DP0701EN-2-2-1-Foursquare-API-py-v1.0

June 26, 2020

Learning FourSquare API with Python

0.1 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.

0.2 Table of Contents

0.2.1 Import necessary Libraries

[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
    ---------------------------|-----------------
    ca-certificates-2020.6.20  |       hecda079_0         145 KB  conda-forge
    certifi-2020.6.20          |   py36h9f0ad1d_0         151 KB  conda-forge
    geographiclib-1.50         |             py_0          34 KB  conda-forge
    geopy-1.22.0               |     pyh9f0ad1d_0          63 KB  conda-forge
    ------------------------------------------------------------
                                           Total:         393 KB

The following NEW packages will be INSTALLED:

  geographiclib      conda-forge/noarch::geographiclib-1.50-py_0
  geopy              conda-forge/noarch::geopy-1.22.0-pyh9f0ad1d_0

The following packages will be UPDATED:

  ca-certificates                      2020.4.5.2-hecda079_0 -->
2020.6.20-hecda079_0
  certifi                          2020.4.5.2-py36h9f0ad1d_0 -->
2020.6.20-py36h9f0ad1d_0



Downloading and Extracting Packages
certifi-2020.6.20    | 151 KB    | ##################################### | 100%
geopy-1.22.0         | 63 KB     | ##################################### | 100%
ca-certificates-2020 | 145 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 flexible
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-0.4.1               |             py_0          26 KB  conda-forge
    brotlipy-0.7.0             |py36h8c4c3a4_1000         346 KB  conda-forge
    chardet-3.0.4              |py36h9f0ad1d_1006         188 KB  conda-forge
    cryptography-2.9.2         |   py36h45558ae_0         613 KB  conda-forge
    folium-0.5.0               |             py_0          45 KB  conda-forge
    pandas-1.0.5               |   py36h830a2c2_0        10.1 MB  conda-forge
    pysocks-1.7.1              |   py36h9f0ad1d_1          27 KB  conda-forge
    requests-2.24.0            |     pyh9f0ad1d_0          47 KB  conda-forge
    toolz-0.10.0               |             py_0          46 KB  conda-forge
    vincent-0.4.4              |             py_1          28 KB  conda-forge
    ------------------------------------------------------------
                                           Total:        12.0 MB

The following NEW packages will be INSTALLED:

  altair             conda-forge/noarch::altair-4.1.0-py_1
  attrs              conda-forge/noarch::attrs-19.3.0-py_0
  branca             conda-forge/noarch::branca-0.4.1-py_0
  brotlipy           conda-forge/linux-64::brotlipy-0.7.0-py36h8c4c3a4_1000
  chardet            conda-forge/linux-64::chardet-3.0.4-py36h9f0ad1d_1006
  cryptography       conda-forge/linux-64::cryptography-2.9.2-py36h45558ae_0
  entrypoints        conda-forge/linux-64::entrypoints-0.3-py36h9f0ad1d_1001
  folium             conda-forge/noarch::folium-0.5.0-py_0
  idna               conda-forge/noarch::idna-2.9-py_1
  importlib_metadata conda-forge/noarch::importlib_metadata-1.6.1-0
  jinja2             conda-forge/noarch::jinja2-2.11.2-pyh9f0ad1d_0
  jsonschema         conda-forge/linux-64::jsonschema-3.2.0-py36h9f0ad1d_1
  markupsafe         conda-forge/linux-64::markupsafe-1.1.1-py36h8c4c3a4_1
  pandas             conda-forge/linux-64::pandas-1.0.5-py36h830a2c2_0
  pyopenssl          conda-forge/noarch::pyopenssl-19.1.0-py_1
  pyrsistent         conda-forge/linux-64::pyrsistent-0.16.0-py36h8c4c3a4_0
  pysocks            conda-forge/linux-64::pysocks-1.7.1-py36h9f0ad1d_1
  pytz               conda-forge/noarch::pytz-2020.1-pyh9f0ad1d_0
  requests           conda-forge/noarch::requests-2.24.0-pyh9f0ad1d_0
  toolz              conda-forge/noarch::toolz-0.10.0-py_0
  urllib3            conda-forge/noarch::urllib3-1.25.9-py_0
  vincent            conda-forge/noarch::vincent-0.4.4-py_1



Downloading and Extracting Packages
pysocks-1.7.1        | 27 KB     | ##################################### | 100%
toolz-0.10.0         | 46 KB     | ##################################### | 100%
chardet-3.0.4        | 188 KB    | ##################################### | 100%
folium-0.5.0         | 45 KB     | ##################################### | 100%
branca-0.4.1         | 26 KB     | ##################################### | 100%
cryptography-2.9.2   | 613 KB    | ##################################### | 100%
brotlipy-0.7.0       | 346 KB    | ##################################### | 100%
altair-4.1.0         | 614 KB    | ##################################### | 100%
requests-2.24.0      | 47 KB     | ##################################### | 100%
pandas-1.0.5         | 10.1 MB   | ##################################### | 100%
vincent-0.4.4        | 28 KB     | ##################################### | 100%
Preparing transaction: done
Verifying transaction: done
Executing transaction: done
Folium installed
Libraries imported.
```

0.2.2 Define Foursquare Credentials and Version

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

```
[2]: CLIENT_ID = "LBFXES5UC5AYF5UNJYSO1LSLK5J2KASJ5NL54O4CHKBRKBKF" # your␣
      ↪Foursquare ID
     CLIENT_SECRET = "YX5CCUCAKWKPPCXGRG1MVOTP3CW3EJ2WBYAV5HNP35J3TCSX" # your␣
      ↪Foursquare Secret
     VERSION = '20180604'
     LIMIT = 30
     print('Your credentails:')
     print('CLIENT_ID: ' + CLIENT_ID)
     print('CLIENT_SECRET:' + CLIENT_SECRET)
```

```
Your credentails:
CLIENT_ID: LBFXES5UC5AYF5UNJYSO1LSLK5J2KASJ5NL54O4CHKBRKBKF
```

```
CLIENT_SECRET:YX5CCUCAKWKPPCXGRG1MVOTP3CW3EJ2WBYAV5HNP35J3TCSX
```

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.

[3]:
```python
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)
```

```
40.7151482 -74.0156573
```

0.3 1. Search for a specific venue category

https://api.foursquare.com/v2/venues/search?client_id=**CLIENT_ID**&client_secret=**CLIENT**

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.

[4]:
```python
search_query = 'Italian'
radius = 500
print(search_query + ' .... OK!')
```

```
Italian … OK!
```

Define the corresponding URL

[5]:
```python
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
```

[5]:
```
'https://api.foursquare.com/v2/venues/search?client_id=LBFXES5UC5AYF5UNJYSO1LSLK
5J2KASJ5NL54O4CHKBRKBKF&client_secret=YX5CCUCAKWKPPCXGRG1MVOTP3CW3EJ2WBYAV5HNP35
J3TCSX&ll=40.7151482,-74.0156573&v=20180604&query=Italian&radius=500&limit=30'
```

Send the GET Request and examine the results

[6]:
```python
results = requests.get(url).json()
results
```

```
[6]: {'meta': {'code': 200, 'requestId': '5ef5c98d1ec6724d86d1bb9e'},
      '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': 77,
          '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/pizza_',
            'suffix': '.png'},
           'primary': True}],
         'referralId': 'v-1593166300',
         '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': 499,
          '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/default_',
       'suffix': '.png'},
      'primary': True}],
    'referralId': 'v-1593166300',
    'hasPerk': False}]}}
```

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

```
[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/ipykernel_launcher.py:5: FutureWarning: pandas.io.json.json_normalize
is deprecated, use pandas.json_normalize instead
  """
```

```
[7]:                         id                             name  \
 0  4fa862b3e4b0ebff2f749f06        Harry's Italian Pizza Bar
 1  4f3232e219836c91c7bfde94  Conca Cucina Italian Restaurant

                                           categories    referralId  hasPerk  \
 0  [{'id': '4bf58dd8d48988d1ca941735', 'name': 'P…  v-1593166300    False
 1  [{'id': '4d4b7105d754a06374d81259', 'name': 'F…  v-1593166300    False

   location.address  location.lat  location.lng  \
 0    225 Murray St     40.715218    -74.014739
 1    63 W Broadway     40.714484    -74.009806

                              location.labeledLatLngs  location.distance  \
 0  [{'label': 'display', 'lat': 40.71521779064671…                 77
 1  [{'label': 'display', 'lat': 40.71448400000000…                499

   location.postalCode location.cc location.city location.state  \
 0               10282          US      New York             NY
 1               10007          US      New York             NY

   location.country                         location.formattedAddress
 0    United States  [225 Murray St, New York, NY 10282, United Sta…
 1    United States  [63 W Broadway, New York, NY 10007, United Sta…
```

Define information of interest and filter dataframe

```
[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
```

```
[8]:                              name   categories         address        lat  \
     0        Harry's Italian Pizza Bar  Pizza Place   225 Murray St  40.715218
     1  Conca Cucina Italian Restaurant         Food  63 W Broadway  40.714484

              lng                                      labeledLatLngs  distance  \
     0 -74.014739  [{'label': 'display', 'lat': 40.71521779064671…        77
     1 -74.009806  [{'label': 'display', 'lat': 40.71448400000000…       499

       postalCode  cc      city state        country  \
     0      10282  US  New York    NY  United States
     1      10007  US  New York    NY  United States

                                         formattedAddress                        id
     0  [225 Murray St, New York, NY 10282, United Sta…  4fa862b3e4b0ebff2f749f06
     1  [63 W Broadway, New York, NY 10007, United Sta…  4f3232e219836c91c7bfde94
```

Let's visualize the Italian restaurants that are nearby

```
[9]: dataframe_filtered.name
```

```
[9]: 0          Harry's Italian Pizza Bar
     1    Conca Cucina Italian Restaurant
     Name: name, dtype: object
```

```
[11]: venues_map = folium.Map(location=[latitude, longitude], zoom_start=13) #␣
       ↪generate map 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,␣
       ↪dataframe_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
```

```
[11]: <folium.folium.Map at 0x7f9442ef8a58>
```

0.4 2. Explore a Given Venue

https://api.foursquare.com/v2/venues/**VENUE_ID**?client_id=**CLIENT_ID**&client_secret=**C**

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

```
[12]: 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
```

```
[12]: 'https://api.foursquare.com/v2/venues/4fa862b3e4b0ebff2f749f06?client_id=LBFXES5
      UC5AYF5UNJYSO1LSLK5J2KASJ5NL5404CHKBRKBKF&client_secret=YX5CCUCAKWKPPCXGRG1MVOTP
      3CW3EJ2WBYAV5HNP35J3TCSX&v=20180604'
```

Send GET request for result

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

```
dict_keys(['id', 'name', 'contact', 'location', 'canonicalUrl', 'categories',
'verified', 'stats', 'url', 'price', 'hasMenu', 'likes', 'dislike', 'ok',
'rating', 'ratingColor', 'ratingSignals', 'menu', 'allowMenuUrlEdit',
'beenHere', 'specials', 'photos', 'reasons', 'hereNow', 'createdAt', 'tips',
'shortUrl', 'timeZone', 'listed', 'hours', 'popular', 'seasonalHours',
'defaultHours', 'pageUpdates', 'inbox', 'attributes', 'bestPhoto', 'colors'])
```

```
[13]: {'id': '4fa862b3e4b0ebff2f749f06',
       'name': "Harry's Italian Pizza Bar",
       'contact': {'phone': '2126081007', 'formattedPhone': '(212) 608-1007'},
       '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/4fa862b3e4b0ebff2f749f06',
       'categories': [{'id': '4bf58dd8d48988d1ca941735',
         'name': 'Pizza Place',
         'pluralName': 'Pizza Places',
         'shortName': 'Pizza',
         'icon': {'prefix': 'https://ss3.4sqi.net/img/categories_v2/food/pizza_',
          'suffix': '.png'},
         'primary': True},
        {'id': '4bf58dd8d48988d110941735',
         'name': 'Italian Restaurant',
         'pluralName': 'Italian Restaurants',
         'shortName': 'Italian',
   'icon': {'prefix': 'https://ss3.4sqi.net/img/categories_v2/food/italian_',
    'suffix': '.png'}}],
 'verified': False,
 'stats': {'tipCount': 57},
 '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': 212,
 'menu': {'type': 'Menu',
  'label': 'Menu',
  'anchor': 'View Menu',
  'url': 'https://foursquare.com/v/harrys-italian-pizza-
bar/4fa862b3e4b0ebff2f749f06/menu',
  'mobileUrl': 'https://foursquare.com/v/4fa862b3e4b0ebff2f749f06/device_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': {'id': '13676709',
       'firstName': 'Leony',
       'lastName': 'N',
       'photo': {'prefix': 'https://fastly.4sqi.net/img/user/',
        'suffix': '/T0ANFNGNMCHUDEUE.jpg'}},
      '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': 57,
  'groups': [{'type': 'others',
    'name': 'All tips',
    'count': 57,
    'items': [{'id': '53d27909498e0523841340b6',
      'createdAt': 1406302473,
      'text': "Harry's Italian Pizza bar is known for it's amazing pizza, but
did you know that the brunches here are amazing too? Try the Nutella French
toast and we know you'll be sold.",
      'type': 'user',
      'canonicalUrl': 'https://foursquare.com/item/53d27909498e0523841340b6',
      'lang': 'en',
      'likes': {'count': 4,
       'groups': [{'type': 'others',
         'count': 4,
         'items': [{'id': '369426',
           'firstName': 'P.',
           'lastName': 'M',
           'photo': {'prefix': 'https://fastly.4sqi.net/img/user/',
            'suffix': '/JPQYUWJKUTOH2OO4.jpg'}},
          {'id': '87587879',
           'firstName': 'Diane',
           'lastName': 'D',
           'photo': {'prefix': 'https://fastly.4sqi.net/img/user/',
            'suffix': '/87587879-ESLRSZLQ2CBE2P4W.jpg'}},
          {'id': '87591341',
           'firstName': 'Tim',
           'lastName': 'S',
           'photo': {'prefix': 'https://fastly.4sqi.net/img/user/',
            'suffix': '/-Z4YK4VKEOJSVXIY1.jpg'}},
          {'id': '87473404',
           'firstName': 'TenantKing.com',
           'photo': {'prefix': 'https://fastly.4sqi.net/img/user/',
            'suffix': '/87473404-HI5DTBTKOHX401CA.png'},
           'type': 'page'}]}],
       'summary': '4 likes'},
      'logView': True,
      'agreeCount': 4,
      'disagreeCount': 0,
      'todo': {'count': 0},
      'user': {'id': '87473404',
       'firstName': 'TenantKing.com',
       'photo': {'prefix': 'https://fastly.4sqi.net/img/user/',
        'suffix': '/87473404-HI5DTBTK0HX401CA.png'},
       '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': {'id': '24592223',
       'firstName': 'Becca',
       'lastName': 'M',
       'photo': {'prefix': 'https://fastly.4sqi.net/img/user/',
        'suffix': '/24592223-RAW2UYM0GIB1U40K.jpg'}},
      'editable': False,
      'public': True,
      'collaborative': False,
      'url': '/becca_mcarthur/list/manhattan-haunts',
      'canonicalUrl': 'https://foursquare.com/becca_mcarthur/list/manhattan-
haunts',
      'createdAt': 1336094672,
      'updatedAt': 1380845377,
      'photo': {'id': '4e8cc9461081e3b3544e12e5',
       'createdAt': 1317849414,
       'prefix': 'https://fastly.4sqi.net/img/general/',
       'suffix': '/0NLVU2HC1JF4DXIMKWUFW3QBUT31DC11EFNYYHMJG3NDWAPS.jpg',
       'width': 492,
       'height': 330,
       'user': {'id': '742542',
        'firstName': 'Time Out New York',
        'photo': {'prefix': 'https://fastly.4sqi.net/img/user/',
         'suffix': '/XXHKCBSQHBORZNSR.jpg'},
        '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 wasteland of
Downtown NYC aka FiDi, along with Tribeca & Battery Park City.',
      'type': 'others',
      'user': {'id': '12113441',
        'firstName': 'Kino',
        'photo': {'prefix': 'https://fastly.4sqi.net/img/user/',
         'suffix': '/12113441-K5HTHFLU2MUCM0CM.jpg'}},
       'editable': False,
       'public': True,
       'collaborative': False,
       'url': '/kinosfault/list/usa-nyc-man-fidi',
       'canonicalUrl': 'https://foursquare.com/kinosfault/list/usa-nyc-man-fidi',
       'createdAt': 1336836475,
       'updatedAt': 1556754919,
       'photo': {'id': '55984992498e13ba75e353bb',
        'createdAt': 1436043666,
        'prefix': 'https://fastly.4sqi.net/img/general/',
        'suffix': '/12113441_iOa6Uh-Xi8bhj2-gpzkkw8MKiAIs7RmOcz_RM7m8ink.jpg',
        'width': 540,
        'height': 960,
        'user': {'id': '12113441',
         'firstName': 'Kino',
         'photo': {'prefix': 'https://fastly.4sqi.net/img/user/',
          'suffix': '/12113441-K5HTHFLU2MUCM0CM.jpg'}},
        'visibility': 'public'},
       'followers': {'count': 20},
       'listItems': {'count': 273,
        'items': [{'id': 'v4fa862b3e4b0ebff2f749f06',
          'createdAt': 1373909433}]}},
      {'id': '4fddeff0e4b0e078037ac0d3',
       'name': 'NYC Resturants',
       'description': '',
       'type': 'others',
       'user': {'id': '21563126',
        'firstName': 'Richard',
        'lastName': 'R',
        'photo': {'prefix': 'https://fastly.4sqi.net/img/user/',
         'suffix': '/21563126_v05J1KPw_SVj6Ehq9g8B9jeAGjFUMsU5QGl-
NZ8inUQ7pKQm5bKplW37EmR7jS2A7GYPBBAtl.jpg'}},
       'editable': False,
       'public': True,
       'collaborative': True,
       'url': '/rickr7/list/nyc-resturants',
       'canonicalUrl': 'https://foursquare.com/rickr7/list/nyc-resturants',
       'createdAt': 1339944944,
       'updatedAt': 1591664261,
       'photo': {'id': '5072dd13e4b09145cdf782d1',
        'createdAt': 1349704979,
        'prefix': 'https://fastly.4sqi.net/img/general/',
        'suffix': '/208205_fGh2OuAZ9qJ4agbAA5wMVNOSIm9kNUlRtNwj1N-adqg.jpg',
        'width': 800,
        'height': 800,
        'user': {'id': '208205',
         'firstName': 'Thalia',
         'lastName': 'K',
         'photo': {'prefix': 'https://fastly.4sqi.net/img/user/',
          'suffix': '/SNOOLCAW2AG04ZKD.jpg'}},
        'visibility': 'public'},
       'followers': {'count': 12},
       'listItems': {'count': 193,
        'items': [{'id': 'v4fa862b3e4b0ebff2f749f06',
          'createdAt': 1581655865}]}},
      {'id': '5266c68a498e7c667807fe09',
       'name': 'Foodie Love in NY - 02',
       'description': '',
       'type': 'others',
       'user': {'id': '547977',
        'firstName': 'WiLL',
        'photo': {'prefix': 'https://fastly.4sqi.net/img/user/',
         'suffix': '/-Q5NYGDMFDMOITQRR.jpg'}},
       'editable': False,
       'public': True,
       'collaborative': False,
       'url': '/sweetiewill/list/foodie-love-in-ny--02',
       'canonicalUrl': 'https://foursquare.com/sweetiewill/list/foodie-love-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-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-Thu',
    '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': '$$',
    'count': 1,
    'items': [{'displayName': 'Price', 'displayValue': '$$', 'priceTier': 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': 'Yes'}]},
    {'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'}]},
    {'type': 'diningOptions',
     'name': 'Dining Options',
     'summary': 'Delivery',
     'count': 5,
     'items': [{'displayName': 'Delivery', 'displayValue': 'Delivery'}]}]},
  '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', 'value': -1},
   'algoVersion': 3}}
```

0.4.2 B. Get the venue's overall rating

[14]:
```python
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.

[15]:
```
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.

[16]:
```
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.

0.4.3 C. Get the number of tips

[18]:
```
result['response']['venue']['tips']['count']
```

[18]:
```
19
```

0.4.4 D. Get the venue's tips

https://api.foursquare.com/v2/venues/**VENUE_ID**/tips?client_id=**CLIENT_ID**&client_secr

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

[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
```

[17]:
```
{'meta': {'code': 200, 'requestId': '5ef5cca647efc372b9f9478c'},
 'response': {'tips': {'count': 19,
   'items': [{'id': '5ab1cb46c9a517174651d3fe',
     'createdAt': 1521601350,
     'text': 'A+ Italian food! Trust me on this: my mom's side of the family is
100% Italian. I was born and bred to know good pasta when I see it, and Ecco is
one of my all-time NYC favorites',
     'type': 'user',
     'canonicalUrl': 'https://foursquare.com/item/5ab1cb46c9a517174651d3fe',
     'lang': 'en',
     'likes': {'count': 0, 'groups': []},
     'logView': True,
     'agreeCount': 4,
     'disagreeCount': 0,
     'todo': {'count': 0},
     'user': {'id': '484542633',
      'firstName': 'Nick',
      'lastName': 'E',
      'photo': {'prefix': 'https://fastly.4sqi.net/img/user/',
       'suffix': '/484542633_unymNUmw_FdPs3GjXHujmHcYnN4hf8kEPADlOZuIrdcdm97VX3t
FqL7fFNMNA_8Gl9NlU1GYg.jpg'}},
     'authorInteractionType': 'liked'}]}}}
```

Get tips and list of associated features

[18]:
```
tips = results['response']['tips']['items']

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

[18]:
```
dict_keys(['id', 'createdAt', 'text', 'type', 'canonicalUrl', 'lang', 'likes',
'logView', 'agreeCount', 'disagreeCount', 'todo', 'user',
'authorInteractionType'])
```

Format column width and display all tips

[21]:
```
pd.set_option('display.max_colwidth', -1)

tips_df = json_normalize(tips) # json normalize tips

# columns to keep
filtered_columns = ['text', 'agreeCount', 'disagreeCount', 'id', 'user.
 ↪firstName', 'user.lastName', 'user.id']
tips_filtered = tips_df.loc[:, filtered_columns]
```

```
# display tips
tips_filtered
```

```
/home/jupyterlab/conda/envs/python/lib/python3.6/site-
packages/ipykernel_launcher.py:1: FutureWarning: Passing a negative integer is
deprecated in version 1.0 and will not be supported in future version. Instead,
use None to not limit the column width.
  """Entry point for launching an IPython kernel.
/home/jupyterlab/conda/envs/python/lib/python3.6/site-
packages/ipykernel_launcher.py:3: FutureWarning: pandas.io.json.json_normalize
is deprecated, use pandas.json_normalize instead
  This is separate from the ipykernel package so we can avoid doing imports
until
```

[21]:
```
                 text  \
0  A+ Italian food! Trust me on this: my mom’s side of the family is 100%
Italian. I was born and bred to know good pasta when I see it, and Ecco is one
of my all-time NYC favorites

   agreeCount  disagreeCount                        id user.firstName  \
0  4           0              5ab1cb46c9a517174651d3fe  Nick

  user.lastName    user.id
0  E              484542633
```

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.

0.5 3. Search a Foursquare User

`https://api.foursquare.com/v2/users/`**USER_ID**`?client_id=`**CLIENT_ID**`&client_secret=`**CLIE**

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

[26]:
```
user_id = '484542633' # user ID with most agree counts and complete profile

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

# send GET request
results = requests.get(url).json()
user_data = results['response']['user']

# display features associated with user
user_data.keys()
```

[26]:
```
dict_keys([])
```

[23]:
```
print('First Name: ' + user_data['firstName'])
print('Last Name: ' + user_data['lastName'])
print('Home City: ' + user_data['homeCity'])
```

```
    ␣
↪---------------------------------------------------------------------------

    NameError                                 Traceback (most recent call␣
↪last)

    <ipython-input-23-6c3d173b902c> in <module>
----> 1 print('First Name: ' + user_data['firstName'])
      2 print('Last Name: ' + user_data['lastName'])
      3 print('Home City: ' + user_data['homeCity'])


    NameError: name 'user_data' is not defined
```

How many tips has this user submitted?

[ ]:
```
user_data['tips']
```

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

0.5.2 Get User's tips

[ ]:
```
# 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
```

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

```
[ ]: tip_id = '5ab5575d73fe2516ad8f363b' # tip id

     # define URL
     url = 'http://api.foursquare.com/v2/tips/{}?client_id={}&client_secret={}&v={}'.
      ↪format(tip_id, CLIENT_ID, CLIENT_SECRET, VERSION)

     # send GET Request and examine results
     result = requests.get(url).json()
     print(result['response']['tip']['venue']['name'])
     print(result['response']['tip']['venue']['location'])
```

0.5.3 Get User's friends

```
[ ]: user_friends = json_normalize(user_data['friends']['groups'][0]['items'])
     user_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.

0.5.4 Retrieve the User's Profile Image

```
[ ]: user_data
```

```
[ ]: # 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_7Tn9fWpndidJsmw2Hof_6T5vJBKCHPLMK5OL-U5ZiJGj51iwBstcpDLYa3Zvhvis.
      ↪jpg')
```

0.6 4. Explore a location

`https://api.foursquare.com/v2/venues/explore?client_id=`**CLIENT_ID**`&client_secret=`**CLIEN**

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.

```
[ ]: latitude = 40.715337
     longitude = -74.008848
```

Define URL

```
[ ]: 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
```

Send GET request and examine results

```
[ ]: import requests
```

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

Get relevant part of JSON

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

Process JSON and convert it to a clean dataframe

```
[ ]: dataframe = json_normalize(items) # flatten JSON

     # filter columns
     filtered_columns = ['venue.name', 'venue.categories'] + [col for col in␣
      ↪dataframe.columns 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.
      ↪columns]

     dataframe_filtered.head(10)
```

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

```
[ ]: 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 popular spots to the map as blue circle markers
for lat, lng, label in zip(dataframe_filtered.lat, dataframe_filtered.lng,␣
 ↪dataframe_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
```

0.7 5. Explore Trending Venues

https://api.foursquare.com/v2/venues/**trending**?client_id=**CLIENT_ID**&client_secret=**CLIEN**

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.

[ ]:
```
# 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
```

0.7.1 Check if any venues are trending at this time

[ ]:
```
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)
```

[ ]:
```
# display trending venues
trending_venues_df
```

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.

0.7.2 Visualize trending venues

[ ]:
```
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)
```

[ ]:
```
# display map
venues_map
```

0.7.3 Thank you for completing this lab!

This notebook was created by Alex 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.