Using the Yelp API - Codealong ¶

Introduction

Now that we've discussed HTTP requests and OAuth, it's time to practice applying those skills to a production level API. In this codealong, we'll take you through the process of signing up for an OAuth token and then using that to make requests to the Yelp API!

Objectives

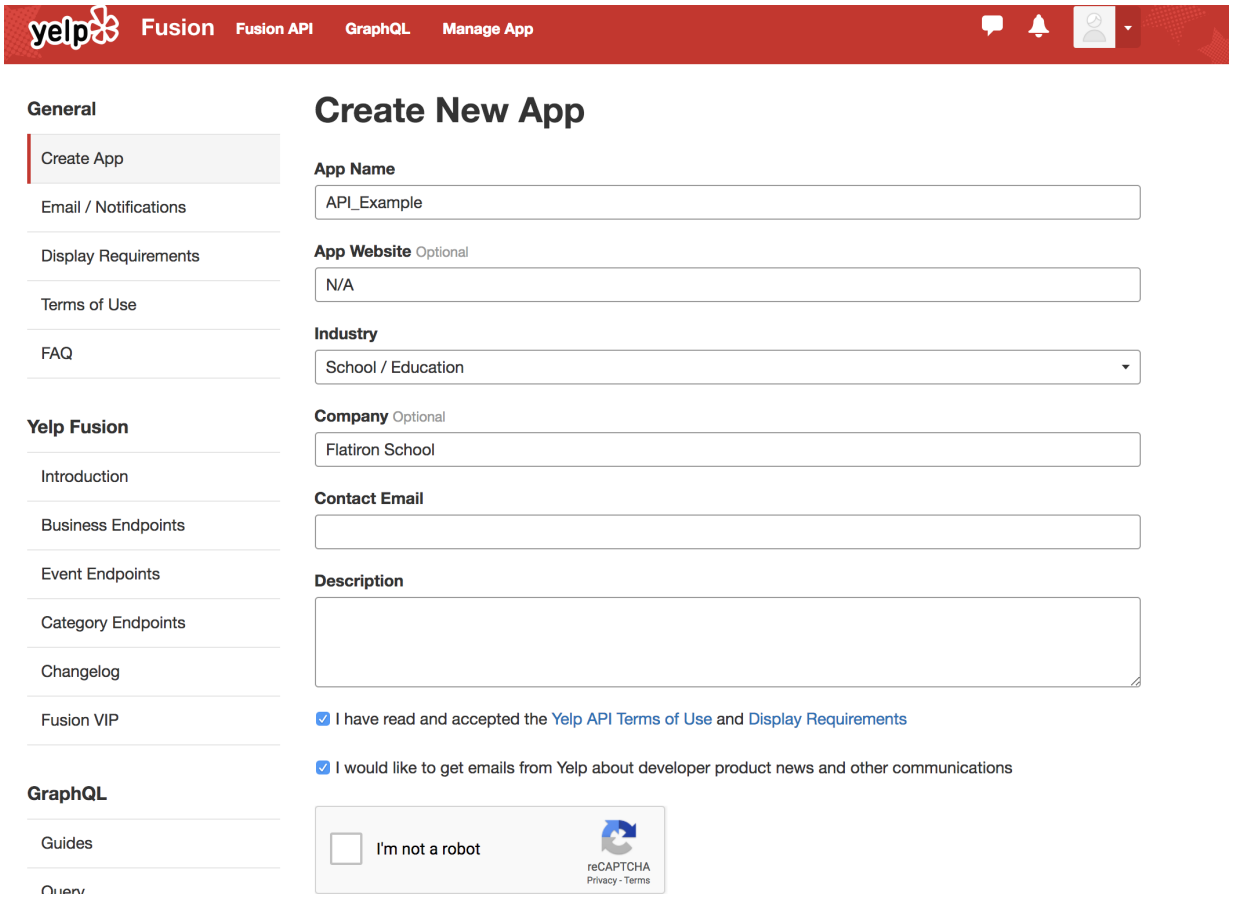
You will be able to:

- Make requests using OAuth
- Use the JSON module to load and parse JSON documents
- Convert JSON to a pandas dataframe

Generating Access Tokens

As discussed, in order to use many APIs, one needs to use OAuth which requires an access token. As such, our first step will be to generate this login information so that we can start making some requests.

With that, let's go grab an access token from an API site and make some API calls! Point your browser over to this yelp page (https://www.yelp.com/developers/v3/manage_app) and start creating an app in order to obtain and API access token:



You can either sign in to an existing Yelp account or create a new one if needed.

On the page you see above, simply fill out some sample information such as "Flatiron Edu API Example" for the app name, or whatever floats your boat. Afterward, you should be presented with an API key that you can use to make requests!

With that, let's set up our authentication tokens so that we can start making some API calls!

Should I publicly share my passwords on Github?

When using an API that requires an API key and password you should **NEVER** hardcode theses values into your main file. When you upload your project onto github it is completely public and vulnerable to attack. Assume that if you put sensitive information publicly on the internet it will be found and abused.

To this end, how can we easily access our API key without opening ourselves up to vulnerabilities?

There are many ways to store sensitive information but we will go with this method.

Move to your home (root) directory:

```
cd ~
```

Now make the `.secret/` directory:

```
mkdir .secret
```

This will create a new folder in your home directory where you can store files for any of the API information you have.

Can you find the file you just made in your terminal? NOTE: dot files won't show up with just `ls` you must use the show all command as well `ls -a`

Move into the newly created `.secret/` folder and create a file using vscode or any text editor to store your yelp API login info.

```
cd .secret/
code yelp_api.json
```

In this file, let's create a dictionary of values representing the client id and API key that looks something like this:

```
{"api_key": "input api key here!"}
```

NOTE: Double quotes are important! You'll copy and paste the `api_key` value that yelp grants you after you create your app.

Ok, so now we have a file in our .secret folder on our home directory. Safe and sound (mostly) from anyone trying to steal our info off github.

Finally, let's get our client id and API key into our jupyter notebook.

If we remember that our file is just a regular JSON file, open the file and pull out the appropriate information from the `~/.secret/yelp_api.json` file.

In [1]:
```
import json

def get_keys(path):
    with open(path) as f:
        return json.load(f)
```

> **Note**: Change the file path below to be your root directory. If you're not sure what your username is, check it with `pwd`
> For example, my current working directory is
> `/Users/matthew.mitchell/Documents/dsc-using-yelp-api-codealong`
> So the line below would become: `keys = get_keys("/Users/matthew.mitchell/.secret/yelp_api.json")`

In [ ]:
```
keys = get_keys("/Users/YOUR_USERNAME_HERE/.secret/yelp_api.json")

api_key = keys['api_key']

#While you may wish to print out these API keys to check that they imported prope
#be sure to clear the output before uploading to Github.
#Again, you don't want your keys stolen!!!
```

An Example Request with OAuth

https://www.yelp.com/developers/documentation/v3/get_started (https://www.yelp.com/developers/documentation/v3/get_started)

In the next lesson, we'll further dissect how to read and translate online documentation like the link here. For now, let's simply look at an example request and dissect it into its constituent parts:

In [19]:
```
import requests
term = 'Mexican'
location = 'Astoria NY'
SEARCH_LIMIT = 10

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

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

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

```
<Response [200]>
<class 'str'>
{"businesses": [{"id": "jeWIYbgBho9vBDhc5S1xvg", "alias": "holy-guacamole-astor
ia", "name": "Holy Guacamole", "image_url": "https://s3-media1.fl.yelpcdn.com/b
photo/8IjT2jd7vKDSOmtdXPI-Zg/o.jpg", "is_closed": false, "url": "https://www.ye
lp.com/biz/holy-guacamole-astoria?adjust_creative=xNHtXRpNa-MXGFJJTHHUvw&utm_ca
mpaign=yelp_api_v3&utm_medium=api_v3_business_search&utm_source=xNHtXRpNa-MXGFJ
JTHHUvw", "review_count": 108, "categories": [{"alias": "mexican", "title": "Me
xican"}, {"alias": "bars", "title": "Bars"}], "rating": 4.0, "coordinates": {"l
atitude": 40.756621, "longitude": -73.929336}, "transactions": ["delivery", "pi
ckup"], "price": "$$", "location": {"address1": "3555 31st St", "address2": "",
"address3": "", "city": "Astoria", "zip_code": "11106", "country": "US", "stat
e": "NY", "display_address": ["3555 31st St", "Astoria, NY 11106"]}, "phone":
"+19178327261", "display_phone": "(917) 832-7261", "distance": 1290.42748751304
48}, {"id": "6AJwsgXr7YwsqneGVAdgzw", "alias": "las-c
```

Breaking Down the Request

As you can see, there are three main parts to our request.

They are:

- The URL
- The header
- The parameters

The URL is fairly straightforward and is simply the base URL as described in the documentation (again more details in the upcoming lesson).

The header is a dictionary of key-value pairs. In this case, we are using a fairly standard header used by many APIs. It has a strict form where 'Authorization' is the key and 'Bearer YourApiKey' is the value.

The parameters are the filters that we wish to pass into the query. These will be embedded into the URL when the request is made to the API. Similar to the header, they form key-value pairs. Valid key parameters by which to structure your queries are described in the API documentation which we'll look at further shortly. A final important note, however, is the need to replace spaces with "+". This is standard to many requests as URLs cannot contain spaces. (Note that the header itself isn't directly embedded into the URL itself and as such, the space between 'Bearer' and YourApiKey is valid.)

The Response

As before, our response object has both a status code, as well as the data itself. With that, let's start with a little data exploration!

In [37]:
```
response.json().keys()
```

Out[37]:
```
dict_keys(['businesses', 'total', 'region'])
```

Now let's go a bit further and start to preview what's stored in each of the values for these keys.

In [39]:
```
for key in response.json().keys():
    print(key)
    value = response.json()[key] #Use standard dictionary formatting
    print(type(value)) #What type is it?
    print('\n\n') #Separate out data
```

```
businesses
<class 'list'>



total
<class 'int'>



region
<class 'dict'>
```

Let's continue to preview these further to get a little better acquainted.

In [45]:
```
response.json()['businesses'][:2]
```

Out[45]:

```
[{'id': 'jeWIYbgBho9vBDhc5S1xvg',
  'alias': 'holy-guacamole-astoria',
  'name': 'Holy Guacamole',
  'image_url': 'https://s3-media1.fl.yelpcdn.com/bphoto/8IjT2jd7vKDSOmtdXPI-Zg/o.jpg',
  'is_closed': False,
  'url': 'https://www.yelp.com/biz/holy-guacamole-astoria?adjust_creative=xNHtXRpNa-MXGFJJTHHUvw&utm_campaign=yelp_api_v3&utm_medium=api_v3_business_search&utm_source=xNHtXRpNa-MXGFJJTHHUvw',
  'review_count': 108,
  'categories': [{'alias': 'mexican', 'title': 'Mexican'},
   {'alias': 'bars', 'title': 'Bars'}],
  'rating': 4.0,
  'coordinates': {'latitude': 40.756621, 'longitude': -73.929336},
  'transactions': ['delivery', 'pickup'],
  'price': '$$',
  'location': {'address1': '3555 31st St',
   'address2': '',
   'address3': '',
   'city': 'Astoria',
   'zip_code': '11106',
   'country': 'US',
   'state': 'NY',
   'display_address': ['3555 31st St', 'Astoria, NY 11106']},
  'phone': '+19178327261',
  'display_phone': '(917) 832-7261',
  'distance': 1290.4274875130448},
 {'id': '6AJwsgXr7YwsqneGVAdgzw',
  'alias': 'las-catrinas-mexican-bar-and-eatery-astoria',
  'name': 'Las Catrinas Mexican Bar & Eatery',
  'image_url': 'https://s3-media3.fl.yelpcdn.com/bphoto/CKRiZUoyTUjs79bWnDEEpg/o.jpg',
  'is_closed': False,
  'url': 'https://www.yelp.com/biz/las-catrinas-mexican-bar-and-eatery-astoria?adjust_creative=xNHtXRpNa-MXGFJJTHHUvw&utm_campaign=yelp_api_v3&utm_medium=api_v3_business_search&utm_source=xNHtXRpNa-MXGFJJTHHUvw',
  'review_count': 163,
  'categories': [{'alias': 'mexican', 'title': 'Mexican'},
   {'alias': 'cocktailbars', 'title': 'Cocktail Bars'}],
  'rating': 4.0,
  'coordinates': {'latitude': 40.7614214682633,
   'longitude': -73.9246649456171},
  'transactions': ['delivery', 'pickup'],
  'price': '$$',
  'location': {'address1': '32-02 Broadway',
   'address2': '',
   'address3': None,
   'city': 'Astoria',
   'zip_code': '11106',
   'country': 'US',
   'state': 'NY',
   'display_address': ['32-02 Broadway', 'Astoria, NY 11106']},
  'phone': '+19177450969',
```

```
 'display_phone': '(917) 745-0969',
 'distance': 642.5257707161409}]
```

In [41]:
```
response.json()['total']
```

Out[41]:
```
638
```

In [42]:
```
response.json()['region']
```

Out[42]:
```
{'center': {'longitude': -73.92219543457031, 'latitude': 40.76688875374591}}
```

As you can see, we're primarily interested in the 'businesses' entry.

Let's go ahead and create a dataframe from that.

In [46]:
```
import pandas as pd

df = pd.DataFrame.from_dict(response.json()['businesses'])
print(len(df)) #Print how many rows
print(df.columns) #Print column names
df.head() #Previews the first five rows.
#You could also write df.head(10) to preview 10 rows or df.tail() to see the bott
```

```
10
Index(['alias', 'categories', 'coordinates', 'display_phone', 'distance', 'id',
       'image_url', 'is_closed', 'location', 'name', 'phone', 'price',
       'rating', 'review_count', 'transactions', 'url'],
      dtype='object')
```

Out[46]:

	alias	categories	coordinates	display_phone	distance	
0	holy-guacamole-astoria	[{'alias': 'mexican', 'title': 'Mexican'}, {'a...	{'latitude': 40.756621, 'longitude': -73.929336}	(917) 832-7261	1290.427488	jeWIYbgBho9vBDhc5S1x
1	las-catrinas-mexican-bar-and-eatery-astoria	[{'alias': 'mexican', 'title': 'Mexican'}, {'a...	{'latitude': 40.7614214682633, 'longitude': -7...	(917) 745-0969	642.525771	6AJwsgXr7YwsqneGVAdgz
2	chela-and-garnacha-astoria	[{'alias': 'mexican', 'title': 'Mexican'}, {'a...	{'latitude': 40.7557171543477, 'longitude': -7...	(917) 832-6876	1316.297661	AUyKmFjpaVLwc3awfUnqg
3	de-mole-astoria-astoria	[{'alias': 'mexican', 'title': 'Mexican'}]	{'latitude': 40.7625999, 'longitude': -73.9129...	(718) 777-1655	917.683267	jzVv_21473lAMYXIhVbu
4	maizal-restaurant-and-tequila-bar-astoria-2	[{'alias': 'mexican', 'title': 'Mexican'}, {'a...	{'latitude': 40.759331, 'longitude': -73.926035}	(718) 406-9431	900.451091	QIsFsiOP3H_NkgeWST7GF

Summary

Congratulations! We've covered a lot here! We took some of your previous knowledge with HTTP requests and OAuth in order to leverage an enterprise API! Then we made some requests to retrieve information that came back as a JSON format. We then transformed this data into a dataframe using the Pandas package. In the next lab, we'll break down how to read API documentation!