(https://cognitiveclass.ai)

Learning FourSquare API with Python

In [ ]:

```
#by Christopher Harrison
```

Introduction

In this lab, you will learn in details how to make calls to the Foursquare API for different purposes. You will learn how to construct a URL to send a request to the API to search for a specific type of venues, to explore a particular venue, to explore a Foursquare user, to explore a geographical location, and to get trending venues around a location. Also, you will learn how to use the visualization library, Folium, to visualize the results.

Table of Contents

Import necessary Libraries

In [1]:

```
import requests # library to handle requests
import pandas as pd # library for data analsysis
import numpy as np # library to handle data in a vectorized manner
import random # library for random number generation

!conda install -c conda-forge geopy --yes
from geopy.geocoders import Nominatim # module to convert an address into latitude
 and longitude values

# libraries for displaying images
from IPython.display import Image
from IPython.core.display import HTML

# tranforming json file into a pandas dataframe library
from pandas.io.json import json_normalize

!conda install -c conda-forge folium=0.5.0 --yes
import folium # plotting library

print('Folium installed')
print('Libraries imported.')
```

```
Collecting package metadata (current_repodata.json): done
Solving environment: done

## Package Plan ##

  environment location: /home/jupyterlab/conda/envs/python

  added / updated specs:
    - geopy


The following packages will be downloaded:

    package                    |            build
    ---------------------------|-----------------
    certifi-2020.12.5          |   py36h5fab9bb_1         143 KB  conda
-forge
    geographiclib-1.50         |             py_0          34 KB  conda
-forge
    geopy-2.1.0                |     pyhd3deb0d_0          64 KB  conda
-forge
    ------------------------------------------------------------
                                           Total:         240 KB

The following NEW packages will be INSTALLED:

  geographiclib      conda-forge/noarch::geographiclib-1.50-py_0
  geopy              conda-forge/noarch::geopy-2.1.0-pyhd3deb0d_0

The following packages will be UPDATED:

  certifi                           2020.12.5-py36h5fab9bb_0 --> 2020.1
2.5-py36h5fab9bb_1



Downloading and Extracting Packages
geopy-2.1.0           | 64 KB     | ####################################
# | 100%
certifi-2020.12.5     | 143 KB    | ####################################
# | 100%
geographiclib-1.50    | 34 KB     | ####################################
# | 100%
Preparing transaction: done
Verifying transaction: done
Executing transaction: done
Collecting package metadata (current_repodata.json): done
Solving environment: failed with initial frozen solve. Retrying with fl
exible solve.
Collecting package metadata (repodata.json): done
Solving environment: done

## Package Plan ##

  environment location: /home/jupyterlab/conda/envs/python

  added / updated specs:
```

```
    - folium=0.5.0


  The following packages will be downloaded:

      package                    |            build
      ---------------------------|-----------------
      altair-4.1.0               |             py_1         614 KB  conda
  -forge
      branca-0.4.2               |     pyhd8ed1ab_0          26 KB  conda
  -forge
      folium-0.5.0               |             py_0          45 KB  conda
  -forge
      pandas-1.1.5               |   py36h284efc9_0        11.3 MB  conda
  -forge
      pytz-2020.5                |     pyhd8ed1ab_0         244 KB  conda
  -forge
      toolz-0.11.1               |             py_0          46 KB  conda
  -forge
      vincent-0.4.4              |             py_1          28 KB  conda
  -forge
      ------------------------------------------------------------
                                             Total:        12.3 MB

  The following NEW packages will be INSTALLED:

    altair             conda-forge/noarch::altair-4.1.0-py_1
    branca             conda-forge/noarch::branca-0.4.2-pyhd8ed1ab_0
    folium             conda-forge/noarch::folium-0.5.0-py_0
    pandas             conda-forge/linux-64::pandas-1.1.5-py36h284efc9_0
    pytz               conda-forge/noarch::pytz-2020.5-pyhd8ed1ab_0
    toolz              conda-forge/noarch::toolz-0.11.1-py_0
    vincent            conda-forge/noarch::vincent-0.4.4-py_1



  Downloading and Extracting Packages
  folium-0.5.0         | 45 KB     | ####################################
  # | 100%
  branca-0.4.2         | 26 KB     | ####################################
  # | 100%
  altair-4.1.0         | 614 KB    | ####################################
  # | 100%
  pandas-1.1.5         | 11.3 MB   | ####################################
  # | 100%
  pytz-2020.5          | 244 KB    | ####################################
  # | 100%
  toolz-0.11.1         | 46 KB     | ####################################
  # | 100%
  vincent-0.4.4        | 28 KB     | ####################################
  # | 100%
  Preparing transaction: done
  Verifying transaction: done
  Executing transaction: done
  Folium installed
  Libraries imported.
```

Define Foursquare Credentials and Version

Make sure that you have created a Foursquare developer account and have your credentials handy

In [2]:

```
CLIENT_ID = 'RNBDTUINUPN3JXITAGQQXQC5ZGV3UDW5KX20XPH3W5E4CYB4' # your Foursquare ID
CLIENT_SECRET = 'JWRBHQLEBVO4JPEODHEZIKUISPCWGRHBGXQ5M3CIZWEBYROM' # your Foursquare Secret
VERSION = '20180604'
LIMIT = 30
print('Your credentails:')
print('CLIENT_ID: ' + CLIENT_ID)
print('CLIENT_SECRET:' + CLIENT_SECRET)
```

```
Your credentails:
CLIENT_ID: RNBDTUINUPN3JXITAGQQXQC5ZGV3UDW5KX20XPH3W5E4CYB4
CLIENT_SECRET:JWRBHQLEBVO4JPEODHEZIKUISPCWGRHBGXQ5M3CIZWEBYROM
```

Let's again assume that you are staying at the Conrad hotel. So let's start by converting the Contrad Hotel's address to its latitude and longitude coordinates.

In order to define an instance of the geocoder, we need to define a user_agent. We will name our agent *foursquare_agent*, as shown below.

In [3]:

```
address = '102 North End Ave, New York, NY'

geolocator = Nominatim(user_agent="foursquare_agent")
location = geolocator.geocode(address)
latitude = location.latitude
longitude = location.longitude
print(latitude, longitude)
```

```
40.7149555 -74.0153365
```

1. Search for a specific venue category

```
https://api.foursquare.com/v2/venues/ search ?
client_id= CLIENT_ID &client_secret= CLIENT_SECRET &ll= LATITUDE , LONGITUDE &v=
```

Now, let's assume that it is lunch time, and you are craving Italian food. So, let's define a query to search for Italian food that is within 500 metres from the Conrad Hotel.

In [4]:

```
search_query = 'Italian'
radius = 500
print(search_query + ' .... OK!')
```

```
Italian .... OK!
```

Define the corresponding URL

In [5]:

```
url = 'https://api.foursquare.com/v2/venues/search?client_id={}&client_secret={}&ll={},{}&v={}&query={}&radius={}&limit={}'.format(CLIENT_ID, CLIENT_SECRET, latitude, longitude, VERSION, search_query, radius, LIMIT)
url
```

Out[5]:

```
'https://api.foursquare.com/v2/venues/search?client_id=RNBDTUINUPN3JXIT
AGQQXQC5ZGV3UDW5KX20XPH3W5E4CYB4&client_secret=JWRBHQLEBVO4JPEODHEZIKUI
SPCWGRHBGXQ5M3CIZWEBYROM&ll=40.7149555,-74.0153365&v=20180604&query=Ita
lian&radius=500&limit=30'
```

Send the GET Request and examine the results

In [6]:

```
results = requests.get(url).json()
results
```

Out[6]:


```
{'meta': {'code': 200, 'requestId': '5ff6b0692686db5d8c5951f0'},
 'response': {'venues': [{'id': '4fa862b3e4b0ebff2f749f06',
    'name': "Harry's Italian Pizza Bar",
    'location': {'address': '225 Murray St',
     'lat': 40.71521779064671,
     'lng': -74.01473940209351,
     'labeledLatLngs': [{'label': 'display',
       'lat': 40.71521779064671,
       'lng': -74.01473940209351},
      {'label': 'entrance', 'lat': 40.715361, 'lng': -74.014975}],
     'distance': 58,
     'postalCode': '10282',
     'cc': 'US',
     'city': 'New York',
     'state': 'NY',
     'country': 'United States',
     'formattedAddress': ['225 Murray St',
      'New York, NY 10282',
      'United States']},
    'categories': [{'id': '4bf58dd8d48988d1ca941735',
      'name': 'Pizza Place',
      'pluralName': 'Pizza Places',
      'shortName': 'Pizza',
      'icon': {'prefix': 'https://ss3.4sqi.net/img/categories_v2/food/p
izza_',
       'suffix': '.png'},
      'primary': True}],
    'referralId': 'v-1610002537',
    'hasPerk': False},
   {'id': '4f3232e219836c91c7bfde94',
    'name': 'Conca Cucina Italian Restaurant',
    'location': {'address': '63 W Broadway',
     'lat': 40.714484000000006,
     'lng': -74.00980600000001,
     'labeledLatLngs': [{'label': 'display',
       'lat': 40.714484000000006,
       'lng': -74.00980600000001}],
     'distance': 469,
     'postalCode': '10007',
     'cc': 'US',
     'city': 'New York',
     'state': 'NY',
     'country': 'United States',
     'formattedAddress': ['63 W Broadway',
      'New York, NY 10007',
      'United States']},
    'categories': [{'id': '4d4b7105d754a06374d81259',
      'name': 'Food',
      'pluralName': 'Food',
      'shortName': 'Food',
      'icon': {'prefix': 'https://ss3.4sqi.net/img/categories_v2/food/d
efault_',
       'suffix': '.png'},
      'primary': True}],
    'referralId': 'v-1610002537',
```

```
      'hasPerk': False},
     {'id': '3fd66200f964a520f4e41ee3',
      'name': 'Ecco',
      'location': {'address': '124 Chambers St',
       'crossStreet': 'btwn Church St & W Broadway',
       'lat': 40.71533713859952,
       'lng': -74.00884766217825,
       'labeledLatLngs': [{'label': 'display',
         'lat': 40.71533713859952,
         'lng': -74.00884766217825},
        {'label': 'entrance', 'lat': 40.715202, 'lng': -74.008779}],
       'distance': 549,
       'postalCode': '10007',
       'cc': 'US',
       'city': 'New York',
       'state': 'NY',
       'country': 'United States',
       'formattedAddress': ['124 Chambers St (btwn Church St & W Broadwa
y)',
        'New York, NY 10007',
        'United States']},
      'categories': [{'id': '4bf58dd8d48988d110941735',
        'name': 'Italian Restaurant',
        'pluralName': 'Italian Restaurants',
        'shortName': 'Italian',
        'icon': {'prefix': 'https://ss3.4sqi.net/img/categories_v2/food/i
talian_',
         'suffix': '.png'},
        'primary': True}],
      'referralId': 'v-1610002537',
      'hasPerk': False}]}}
```


Get relevant part of JSON and transform it into a *pandas* dataframe

In [7]:

```
# assign relevant part of JSON to venues
venues = results['response']['venues']

# tranform venues into a dataframe
dataframe = json_normalize(venues)
dataframe.head()
```

```
/home/jupyterlab/conda/envs/python/lib/python3.6/site-packages/ipykerne
l_launcher.py:5: FutureWarning: pandas.io.json.json_normalize is deprec
ated, use pandas.json_normalize instead
  """
```

Out[7]:

	id	name	categories	referralId	hasPerk	loca
0	4fa862b3e4b0ebff2f749f06	Harry's Italian Pizza Bar	[{'id': '4bf58dd8d48988d1ca941735', 'name': 'P...	v-1610002537	False	2
1	4f3232e219836c91c7bfde94	Conca Cucina Italian Restaurant	[{'id': '4d4b7105d754a06374d81259', 'name': 'F...	v-1610002537	False	63
2	3fd66200f964a520f4e41ee3	Ecco	[{'id': '4bf58dd8d48988d110941735', 'name': 'I...	v-1610002537	False	124

Define information of interest and filter dataframe

In [8]:

```
# keep only columns that include venue name, and anything that is associated with location
filtered_columns = ['name', 'categories'] + [col for col in dataframe.columns if col.startswith('location.')] + ['id']
dataframe_filtered = dataframe.loc[:, filtered_columns]

# function that extracts the category of the venue
def get_category_type(row):
    try:
        categories_list = row['categories']
    except:
        categories_list = row['venue.categories']

    if len(categories_list) == 0:
        return None
    else:
        return categories_list[0]['name']

# filter the category for each row
dataframe_filtered['categories'] = dataframe_filtered.apply(get_category_type, axis=1)

# clean column names by keeping only last term
dataframe_filtered.columns = [column.split('.')[-1] for column in dataframe_filtered.columns]

dataframe_filtered
```

Out[8]:

	name	categories	address	lat	lng	labeledLatLngs	distance	postal
0	Harry's Italian Pizza Bar	Pizza Place	225 Murray St	40.715218	-74.014739	[{'label': 'display', 'lat': 40.71521779064671...	58	
1	Conca Cucina Italian Restaurant	Food	63 W Broadway	40.714484	-74.009806	[{'label': 'display', 'lat': 40.71448400000000...	469	
2	Ecco	Italian Restaurant	124 Chambers St	40.715337	-74.008848	[{'label': 'display', 'lat': 40.71533713859952...	549	

Let's visualize the Italian restaurants that are nearby

In [9]:

```
dataframe_filtered.name
```

Out[9]:

```
0          Harry's Italian Pizza Bar
1    Conca Cucina Italian Restaurant
2                               Ecco
Name: name, dtype: object
```

In [10]:

```
venues_map = folium.Map(location=[latitude, longitude], zoom_start=13) # generate m
ap centred around the Conrad Hotel

# add a red circle marker to represent the Conrad Hotel
folium.features.CircleMarker(
    [latitude, longitude],
    radius=10,
    color='red',
    popup='Conrad Hotel',
    fill = True,
    fill_color = 'red',
    fill_opacity = 0.6
).add_to(venues_map)

# add the Italian restaurants as blue circle markers
for lat, lng, label in zip(dataframe_filtered.lat, dataframe_filtered.lng, datafram
e_filtered.categories):
    folium.features.CircleMarker(
        [lat, lng],
        radius=5,
        color='blue',
        popup=label,
        fill = True,
        fill_color='blue',
        fill_opacity=0.6
    ).add_to(venues_map)

# display map
venues_map
```

Out[10]:

Make this Notebook Trusted to load map: File -> Trust Notebook

2. Explore a Given Venue

> https://api.foursquare.com/v2/venues/ **VENUE_ID** ?
> client_id= **CLIENT_ID** &client_secret= **CLIENT_SECRET** &v= **VERSION**

A. Let's explore the closest Italian restaurant -- *Harry's Italian Pizza Bar*

In [11]:

```
venue_id = '4fa862b3e4b0ebff2f749f06' # ID of Harry's Italian Pizza Bar
url = 'https://api.foursquare.com/v2/venues/{}?client_id={}&client_secret={}&v={}'.
format(venue_id, CLIENT_ID, CLIENT_SECRET, VERSION)
url
```

Out[11]:

```
'https://api.foursquare.com/v2/venues/4fa862b3e4b0ebff2f749f06?client_i
d=RNBDTUINUPN3JXITAGQQXQC5ZGV3UDW5KX20XPH3W5E4CYB4&client_secret=JWRBHQ
LEBVO4JPEODHEZIKUISPCWGRHBGXQ5M3CIZWEBYROM&v=20180604'
```

Send GET request for result

In [12]:

```
result = requests.get(url).json()
print(result['response']['venue'].keys())
result['response']['venue']
```

```
dict_keys(['id', 'name', 'contact', 'location', 'canonicalUrl', 'catego
ries', 'verified', 'stats', 'url', 'price', 'hasMenu', 'likes', 'dislik
e', 'ok', 'rating', 'ratingColor', 'ratingSignals', 'menu', 'allowMenuU
rlEdit', 'beenHere', 'specials', 'photos', 'reasons', 'hereNow', 'creat
edAt', 'tips', 'shortUrl', 'timeZone', 'listed', 'hours', 'popular', 's
easonalHours', 'defaultHours', 'pageUpdates', 'inbox', 'attributes', 'b
estPhoto', 'colors'])
```

Out[12]:


```
{'id': '4fa862b3e4b0ebff2f749f06',
 'name': "Harry's Italian Pizza Bar",
 'contact': {'phone': '2126081007', 'formattedPhone': '(212) 608-100
7'},
 'location': {'address': '225 Murray St',
  'lat': 40.71521779064671,
  'lng': -74.01473940209351,
  'labeledLatLngs': [{'label': 'display',
    'lat': 40.71521779064671,
    'lng': -74.01473940209351},
   {'label': 'entrance', 'lat': 40.715361, 'lng': -74.014975}],
  'postalCode': '10282',
  'cc': 'US',
  'city': 'New York',
  'state': 'NY',
  'country': 'United States',
  'formattedAddress': ['225 Murray St',
   'New York, NY 10282',
   'United States']},
 'canonicalUrl': 'https://foursquare.com/v/harrys-italian-pizza-bar/4fa
862b3e4b0ebff2f749f06',
 'categories': [{'id': '4bf58dd8d48988d1ca941735',
   'name': 'Pizza Place',
   'pluralName': 'Pizza Places',
   'shortName': 'Pizza',
   'icon': {'prefix': 'https://ss3.4sqi.net/img/categories_v2/food/pizz
a_',
    'suffix': '.png'},
   'primary': True},
  {'id': '4bf58dd8d48988d110941735',
   'name': 'Italian Restaurant',
   'pluralName': 'Italian Restaurants',
   'shortName': 'Italian',
   'icon': {'prefix': 'https://ss3.4sqi.net/img/categories_v2/food/ital
ian_',
    'suffix': '.png'}}],
 'verified': False,
 'stats': {'tipCount': 56},
 'url': 'http://harrysitalian.com',
 'price': {'tier': 2, 'message': 'Moderate', 'currency': '$'},
 'hasMenu': True,
 'likes': {'count': 120,
  'groups': [{'type': 'others', 'count': 120, 'items': []}],
  'summary': '120 Likes'},
 'dislike': False,
 'ok': False,
 'rating': 6.9,
 'ratingColor': 'FFC800',
 'ratingSignals': 211,
 'menu': {'type': 'Menu',
  'label': 'Menu',
  'anchor': 'View Menu',
  'url': 'https://foursquare.com/v/harrys-italian-pizza-bar/4fa862b3e4b
0ebff2f749f06/menu',
  'mobileUrl': 'https://foursquare.com/v/4fa862b3e4b0ebff2f749f06/devic
```

```
e_menu'},
 'allowMenuUrlEdit': True,
 'beenHere': {'count': 0,
  'unconfirmedCount': 0,
  'marked': False,
  'lastCheckinExpiredAt': 0},
 'specials': {'count': 0, 'items': []},
 'photos': {'count': 146,
  'groups': [{'type': 'venue',
    'name': 'Venue photos',
    'count': 146,
    'items': [{'id': '4fad980de4b091b4626c3633',
      'createdAt': 1336776717,
      'source': {'name': 'Foursquare for Android',
       'url': 'https://foursquare.com/download/#/android'},
      'prefix': 'https://fastly.4sqi.net/img/general/',
      'suffix': '/ya1iQFI7pLjuIJp1PGDKlrZS3OJdHCF7tpILMmjv_2w.jpg',
      'width': 480,
      'height': 640,
      'user': {'firstName': 'Leony', 'lastName': 'N', 'countryCode': 'U
S'},
      'visibility': 'public'}]}]},
 'reasons': {'count': 1,
  'items': [{'summary': 'Lots of people like this place',
    'type': 'general',
    'reasonName': 'rawLikesReason'}]},
 'hereNow': {'count': 0, 'summary': 'Nobody here', 'groups': []},
 'createdAt': 1336435379,
 'tips': {'count': 56,
  'groups': [{'type': 'others',
    'name': 'All tips',
    'count': 56,
    'items': [{'id': '53d27909498e0523841340b6',
      'createdAt': 1406302473,
      'text': "Harry's Italian Pizza bar is known for it's amazing pizz
a, but did you know that the brunches here are amazing too? Try the Nut
ella French toast and we know you'll be sold.",
      'type': 'user',
      'canonicalUrl': 'https://foursquare.com/item/53d27909498e05238413
40b6',
      'lang': 'en',
      'likes': {'count': 4,
       'groups': [{'type': 'others',
         'count': 4,
         'items': [{'firstName': 'Diane',
           'lastName': 'D',
           'countryCode': 'US'},
          {'firstName': 'Tim', 'lastName': 'S', 'countryCode': 'US'},
          {'firstName': 'TenantKing.com',
           'countryCode': 'US',
           'type': 'page'}]}],
       'summary': '4 likes'},
      'logView': True,
      'agreeCount': 3,
      'disagreeCount': 0,
      'todo': {'count': 0},
      'user': {'firstName': 'TenantKing.com',
```

```
        'countryCode': 'US',
        'type': 'page'}}]}]},
  'shortUrl': 'http://4sq.com/JNblHV',
  'timeZone': 'America/New_York',
  'listed': {'count': 54,
   'groups': [{'type': 'others',
     'name': 'Lists from other people',
     'count': 54,
     'items': [{'id': '4fa32fd0e4b04193744746b1',
       'name': 'Manhattan Haunts',
       'description': '',
       'type': 'others',
       'user': {'firstName': 'Becca', 'lastName': 'M', 'countryCode': 'U
S'},
       'editable': False,
       'public': True,
       'collaborative': False,
       'url': '/becca_mcarthur/list/manhattan-haunts',
       'canonicalUrl': 'https://foursquare.com/becca_mcarthur/list/manha
ttan-haunts',
       'createdAt': 1336094672,
       'updatedAt': 1380845377,
       'photo': {'id': '4e8cc9461081e3b3544e12e5',
        'createdAt': 1317849414,
        'prefix': 'https://fastly.4sqi.net/img/general/',
        'suffix': '/0NLVU2HC1JF4DXIMKWUFW3QBUT31DC11EFNYYHMJG3NDWAPS.jp
g',
        'width': 492,
        'height': 330,
        'user': {'firstName': 'Time Out New York',
         'countryCode': 'US',
         'type': 'page'},
        'visibility': 'public'},
       'followers': {'count': 22},
       'listItems': {'count': 187,
        'items': [{'id': 'v4fa862b3e4b0ebff2f749f06',
          'createdAt': 1342934485}]}},
      {'id': '4fae817be4b085f6b2a74d19',
       'name': 'USA NYC MAN FiDi',
       'description': 'Where to go for decent eats in the restaurant was
teland of Downtown NYC aka FiDi, along with Tribeca & Battery Park Cit
y.',
       'type': 'others',
       'user': {'firstName': 'Kino', 'countryCode': 'US'},
       'editable': False,
       'public': True,
       'collaborative': False,
       'url': '/kinosfault/list/usa-nyc-man-fidi',
       'canonicalUrl': 'https://foursquare.com/kinosfault/list/usa-nyc-m
an-fidi',
       'createdAt': 1336836475,
       'updatedAt': 1556754919,
       'photo': {'id': '55984992498e13ba75e353bb',
        'createdAt': 1436043666,
        'prefix': 'https://fastly.4sqi.net/img/general/',
        'suffix': '/12113441_iOa6Uh-Xi8bhj2-gpzkkw8MKiAIs7RmOcz_RM7m8in
k.jpg',
```

```
        'width': 540,
        'height': 960,
        'user': {'firstName': 'Kino', 'countryCode': 'US'},
        'visibility': 'public'},
       'followers': {'count': 20},
       'listItems': {'count': 273,
        'items': [{'id': 'v4fa862b3e4b0ebff2f749f06',
          'createdAt': 1373909433}]}},
     {'id': '4fddeff0e4b0e078037ac0d3',
      'name': 'NYC Resturants',
      'description': '',
      'type': 'others',
      'user': {'firstName': 'Richard', 'lastName': 'R', 'countryCode':
'US'},
      'editable': False,
      'public': True,
      'collaborative': True,
      'url': '/rickr7/list/nyc-resturants',
      'canonicalUrl': 'https://foursquare.com/rickr7/list/nyc-resturant
s',
      'createdAt': 1339944944,
      'updatedAt': 1607653900,
      'photo': {'id': '5072dd13e4b09145cdf782d1',
       'createdAt': 1349704979,
       'prefix': 'https://fastly.4sqi.net/img/general/',
       'suffix': '/208205_fGh2OuAZ9qJ4agbAA5wMVNOSIm9kNUlRtNwj1N-adqg.j
pg',
       'width': 800,
       'height': 800,
       'user': {'firstName': 'Thalia', 'lastName': 'K', 'countryCode':
'US'},
       'visibility': 'public'},
      'followers': {'count': 12},
      'listItems': {'count': 186,
       'items': [{'id': 'v4fa862b3e4b0ebff2f749f06',
         'createdAt': 1581655865}]}},
     {'id': '5266c68a498e7c667807fe09',
      'name': 'Foodie Love in NY - 02',
      'description': '',
      'type': 'others',
      'user': {'firstName': 'WiLL', 'countryCode': 'CN'},
      'editable': False,
      'public': True,
      'collaborative': False,
      'url': '/sweetiewill/list/foodie-love-in-ny--02',
      'canonicalUrl': 'https://foursquare.com/sweetiewill/list/foodie-l
ove-in-ny--02',
      'createdAt': 1382467210,
      'updatedAt': 1391995585,
      'followers': {'count': 7},
      'listItems': {'count': 200,
       'items': [{'id': 'v4fa862b3e4b0ebff2f749f06',
         'createdAt': 1386809936}]}}]}],
 'hours': {'status': 'Closed until 11:30 AM',
  'richStatus': {'entities': [], 'text': 'Closed until 11:30 AM'},
  'isOpen': False,
  'isLocalHoliday': False,
```

```
    'dayData': [],
    'timeframes': [{'days': 'Mon–Wed, Sun',
      'open': [{'renderedTime': '11:30 AM–11:00 PM'}],
      'segments': []},
     {'days': 'Thu–Sat',
      'includesToday': True,
      'open': [{'renderedTime': '11:30 AM–Midnight'}],
      'segments': []}]},
   'popular': {'isOpen': False,
    'isLocalHoliday': False,
    'timeframes': [{'days': 'Today',
      'includesToday': True,
      'open': [{'renderedTime': 'Noon–2:00 PM'},
       {'renderedTime': '5:00 PM–10:00 PM'}],
      'segments': []},
     {'days': 'Fri',
      'open': [{'renderedTime': 'Noon–3:00 PM'},
       {'renderedTime': '5:00 PM–11:00 PM'}],
      'segments': []},
     {'days': 'Sat',
      'open': [{'renderedTime': 'Noon–11:00 PM'}],
      'segments': []},
     {'days': 'Sun',
      'open': [{'renderedTime': 'Noon–3:00 PM'},
       {'renderedTime': '5:00 PM–8:00 PM'}],
      'segments': []},
     {'days': 'Mon',
      'open': [{'renderedTime': 'Noon–2:00 PM'},
       {'renderedTime': '6:00 PM–8:00 PM'}],
      'segments': []},
     {'days': 'Tue–Wed',
      'open': [{'renderedTime': 'Noon–2:00 PM'},
       {'renderedTime': '5:00 PM–10:00 PM'}],
      'segments': []}]},
   'seasonalHours': [],
   'defaultHours': {'status': 'Closed until 11:30 AM',
    'richStatus': {'entities': [], 'text': 'Closed until 11:30 AM'},
    'isOpen': False,
    'isLocalHoliday': False,
    'dayData': [],
    'timeframes': [{'days': 'Mon–Wed, Sun',
      'open': [{'renderedTime': '11:30 AM–11:00 PM'}],
      'segments': []},
     {'days': 'Thu–Sat',
      'includesToday': True,
      'open': [{'renderedTime': '11:30 AM–Midnight'}],
      'segments': []}]},
   'pageUpdates': {'count': 0, 'items': []},
   'inbox': {'count': 0, 'items': []},
   'attributes': {'groups': [{'type': 'price',
      'name': 'Price',
      'summary': '$$',
      'count': 1,
      'items': [{'displayName': 'Price', 'displayValue': '$$', 'priceTie
  r': 2}]},
     {'type': 'payments',
      'name': 'Credit Cards',
```

```
     'summary': 'Credit Cards',
     'count': 7,
     'items': [{'displayName': 'Credit Cards',
       'displayValue': 'Yes (incl. American Express)'}]},
    {'type': 'outdoorSeating',
     'name': 'Outdoor Seating',
     'summary': 'Outdoor Seating',
     'count': 1,
     'items': [{'displayName': 'Outdoor Seating', 'displayValue': 'Ye
s'}]},
    {'type': 'serves',
     'name': 'Menus',
     'summary': 'Happy Hour, Brunch & more',
     'count': 8,
     'items': [{'displayName': 'Brunch', 'displayValue': 'Brunch'},
      {'displayName': 'Lunch', 'displayValue': 'Lunch'},
      {'displayName': 'Dinner', 'displayValue': 'Dinner'},
      {'displayName': 'Happy Hour', 'displayValue': 'Happy Hour'}]},
    {'type': 'drinks',
     'name': 'Drinks',
     'summary': 'Beer, Wine & Cocktails',
     'count': 5,
     'items': [{'displayName': 'Beer', 'displayValue': 'Beer'},
      {'displayName': 'Wine', 'displayValue': 'Wine'},
      {'displayName': 'Cocktails', 'displayValue': 'Cocktails'}]}]},
  'bestPhoto': {'id': '4fad980de4b091b4626c3633',
   'createdAt': 1336776717,
   'source': {'name': 'Foursquare for Android',
    'url': 'https://foursquare.com/download/#/android'},
   'prefix': 'https://fastly.4sqi.net/img/general/',
   'suffix': '/ya1iQFI7pLjuIJp1PGDKlrZS3OJdHCF7tpILMmjv_2w.jpg',
   'width': 480,
   'height': 640,
   'visibility': 'public'},
  'colors': {'highlightColor': {'photoId': '4fad980de4b091b4626c3633',
    'value': -13619152},
   'highlightTextColor': {'photoId': '4fad980de4b091b4626c3633', 'valu
e': -1},
   'algoVersion': 3}}
```


B. Get the venue's overall rating

In [13]:

```
try:
    print(result['response']['venue']['rating'])
except:
    print('This venue has not been rated yet.')
```

```
6.9
```

That is not a very good rating. Let's check the rating of the second closest Italian restaurant.

In [14]:

```
venue_id = '4f3232e219836c91c7bfde94' # ID of Conca Cucina Italian Restaurant
url = 'https://api.foursquare.com/v2/venues/{}?client_id={}&client_secret={}&v={}'.
format(venue_id, CLIENT_ID, CLIENT_SECRET, VERSION)

result = requests.get(url).json()
try:
    print(result['response']['venue']['rating'])
except:
    print('This venue has not been rated yet.')
```

```
This venue has not been rated yet.
```

Since this restaurant has no ratings, let's check the third restaurant.

In [15]:

```
venue_id = '3fd66200f964a520f4e41ee3' # ID of Ecco
url = 'https://api.foursquare.com/v2/venues/{}?client_id={}&client_secret={}&v={}'.
format(venue_id, CLIENT_ID, CLIENT_SECRET, VERSION)

result = requests.get(url).json()
try:
    print(result['response']['venue']['rating'])
except:
    print('This venue has not been rated yet.')
```

```
7.3
```

Since this restaurant has a slightly better rating, let's explore it further.

C. Get the number of tips

In [16]:

```
result['response']['venue']['tips']['count']
```

Out[16]:

```
19
```

D. Get the venue's tips

https://api.foursquare.com/v2/venues/ **VENUE_ID** /tips?
client_id= **CLIENT_ID** &client_secret= **CLIENT_SECRET** &v= **VERSION** &limit= **LIMIT**

Create URL and send GET request. Make sure to set limit to get all tips

In [17]:

```
## Ecco Tips
limit = 15 # set limit to be greater than or equal to the total number of tips
url = 'https://api.foursquare.com/v2/venues/{}/tips?client_id={}&client_secret={}&v
={}&limit={}'.format(venue_id, CLIENT_ID, CLIENT_SECRET, VERSION, limit)

results = requests.get(url).json()
results
```

Out[17]:


```
{'meta': {'code': 200, 'requestId': '5ff6b06a76c3723f2219a8e5'},
 'response': {'tips': {'count': 19,
   'items': [{'id': '5ab1cb46c9a517174651d3fe',
     'createdAt': 1521601350,
     'text': 'A+ Italian food! Trust me on this: my mom’s side of the f
amily is 100% Italian. I was born and bred to know good pasta when I se
e it, and Ecco is one of my all-time NYC favorites',
     'type': 'user',
     'canonicalUrl': 'https://foursquare.com/item/5ab1cb46c9a517174651d
3fe',
     'lang': 'en',
     'likes': {'count': 0, 'groups': []},
     'logView': True,
     'agreeCount': 5,
     'disagreeCount': 0,
     'todo': {'count': 0},
     'user': {'firstName': 'Nick', 'lastName': 'E', 'countryCode': 'U
S'},
     'authorInteractionType': 'liked'}]}}
```


Get tips and list of associated features

In [18]:

```
tips = results['response']['tips']['items']

tip = results['response']['tips']['items'][0]
tip.keys()
```

Out[18]:

```
dict_keys(['id', 'createdAt', 'text', 'type', 'canonicalUrl', 'lang',
'likes', 'logView', 'agreeCount', 'disagreeCount', 'todo', 'user', 'aut
horInteractionType'])
```

Format column width and display all tips

Now remember that because we are using a personal developer account, then we can access only 2 of the restaurant's tips, instead of all 15 tips.

3. Search a Foursquare User

```
 https://api.foursquare.com/v2/users/ USER_ID ?
client_id= CLIENT_ID &client_secret= CLIENT_SECRET &v= VERSION
```

Define URL, send GET request and display features associated with user

How many tips has this user submitted?

Wow! So it turns out that Nick is a very active Foursquare user, with more than 250 tips.

Get User's tips

In [23]:

```
# define tips URL
url = 'https://api.foursquare.com/v2/users/{}/tips?client_id={}&client_secret={}&v=
{}&limit={}'.format(user_id, CLIENT_ID, CLIENT_SECRET, VERSION, limit)

# send GET request and get user's tips
results = requests.get(url).json()
tips = results['response']['tips']['items']

# format column width
pd.set_option('display.max_colwidth', -1)

tips_df = json_normalize(tips)

# filter columns
filtered_columns = ['text', 'agreeCount', 'disagreeCount', 'id']
tips_filtered = tips_df.loc[:, filtered_columns]

# display user's tips
tips_filtered
```

```
/home/jupyterlab/conda/envs/python/lib/python3.6/site-packages/ipykerne
l_launcher.py:9: FutureWarning: Passing a negative integer is deprecate
d in version 1.0 and will not be supported in future version. Instead,
use None to not limit the column width.
  if __name__ == '__main__':
/home/jupyterlab/conda/envs/python/lib/python3.6/site-packages/ipykerne
l_launcher.py:11: FutureWarning: pandas.io.json.json_normalize is depre
cated, use pandas.json_normalize instead
  # This is added back by InteractiveShellApp.init_path()
```

Out[23]:

	text	agreeCount	disagreeCount	id
0	They serve coffee!!!!!!	1	0	5accc98c0313204c9d7ec157
1	Quick, cheap lunch that tastes good! Way shorter line than Chipotle, too.	2	0	5acbec70a0215b732e264fe8
2	You're not a real New Yorker until you've shame-ordered Insomnia Cookies for delivery at 3am	1	0	5acbbd4eb1538e45373b07f5
3	Good for you yet still tasty! Clean green protein is my go-to after I hit the gym 💪	2	0	5acbbcda01235808d5d6dc75
4	Burger game strong 💪	1	0	5ab575fb6bdee65f759da8c1
5	Great burgers & fries! Also, this place is exactly what it's like when you go to a bar in the Southwest. Source: I'm from Arizona.	2	0	5ab5575d73fe2516ad8f363b
6	Açaí bowl + peanut butter + whey protein = 💪💪💪	1	0	5ab42db53c858d64af2688a4
7	Highly underrated and way less crowded than Central Park!	3	0	5ab42c396f706a29f53ad1a8
8	Get the açaí bowl with peanut butter after your work out and thank me later 👌	1	0	5ab42aca2a7ab6333652b266
9	When you want a burger, this should be the first thing that comes to mind. A+!	1	0	5ab42a28da5e5617d18e3a6a
10	Way less crowded than Central Park! People who live in the neighborhood rave about Carl Schurz Park.	3	0	5ab429db1ffe971b060083f5
11	The best Mexican food in the Murray Hill / Kips Bay area!	1	0	5ab3f53f8496ca57542d5549
12	Best coffee shop in the neighborhood!	1	0	5ab3f428da5e5617d17d1475
13	When there's nice weather, the rooftop at Tonic East is the best place to watch the game. Perfect for March Madness & NBA finals!	2	0	5ab3f3fedd70c572de886c9d
14	Rib game level 💯	1	0	5ab3f372da2e00604ca53924

Let's get the venue for the tip with the greatest number of agree counts

Get User's friends

Interesting. Despite being very active, it turns out that Nick does not have any friends on Foursquare. This might definitely change in the future.

Retrieve the User's Profile Image

In [ ]:

```
# 1. grab prefix of photo
# 2. grab suffix of photo
# 3. concatenate them using the image size
Image(url='https://igx.4sqi.net/img/user/300x300/484542633_mK2Yum7T_7Tn9fWpndidJsmw
2Hof_6T5vJBKCHPLMK5OL-U5ZiJGj51iwBstcpDLYa3Zvhvis.jpg')
```

4. Explore a location

```
https://api.foursquare.com/v2/venues/ explore ?
client_id= CLIENT_ID &client_secret= CLIENT_SECRET &ll= LATITUDE , LONGITUDE &v=
```

So, you just finished your gourmet dish at Ecco, and are just curious about the popular spots around the restaurant. In order to explore the area, let's start by getting the latitude and longitude values of Ecco Restaurant.

In [27]:

```
latitude = 40.715337
longitude = -74.008848
```

Define URL

In [28]:

```
url = 'https://api.foursquare.com/v2/venues/explore?client_id={}&client_secret={}&ll={},{}&v={}&radius={}&limit={}'.format(CLIENT_ID, CLIENT_SECRET, latitude, longitude, VERSION, radius, LIMIT)
url
```

Out[28]:

```
'https://api.foursquare.com/v2/venues/explore?client_id=RNBDTUINUPN3JXI
TAGQQXQC5ZGV3UDW5KX20XPH3W5E4CYB4&client_secret=JWRBHQLEBVO4JPEODHEZIKU
ISPCWGRHBGXQ5M3CIZWEBYROM&ll=40.715337,-74.008848&v=20180604&radius=500
&limit=30'
```

Send GET request and examine results

In [29]:

```
import requests
```

In [30]:

```
results = requests.get(url).json()
'There are {} around Ecco restaurant.'.format(len(results['response']['groups'][0]['items']))
```

Out[30]:

```
'There are 30 around Ecco restaurant.'
```

Get relevant part of JSON

In [31]:

```
items = results['response']['groups'][0]['items']
items[0]
```

Out[31]:


```
{'reasons': {'count': 0,
  'items': [{'summary': 'This spot is popular',
    'type': 'general',
    'reasonName': 'globalInteractionReason'}]},
 'venue': {'id': '4af5d65ff964a52091fd21e3',
  'name': 'Korin',
  'location': {'address': '57 Warren St',
   'crossStreet': 'Church St',
   'lat': 40.71482437714839,
   'lng': -74.00940425461492,
   'labeledLatLngs': [{'label': 'display',
     'lat': 40.71482437714839,
     'lng': -74.00940425461492},
    {'label': 'entrance', 'lat': 40.714727, 'lng': -74.009399}],
   'distance': 73,
   'postalCode': '10007',
   'cc': 'US',
   'neighborhood': 'Tribeca',
   'city': 'New York',
   'state': 'NY',
   'country': 'United States',
   'formattedAddress': ['57 Warren St (Church St)',
    'New York, NY 10007',
    'United States']},
  'categories': [{'id': '4bf58dd8d48988d1f8941735',
    'name': 'Furniture / Home Store',
    'pluralName': 'Furniture / Home Stores',
    'shortName': 'Furniture / Home',
    'icon': {'prefix': 'https://ss3.4sqi.net/img/categories_v2/shops/fu
rniture_',
     'suffix': '.png'},
    'primary': True}],
  'photos': {'count': 0, 'groups': []},
  'venuePage': {'id': '33104775'}},
 'referralId': 'e-0-4af5d65ff964a52091fd21e3-0'}
```


Process JSON and convert it to a clean dataframe

In [32]:

```
dataframe = json_normalize(items) # flatten JSON

# filter columns
filtered_columns = ['venue.name', 'venue.categories'] + [col for col in dataframe.c
olumns if col.startswith('venue.location.')] + ['venue.id']
dataframe_filtered = dataframe.loc[:, filtered_columns]

# filter the category for each row
dataframe_filtered['venue.categories'] = dataframe_filtered.apply(get_category_type
, axis=1)

# clean columns
dataframe_filtered.columns = [col.split('.')[-1] for col in dataframe_filtered.colu
mns]

dataframe_filtered.head(10)
```

```
/home/jupyterlab/conda/envs/python/lib/python3.6/site-packages/ipykerne
l_launcher.py:1: FutureWarning: pandas.io.json.json_normalize is deprec
ated, use pandas.json_normalize instead
  """Entry point for launching an IPython kernel.
```

Out[32]:

	name	categories	address	crossStreet	lat	lng	labeledLatLngs	dis
0	Korin	Furniture / Home Store	57 Warren St	Church St	40.714824	-74.009404	[{'label': 'display', 'lat': 40.71482437714839, 'lng': -74.00940425461492}, {'label': 'entrance', 'lat': 40.714727, 'lng': -74.009399}]	
1	Los Tacos No. 1	Taco Place	136 Church St	NaN	40.714267	-74.008756	[{'label': 'display', 'lat': 40.714267, 'lng': -74.008756}]	
2	Takahachi Bakery	Bakery	25 Murray St	at Church St	40.713653	-74.008804	[{'label': 'display', 'lat': 40.713652845301894, 'lng': -74.0088038953017}, {'label': 'entrance', 'lat': 40.713716, 'lng': -74.008443}]	
3	Juice Press	Vegetarian / Vegan Restaurant	83 Murray St	btwn Greenwich St & W Broadway	40.714788	-74.011132	[{'label': 'display', 'lat': 40.71478769908051, 'lng': -74.0111317502157}]	
4	Chambers Street Wines	Wine Shop	148 Chambers St	btwn West Broadway & Hudson St	40.715773	-74.009718	[{'label': 'display', 'lat': 40.715773063928374, 'lng': -74.00971823312332}, {'label': 'entrance', 'lat': 40.715696, 'lng': -74.00988}]	
5	Heyday	Spa	92 Reade St	NaN	40.715726	-74.007767	[{'label': 'display', 'lat': 40.715726, 'lng': -74.007767}, {'label': 'entrance', 'lat': 40.715654, 'lng': -74.00782}]	
6	Takahachi	Sushi Restaurant	145 Duane St	btwn W Broadway & Church St	40.716526	-74.008101	[{'label': 'display', 'lat': 40.71652647412374, 'lng': -74.00810108466207}, {'label': 'entrance', 'lat': 40.716508, 'lng': -74.007989}]	
7	Equinox Tribeca	Gym	54 Murray St	at W Broadway	40.714099	-74.009686	[{'label': 'display', 'lat': 40.71409860726041, 'lng': -74.0096857179283}]	
8	Lekka Burger	Burger Joint	81 Warren St	btw Greenwich & West Broadway	40.715246	-74.010559	[{'label': 'display', 'lat': 40.715246, 'lng': -74.010559}]	

	name	categories	address	crossStreet	lat	lng	labeledLatLngs	dis
9	Weather Up	Cocktail Bar	159 Duane St	btwn Hudson St. & W Broadway	40.716741	-74.008666	[{'label': 'display', 'lat': 40.71674084163369, 'lng': -74.0086664438893}, {'label': 'entrance', 'lat': 40.71685, 'lng': -74.008729}]	

Let's visualize these items on the map around our location

In [33]:

```
venues_map = folium.Map(location=[latitude, longitude], zoom_start=15) # generate m
ap centred around Ecco


# add Ecco as a red circle mark
folium.features.CircleMarker(
    [latitude, longitude],
    radius=10,
    popup='Ecco',
    fill=True,
    color='red',
    fill_color='red',
    fill_opacity=0.6
    ).add_to(venues_map)


# add popular spots to the map as blue circle markers
for lat, lng, label in zip(dataframe_filtered.lat, dataframe_filtered.lng, datafram
e_filtered.categories):
    folium.features.CircleMarker(
        [lat, lng],
        radius=5,
        popup=label,
        fill=True,
        color='blue',
        fill_color='blue',
        fill_opacity=0.6
        ).add_to(venues_map)

# display map
venues_map
```

Out[33]:

Make this Notebook Trusted to load map: File -> Trust Notebook

5. Explore Trending Venues

```
https://api.foursquare.com/v2/venues/ trending ?
client_id= CLIENT_ID &client_secret= CLIENT_SECRET &ll= LATITUDE , LONGITUDE &v=
```

Now, instead of simply exploring the area around Ecco, you are interested in knowing the venues that are trending at the time you are done with your lunch, meaning the places with the highest foot traffic. So let's do that and get the trending venues around Ecco.

In [34]:

```
# define URL
url = 'https://api.foursquare.com/v2/venues/trending?client_id={}&client_secret={}&ll={},{}&v={}'.format(CLIENT_ID, CLIENT_SECRET, latitude, longitude, VERSION)

# send GET request and get trending venues
results = requests.get(url).json()
results
```

Out[34]:

```
{'meta': {'code': 200, 'requestId': '5ff6b11145eede071e4e27c0'},
 'response': {'venues': []}}
```

Check if any venues are trending at this time

In [35]:

```
if len(results['response']['venues']) == 0:
    trending_venues_df = 'No trending venues are available at the moment!'

else:
    trending_venues = results['response']['venues']
    trending_venues_df = json_normalize(trending_venues)

    # filter columns
    columns_filtered = ['name', 'categories'] + ['location.distance', 'location.city', 'location.postalCode', 'location.state', 'location.country', 'location.lat', 'location.lng']
    trending_venues_df = trending_venues_df.loc[:, columns_filtered]

    # filter the category for each row
    trending_venues_df['categories'] = trending_venues_df.apply(get_category_type, axis=1)
```

In [36]:

```
# display trending venues
trending_venues_df
```

Out[36]:

```
'No trending venues are available at the moment!'
```

Now, depending on when you run the above code, you might get different venues since the venues with the highest foot traffic are fetched live.

Visualize trending venues

In [37]:

```
if len(results['response']['venues']) == 0:
    venues_map = 'Cannot generate visual as no trending venues are available at the moment!'

else:
    venues_map = folium.Map(location=[latitude, longitude], zoom_start=15) # generate map centred around Ecco


    # add Ecco as a red circle mark
    folium.features.CircleMarker(
        [latitude, longitude],
        radius=10,
        popup='Ecco',
        fill=True,
        color='red',
        fill_color='red',
        fill_opacity=0.6
    ).add_to(venues_map)


    # add the trending venues as blue circle markers
    for lat, lng, label in zip(trending_venues_df['location.lat'], trending_venues_df['location.lng'], trending_venues_df['name']):
        folium.features.CircleMarker(
            [lat, lng],
            radius=5,
            poup=label,
            fill=True,
            color='blue',
            fill_color='blue',
            fill_opacity=0.6
        ).add_to(venues_map)
```

In [38]:

```
# display map
venues_map
```

Out[38]:

```
'Cannot generate visual as no trending venues are available at the mome
nt!'
```

Thank you for completing this lab!

This notebook was created by Alex Aklson (https://www.linkedin.com/in/aklson/). I hope you found this lab interesting and educational. Feel free to contact me if you have any questions!

This notebook is part of a course on **Coursera** called *Applied Data Science Capstone*. If you accessed this notebook outside the course, you can take this course online by clicking here (http://cocl.us/DP0701EN_Coursera_Week2_LAB1).
