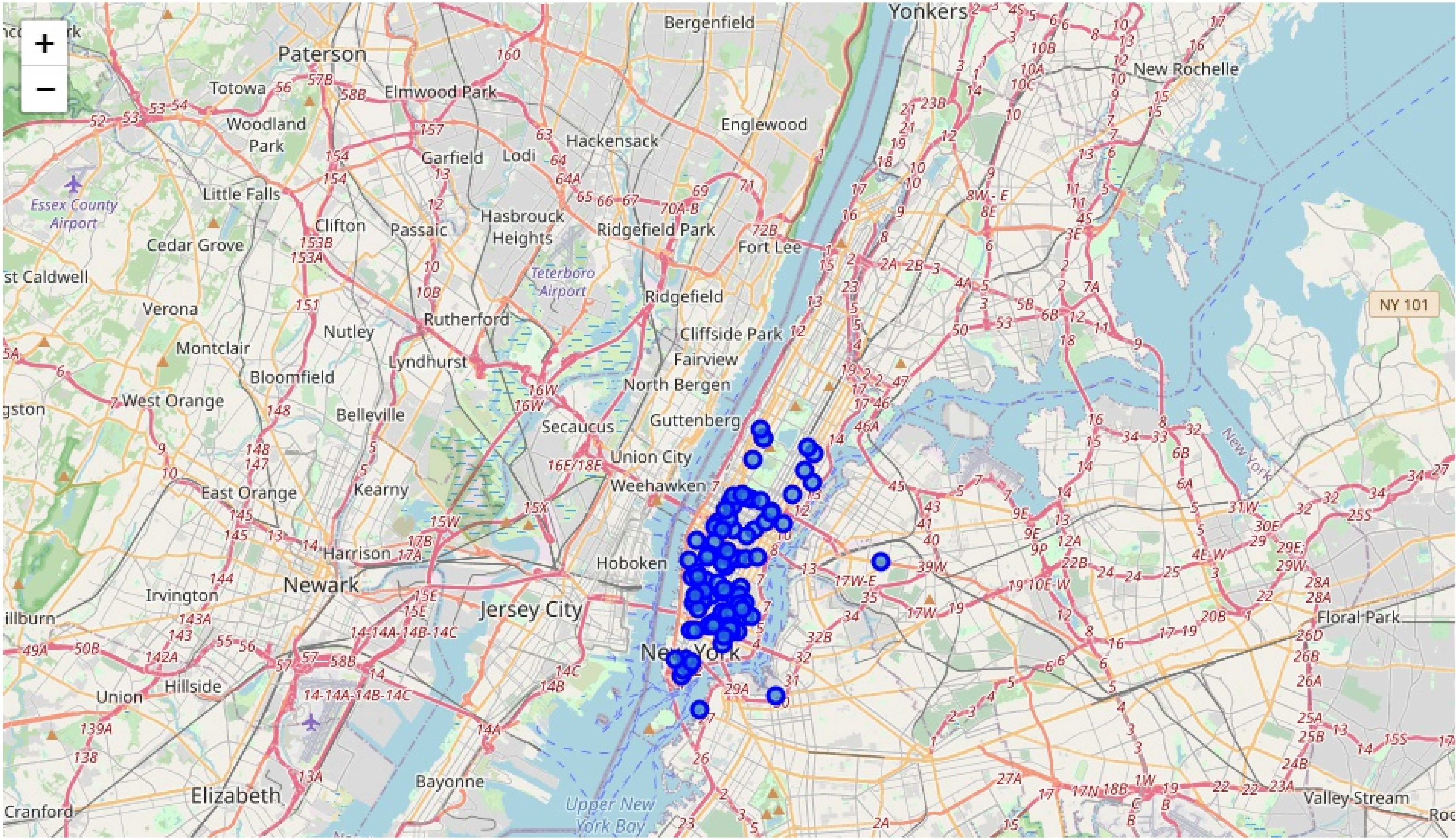
The Foursquare API Only gives us the nearest 100 venues in the city.

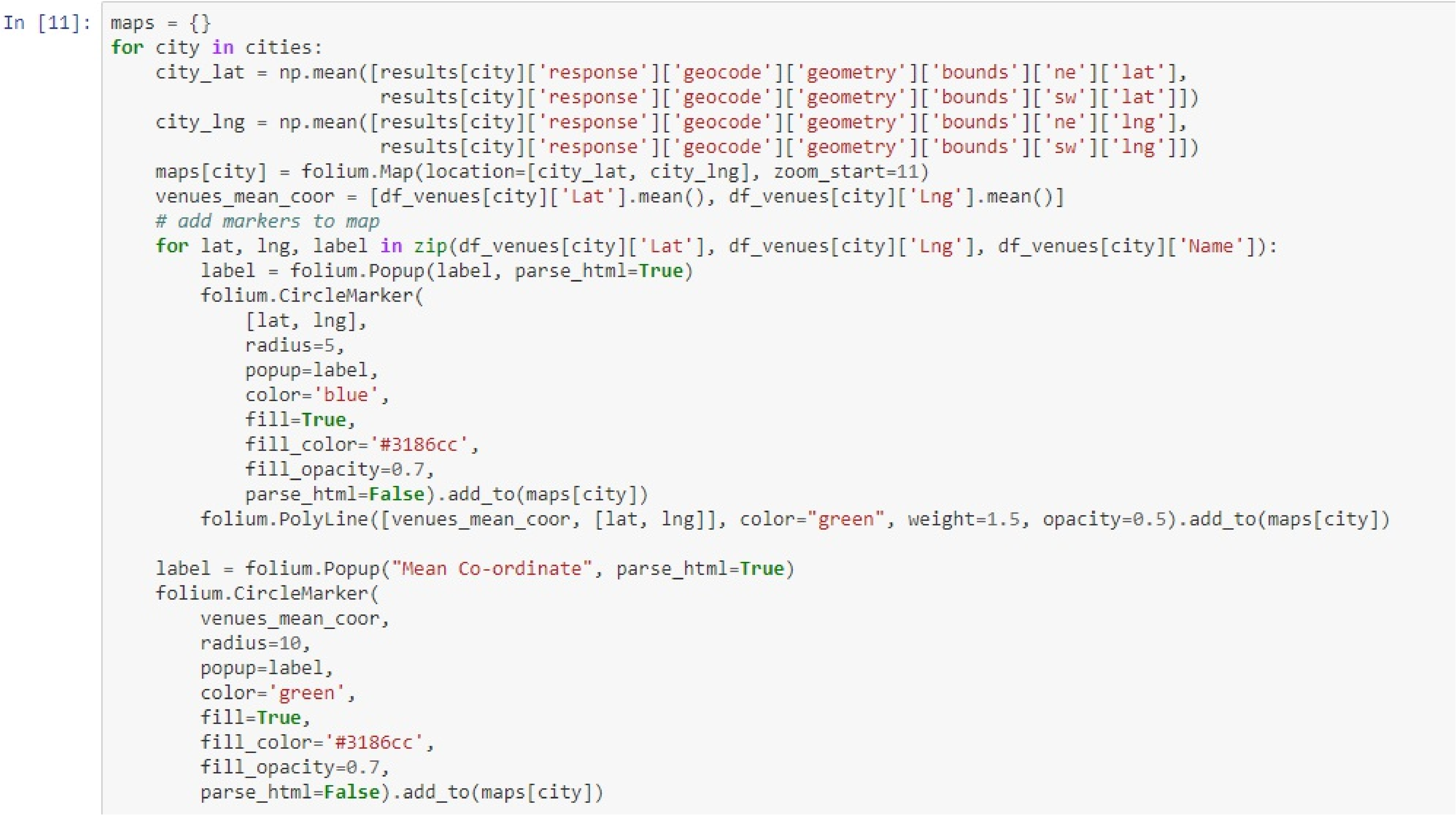
Let's first check out their densities by our eyes

In [5]:
```
maps = {}
for city in cities:
    city_lat = np.mean([results[city]['response']['geocode']['geometry']['bounds']['ne']['lat'],
                        results[city]['response']['geocode']['geometry']['bounds']['sw']['lat']])
    city_lng = np.mean([results[city]['response']['geocode']['geometry']['bounds']['ne']['lng'],
                        results[city]['response']['geocode']['geometry']['bounds']['sw']['lng']])
    maps[city] = folium.Map(location=[city_lat, city_lng], zoom_start=11)
```

```
In [6]: maps[cities[0]]
```

```
Out[6]:
```

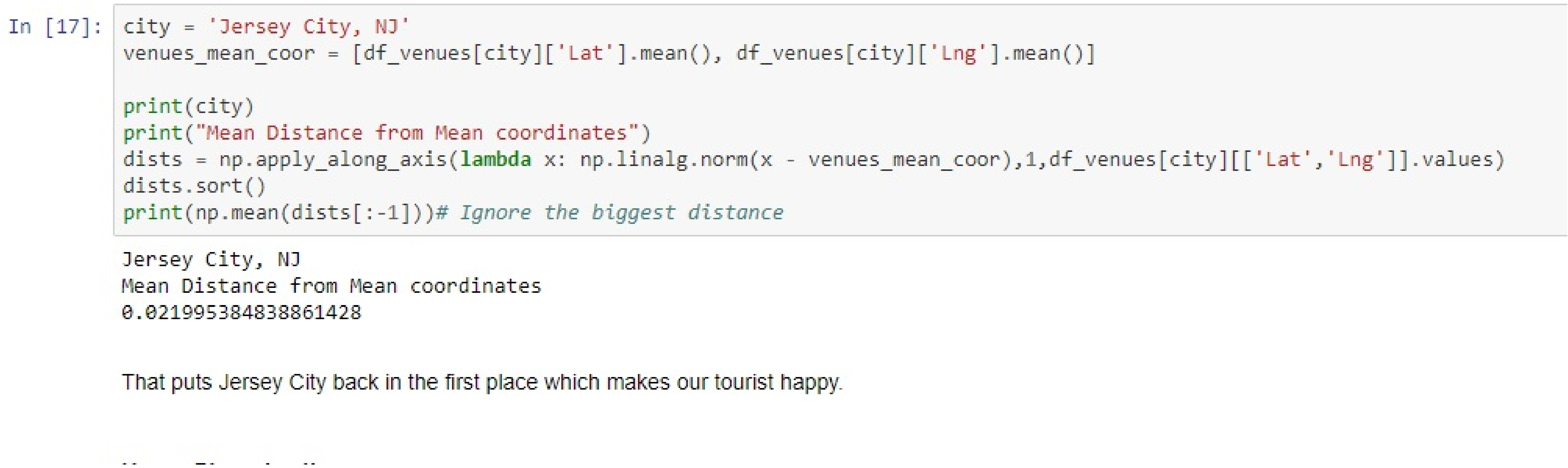




In [11]:

```
maps = {}
for city in cities:
    city_lat = np.mean([results[city]['response']['geocode']['geometry']['bounds']['ne']['lat'],
                        results[city]['response']['geocode']['geometry']['bounds']['sw']['lat']])
    city_lng = np.mean([results[city]['response']['geocode']['geometry']['bounds']['ne']['lng'],
                        results[city]['response']['geocode']['geometry']['bounds']['sw']['lng']])
    maps[city] = folium.Map(location=[city_lat, city_lng], zoom_start=11)
    venues_mean_coor = [df_venues[city]['Lat'].mean(), df_venues[city]['Lng'].mean()]
    # add markers to map
    for lat, lng, label in zip(df_venues[city]['Lat'], df_venues[city]['Lng'], df_venues[city]['Name']):
        label = folium.Popup(label, parse_html=True)
        folium.CircleMarker(
            [lat, lng],
            radius=5,
            popup=label,
            color='blue',
            fill=True,
            fill_color='#3186cc',
            fill_opacity=0.7,
            parse_html=False).add_to(maps[city])
        folium.PolyLine([venues_mean_coor, [lat, lng]], color="green", weight=1.5, opacity=0.5).add_to(maps[city])

    label = folium.Popup("Mean Co-ordinate", parse_html=True)
    folium.CircleMarker(
        venues_mean_coor,
        radius=10,
        popup=label,
        color='green',
        fill=True,
        fill_color='#3186cc',
        fill_opacity=0.7,
        parse_html=False).add_to(maps[city])
```



```
city = 'Jersey City, NJ'
venues_mean_coor = [df_venues[city]['Lat'].mean(), df_venues[city]['Lng'].mean()]

print(city)
print("Mean Distance from Mean coordinates")
dists = np.apply_along_axis(lambda x: np.linalg.norm(x - venues_mean_coor),1,df_venues[city][['Lat','Lng']].values)
dists.sort()
print(np.mean(dists[:-1]))# Ignore the biggest distance
```

```
Jersey City, NJ
Mean Distance from Mean coordinates
0.021995384838861428
```

That puts Jersey City back in the first place which makes our tourist happy.