(https://cognitiveclass.ai)

Learning FourSquare API with Python

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 [ ]:
```
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 latitud

# 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.')
```

```
Solving environment: -
```

Define Foursquare Credentials and Version

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

In [2]:
```
CLIENT_ID = 'MYP5LS452CUPIMYZU1XWFTI3BN3J2COWMU3UP5BADYU1J55G' # your Foursquare
CLIENT_SECRET = 'SDG1AUOYAOQ0QB0KDFUD0U21CLEMMVD4WRAAB5HJARXXPBRL' # your Foursqu
VERSION = '20180604'
LIMIT = 30
print('Your credentails:')
print('CLIENT_ID: ' + CLIENT_ID)
print('CLIENT_SECRET:' + CLIENT_SECRET)
```

```
Your credentails:
CLIENT_ID: MYP5LS452CUPIMYZU1XWFTI3BN3J2COWMU3UP5BADYU1J55G
CLIENT_SECRET:SDG1AUOYAOQ0QB0KDFUD0U21CLEMMVD4WRAAB5HJARXXPBRL
```

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 , LONGI
```

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={}&
url
```

Out[5]:
```
'https://api.foursquare.com/v2/venues/search?client_id=MYP5LS452CUPIMYZU1XWFTI3
BN3J2COWMU3UP5BADYU1J55G&client_secret=SDG1AUOYAOQ0QB0KDFUD0U21CLEMMVD4WRAAB5HJ
ARXXPBRL&ll=40.7149555,-74.0153365&v=20180604&query=Italian&radius=500&limit=3
0'
```

Send the GET Request and examine the results

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

Out[6]:

```
{'meta': {'code': 200, 'requestId': '5d7ec0a1b9961d002b220e24'},
 '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}],
     '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',
```


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()
```

Out[7]:

	id	name	categories	referralId	hasPerk	locatio
0	4fa862b3e4b0ebff2f749f06	Harry's Italian Pizza Bar	[{'id': '4bf58dd8d48988d1ca941735', 'name': 'P...	v-1568587937	False	225
1	4f3232e219836c91c7bfde94	Conca Cucina Italian Restaurant	[{'id': '4d4b7105d754a06374d81259', 'name': 'F...	v-1568587937	False	63 W
2	3fd66200f964a520f4e41ee3	Ecco	[{'id': '4bf58dd8d48988d110941735', 'name': 'I...	v-1568587937	False	124 Ch

3 rows × 23 columns

Define information of interest and filter dataframe

In [8]:
```
# keep only columns that include venue name, and anything that is associated with
filtered_columns = ['name', 'categories'] + [col for col in dataframe.columns if
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, ax

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

dataframe_filtered
```

Out[8]:

	name	categories	address	lat	lng	labeledLatLngs	distance	postalC
0	Harry's Italian Pizza Bar	Pizza Place	225 Murray St	40.715218	-74.014739	[{'label': 'display', 'lat': 40.71521779064671...	58	10
1	Conca Cucina Italian Restaurant	Food	63 W Broadway	40.714460	-74.010086	[{'label': 'display', 'lat': 40.71446, 'lng': ...	446	10
2	Ecco	Italian Restaurant	124 Chambers St	40.715337	-74.008848	[{'label': 'display', 'lat': 40.71533713859952...	549	10

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

# 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, datafr
    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]:

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={}
url
```

Out[11]:
```
'https://api.foursquare.com/v2/venues/4fa862b3e4b0ebff2f749f06?client_id=MYP5LS
452CUPIMYZU1XWFTI3BN3J2COWMU3UP5BADYU1J55G&client_secret=SDG1AUOYAOQ0QB0KDFUD0U
21CLEMMVD4WRAAB5HJARXXPBRL&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', 'categories',
'verified', 'stats', 'url', 'price', 'hasMenu', 'likes', 'dislike', 'ok', 'ra
ting', 'ratingColor', 'ratingSignals', 'delivery', 'menu', 'allowMenuUrlEdi
t', 'beenHere', 'specials', 'photos', 'reasons', 'hereNow', 'createdAt', 'tip
s', 'shortUrl', 'timeZone', 'listed', 'hours', 'popular', 'pageUpdates', 'inb
ox', 'attributes', 'bestPhoto', 'colors'])
```

B. Get the venue's overall rating

In [13]:
```
try:
    print(result['response']['venue']['rating'])
except:
    print('This venue has not been rated yet.')
```

```
7.0
```

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={}

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={}

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

```
7.9
```

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]:
```
17
```

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= LI
```

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={}

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

Out[17]:

```
{'meta': {'code': 200, 'requestId': '5d7ec1376e46500039c5594a'},
 'response': {'tips': {'count': 17,
   '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': 3,
     'disagreeCount': 0,
     'todo': {'count': 0},
     'user': {'id': '484542633',
      'firstName': 'Nick',
      'lastName': 'El-Tawil',
      'gender': 'male',
      'photo': {'prefix': 'https://fastly.4sqi.net/img/user/',
       'suffix': '/484542633_mK2Yum7T_7Tn9fWpndidJsmw2Hof_6T5vJBKCHPLMK5OL-U5Zi
JGj51iwBstcpDLYa3Zvhvis.jpg'}},
     'authorInteractionType': 'liked'},
    {'id': '5cb7051b180b910039f90625',
     'createdAt': 1555498267,
     'text': 'Excellent food!! Osso bucco special one of the best I ever had! L
obster ravioli with porcini mushroomhomemade Italian cheesecake, tiramisu and n
apoleons...calamari fra diavolo was sautéed not fried',
     'type': 'user',
     'canonicalUrl': 'https://foursquare.com/item/5cb7051b180b910039f90625',
     'lang': 'en',
     'likes': {'count': 0, 'groups': []},
     'logView': True,
     'agreeCount': 1,
     'disagreeCount': 0,
     'todo': {'count': 0},
     'user': {'id': '446298346',
      'firstName': 'Lynn',
      'lastName': 'Bednar-Rando',
      'gender': 'female',
      'photo': {'prefix': 'https://fastly.4sqi.net/img/user/',
       'suffix': '/446298346_fdtqugvP_1WztOWXXDQX92HwIrjzLb6nMQsYiCVlkiTp58UGSF
VBwQ0-5G5AdqxlEWphYWTnK.jpg'}},
     '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', 'authorInteractionTyp
e'])
```

Format column width and display all tips

In [19]:
```
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'
tips_filtered = tips_df.loc[:, filtered_columns]

# display tips
tips_filtered
```

Out[19]:

	text	agreeCount	disagreeCount	id	user.firstName	user
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	3	0	5ab1cb46c9a517174651d3fe	Nick	
1	Excellent food!! Osso bucco special one of the best I ever had! Lobster ravioli with porcini mushroomhomemade Italian cheesecake, tiramisu and napoleons...calamari fra diavolo was sautéed not fried	1	0	5cb7051b180b910039f90625	Lynn	Bed

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

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

url = 'https://api.foursquare.com/v2/users/{}?client_id={}&client_secret={}&v={}'

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

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

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

How many tips has this user submitted?

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

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

Get User's tips

In [ ]:
```
# define tips URL
url = 'https://api.foursquare.com/v2/users/{}/tips?client_id={}&client_secret={}&

# 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

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

# define URL
url = 'http://api.foursquare.com/v2/tips/{}?client_id={}&client_secret={}&v={}'.f

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

Get User's friends

In [ ]:
```
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.

Retrieve the User's Profile Image

In [ ]:
```
user_data
```

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_7Tn9fWpndidJs
```

4. Explore a location

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

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

Define URL

In [ ]:
```
url = 'https://api.foursquare.com/v2/venues/explore?client_id={}&client_secret={}
url
```

Send GET request and examine results

In [ ]:
```
import requests
```

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

Get relevant part of JSON

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

Process JSON and convert it to a clean dataframe

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

# filter columns
filtered_columns = ['venue.name', 'venue.categories'] + [col for col in datafram
dataframe_filtered = dataframe.loc[:, filtered_columns]

# filter the category for each row
dataframe_filtered['venue.categories'] = dataframe_filtered.apply(get_category_ty

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

dataframe_filtered.head(10)
```

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

In [ ]:
```
venues_map = folium.Map(location=[latitude, longitude], zoom_start=15) # generate


# 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, datafr
    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
```

5. Explore Trending Venues

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

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 [ ]:
```
# define URL
url = 'https://api.foursquare.com/v2/venues/trending?client_id={}&client_secret={

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

Check if any venues are trending at this time

In [ ]:
```
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.c
    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
```

In [ ]:
```
# 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.

Visualize trending venues

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

else:
    venues_map = folium.Map(location=[latitude, longitude], zoom_start=15) # gene


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

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).

---

source code are released under the terms of the [MIT License](https://bigdatauniversity.com/mit-license/).