Yelp API - Lab

The previously deployed lab on working around building a GIS with Yelp API and Folium can be found here (not relevant for new students).

Introduction

Now that we've seen how the Yelp API works, and some basic Folium visualizations its time to put those skills to work in order to create a working map! Taking things a step further, you'll also independently explore how to perform pagination in order to retrieve a full results set from the Yelp API!

Objectives

You will be able to:

- Create HTTP requests to get data from Yelp API
- Parse HTTP responses and perform data analysis on the data returned
- Perform pagination to retrieve troves of data!
- Create a simple geographical system on to view information about selected businesses, at a given location.

Problem Introduction

You've now worked with some API calls, but we have yet to see how to retrieve a more complete dataset in a programmatic manner. Returning to the Yelp API, the documentation also provides us details regarding the API limits. These often include details about the number of requests a user is allowed to make within a specified time limit and the maximum number of results to be returned. In this case, we are told that any request has a maximum of 50 results per request and defaults to 20. Furthermore, any search will be limited to a total of 1000 results. To retrieve all 1000 of these results, we would have to page through the results piece by piece, retriving 50 at a time. Processes such as these are often refered to as pagination.

In this lab, you will define a search and then paginate over the results to retrieve all of the results. You'll then parse these responses as a DataFrame (for further exploration) and create a map using Folium to visualize the results geographically.

Part I - Make the Initial Request

Start by making an initial request to the Yelp API. Your search must include at least 2 parameters: **term** and **location**. For example, you might search for pizza restaurants in NYC. The term and location is up to you, but make the request below.

Warning: the solution code will not rerun unless you fill in your client_id and api_key.

In [1]:
```
#Your code here
import requests
import pandas as pd

client_id =
api_key =

term = 'pizza'
location = 'New York NY'

url = 'https://api.yelp.com/v3/businesses/search'

headers = {
        'Authorization': 'Bearer {}'.format(api_key),
    }

url_params = {
                'term': term.replace(' ', '+'),
                'location': location.replace(' ', '+'),
            }
response = requests.get(url, headers=headers, params=url_params) #Your code here
print(response)
print(type(response.text))
print(response.text[:1000])
```

```
<Response [200]>
<class 'str'>
{"businesses": [{"id": "ysqgdbSrezXgVwER2kQWKA", "alias": "julianas-pizza-brooklyn-5", "name": "Juliana's Pizza", "image_ur
l": "https://s3-media1.fl.yelpcdn.com/bphoto/7JtwTxhWHf3YS70Ss_CfxA/o.jpg", "is_closed": false, "url": "https://www.yelp.co
m/biz/julianas-pizza-brooklyn-5?adjust_creative=xNHtXRpNa-MXGFJJTHHUvw&utm_campaign=yelp_api_v3&utm_medium=api_v3_business_s
earch&utm_source=xNHtXRpNa-MXGFJJTHHUvw", "review_count": 1882, "categories": [{"alias": "pizza", "title": "Pizza"}], "ratin
g": 4.5, "coordinates": {"latitude": 40.7026153030093, "longitude": -73.9934159993549}, "transactions": [], "price": "$$",
"location": {"address1": "19 Old Fulton St", "address2": "", "address3": "", "city": "Brooklyn", "zip_code": "11201", "count
ry": "US", "state": "NY", "display_address": ["19 Old Fulton St", "Brooklyn, NY 11201"]}, "phone": "+17185966700", "display_
phone": "(718) 596-6700", "distance": 323.20506308227306}, {"id": "WIhm0W9197f_rRtDziq5qQ", "alias": "lombardis-pizza-new-yo
rk-4", "nam
```

In [2]:
```
len(response.json()['businesses'])
```

Out[2]: 20

In [3]:
```
response.json()['total']
```

Out[3]: 10600

Pagination

Now that you have an initial response, you can examine the contents of the json container. For example, you might start with `response.josn().keys()`. Here, you'll see a key for `'total'`, which tells you the full number of matching results given your query parameters. Write a loop (or ideally a function) which then makes successive API calls using the offset parameter to retrieve all of the results (or 5000 for a particularly large result set) for the original query. As you do this, be mindful of how you store the data. Your final goal will be to reformat the data concerning the businesses themselves into a pandas DataFrame from the json objects.

Note: be mindful of the API rate limits. You can only make 5000 requests per day, and are also can make requests too fast. Start prototyping small before running a loop that could be faulty. You can also use time.sleep(n) to add delays. For more details see https://www.yelp.com/developers/documentation/v3/rate_limiting.

In [4]:
```
# Your code here; use a function or loop to retrieve all the results from your original request
import pandas as pd
import time

def yelp_call(url_params, api_key):
    url = 'https://api.yelp.com/v3/businesses/search'
    headers = {'Authorization': 'Bearer {}'.format(api_key)}
    response = requests.get(url, headers=headers, params=url_params)

    df = pd.DataFrame(response.json()['businesses'])
    return df

def all_results(url_params, api_key):
    num = response.json()['total']
    print('{} total matches found.'.format(num))
    cur = 0
    dfs = []
    while cur < num and cur < 1000:
        url_params['offset'] = cur
        dfs.append(yelp_call(url_params, api_key))
        time.sleep(1) #Wait a second
        cur += 50
    df = pd.concat(dfs, ignore_index=True)
    return df

term = 'pizza'
location = 'Astoria NY'
url_params = {  'term': term.replace(' ', '+'),
                'location': location.replace(' ', '+'),
                'limit' : 50
             }
df = all_results(url_params, api_key)
print(len(df))
df.head()
```

```
10600 total matches found.
1000
```

Out[4]:

	alias	categories	coordinates	display_phone	distance	id	image_url	is_closed	l
0	rizzos-fine-pizza-astoria	[{'alias': 'pizza', 'title': 'Pizza'}]	{'latitude': 40.76335, 'longitude': -73.91516}	(718) 721-9862	713.161109	hB2S1y5T9ufMz5ksHHhdIA	https://s3-media3.fl.yelpcdn.com/bphoto/p1IQ1w...	False	{'ad... S 'ad...
1	milkflower-astoria	[{'alias': 'pizza', 'title': 'Pizza'}]	{'latitude': 40.7628961, 'longitude': -73.9207...	(718) 204-1300	461.294783	1OZPOWZwJr-hpvMt6TD4ug	https://s3-media3.fl.yelpcdn.com/bphoto/SQG1kI...	False	{'ad... '34 'ad...
2	sacs-place-astoria	[{'alias': 'pizza', 'title': 'Pizza'}, {'alias...	{'latitude': 40.7628982173022, 'longitude': -7...	(718) 204-5002	605.395513	Y3xDbSHXs6apMA4JLGJZ1Q	https://s3-media3.fl.yelpcdn.com/bphoto/DvDlQ_...	False	{'ad... Bro 'ad...
3	rosarios-astoria-2	[{'alias': 'delis', 'title': 'Delis'}, {'alias...	{'latitude': 40.7749914695423, 'longitude': -7...	(718) 728-2920	1245.611380	4q8jF2pKoWYbJXYmSNMG1g	https://s3-media1.fl.yelpcdn.com/bphoto/Apn8RO...	False	{'ad... '22 'ad...
4	napoli-pizza-and-pasta-astoria-4	[{'alias': 'pizza', 'title': 'Pizza'}, {'alias...	{'latitude': 40.75718, 'longitude': -73.92686}	(718) 472-1146	1148.015816	Lw58gDv9ipwtOcg2ElJp3w	https://s3-media3.fl.yelpcdn.com/bphoto/os_Qgp...	False	{'ad... '33- 'ad...

Exploratory Analysis

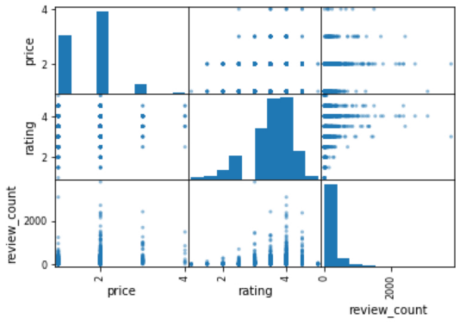
Take the restaurants from the previous question and do an intial exploratory analysis. At minimum, this should include looking at the distribution of features such as price, rating and number of reviews as well as the relations between these dimensions.

In [5]:
```
import matplotlib.pyplot as plt
%matplotlib inline

df.price = df.price.fillna(value=0)
price_dict = {"$": 1, "$$":2, "$$$": 3, "$$$$":4}
df.price = df.price.map(price_dict)

pd.plotting.scatter_matrix(df[['price', 'rating', 'review_count']])
```

Out[5]:
```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x10a8e0400>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x10a6ffbe0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x1130699b0>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x11309a080>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x1130c1710>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x1130c1748>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x113119470>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x11313e9e8>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x1131720b8>]],
      dtype=object)
```



Mapping

Look at the initial Yelp example and try and make a map using Folium of the restaurants you retrieved. Be sure to also add popups to the markers giving some basic information such as name, rating and price.

In [25]:
```
#Your code here
import folium

lat_long = df['coordinates'].iloc[0]
lat = lat_long['latitude']
long = lat_long['longitude']
yelp_map = folium.Map([lat, long])

for row in df.index:
    try:
        lat_long = df['coordinates'][row]
        lat = lat_long['latitude']
        long = lat_long['longitude']
        name = df['name'][row]
        rating = df['rating'][row]
        price = df['price'][row]
        details = "{}\nPrice: {} Rating:{}".format(name,str(price),str(rating))
        popup = folium.Popup(details, parse_html=True)
        marker = folium.Marker([lat, long], popup=popup)
        marker.add_to(yelp_map)
    except:
        print('Hit error on row: {}'.format(row))
yelp_map
```

Out[25]:

Summary

Nice work! In this lab, you synthesized your skills for the day, making multiple API calls to Yelp in order to paginate through a results set, performing some

basic exploratory analysis and then creating a nice map visual to display the results! Well done!