Reading API Documentation ¶

Introduction

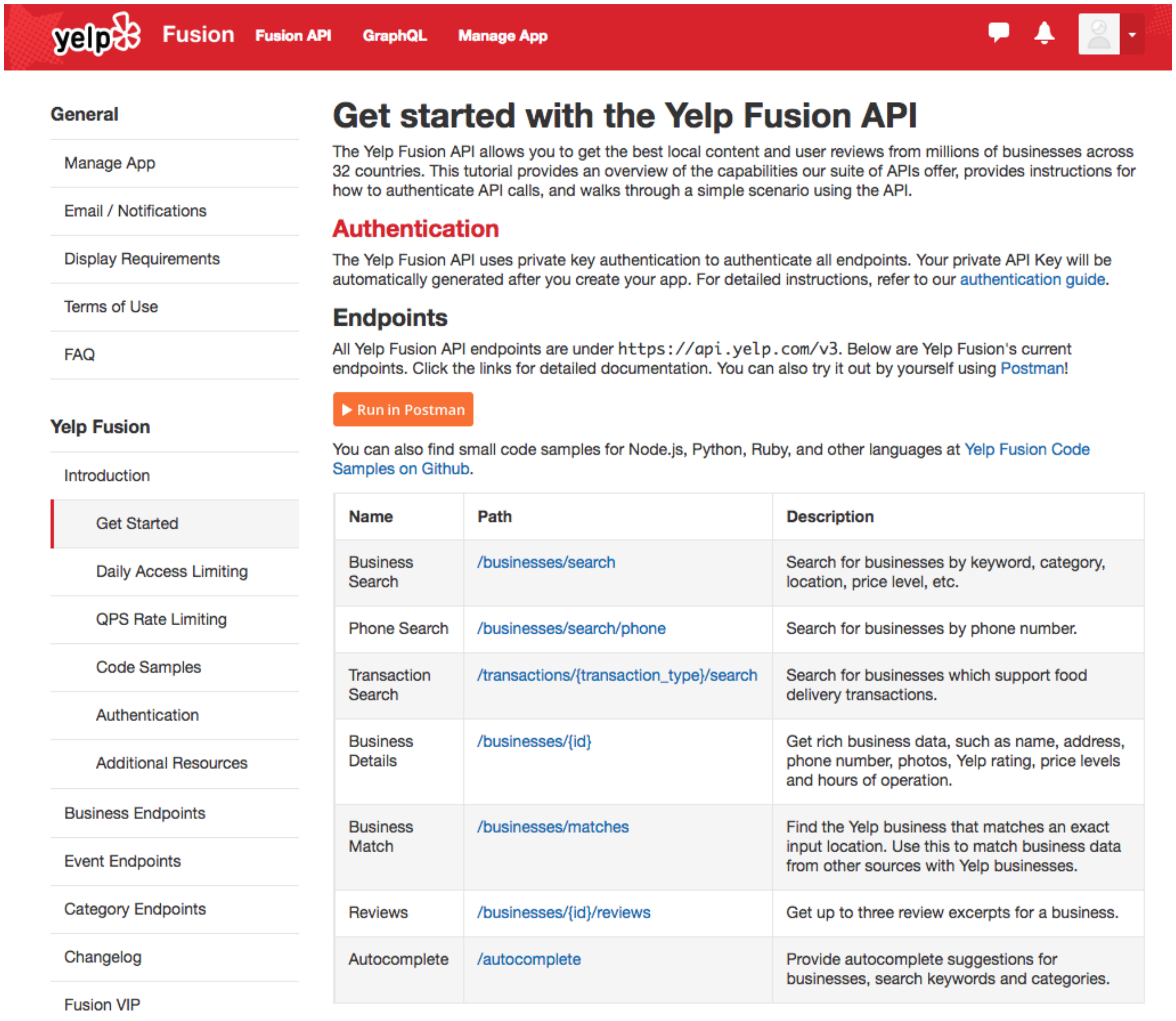
We've now covered an example API request, but how on Earth would you know how to do that on your own? The answer is through documentation! All APIs will have associated documentation, and while there are substantial similarities, all will be different to one degree or another. The best way to get more comfortable is to practice! So with that, let's take a look at the yelp documentation associated with our previous example.

Objectives

You will be able to:

- Interpret API documentation into Python code
- Make requests using OAuth

Start by navigating to https://www.yelp.com/developers/documentation/v3/get_started (https://www.yelp.com/developers/documentation/v3/get_started) and having a look for yourself!



Get started with the Yelp Fusion API

The Yelp Fusion API allows you to get the best local content and user reviews from millions of businesses across 32 countries. This tutorial provides an overview of the capabilities our suite of APIs offer, provides instructions for how to authenticate API calls, and walks through a simple scenario using the API.

Authentication

The Yelp Fusion API uses private key authentication to authenticate all endpoints. Your private API Key will be automatically generated after you create your app. For detailed instructions, refer to our authentication guide.

Endpoints

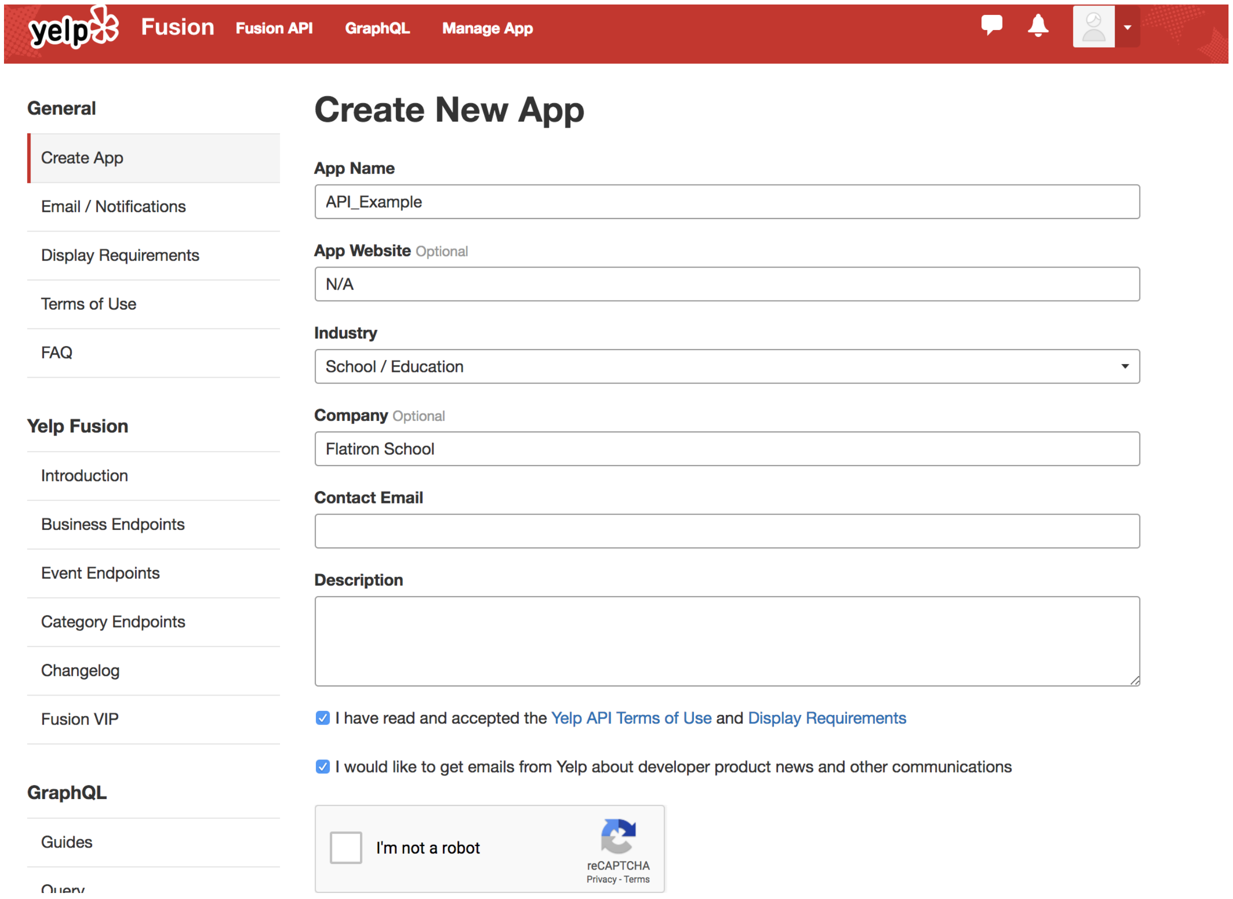
All Yelp Fusion API endpoints are under `https://api.yelp.com/v3`. Below are Yelp Fusion's current endpoints. Click the links for detailed documentation. You can also try it out by yourself using Postman!

You can also find small code samples for Node.js, Python, Ruby, and other languages at Yelp Fusion Code Samples on Github.

Name	Path	Description
Business Search	/businesses/search	Search for businesses by keyword, category, location, price level, etc.
Phone Search	/businesses/search/phone	Search for businesses by phone number.
Transaction Search	/transactions/{transaction_type}/search	Search for businesses which support food delivery transactions.
Business Details	/businesses/{id}	Get rich business data, such as name, address, phone number, photos, Yelp rating, price levels and hours of operation.
Business Match	/businesses/matches	Find the Yelp business that matches an exact input location. Use this to match business data from other sources with Yelp businesses.
Reviews	/businesses/{id}/reviews	Get up to three review excerpts for a business.
Autocomplete	/autocomplete	Provide autocomplete suggestions for businesses, search keywords and categories.

As you see at the top, the first piece of almost every API is authentication.

This is where we started in the previous codealong, where we went to https://www.yelp.com/developers/v3/manage_app (https://www.yelp.com/developers/v3/manage_app) and created a new app.



Create New App

App Name: API_Example

App Website Optional: N/A

Industry: School / Education

Company Optional: Flatiron School

Contact Email:

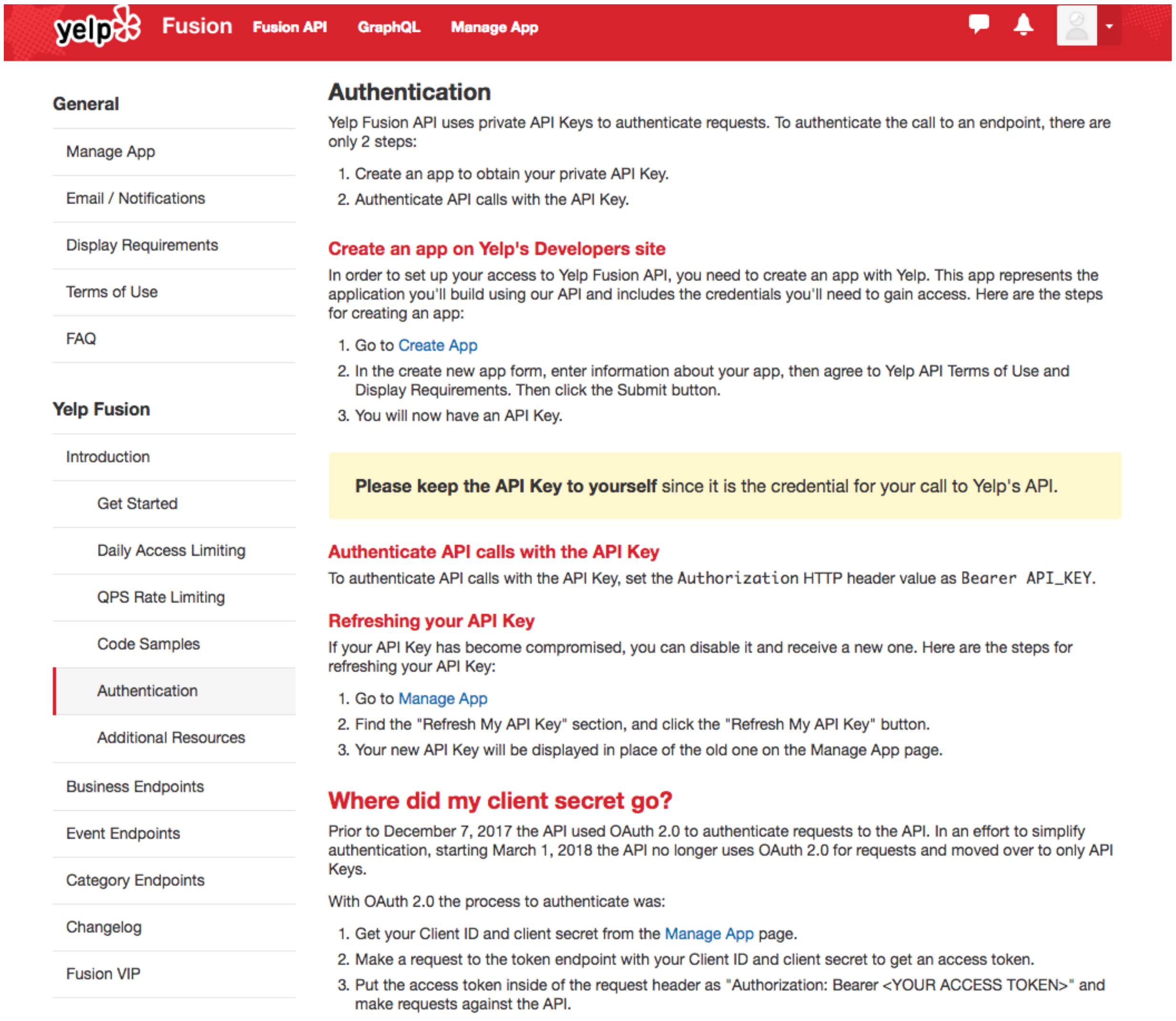
Description:

[✓] I have read and accepted the Yelp API Terms of Use and Display Requirements

[✓] I would like to get emails from Yelp about developer product news and other communications

[ ] I'm not a robot

Let's take a closer look at Yelp's authentication documentation: https://www.yelp.com/developers/documentation/v3/authentication (https://www.yelp.com/developers/documentation/v3/authentication)



Authentication

Yelp Fusion API uses private API Keys to authenticate requests. To authenticate the call to an endpoint, there are only 2 steps:

1. Create an app to obtain your private API Key.
2. Authenticate API calls with the API Key.

Create an app on Yelp's Developers site

In order to set up your access to Yelp Fusion API, you need to create an app with Yelp. This app represents the application you'll build using our API and includes the credentials you'll need to gain access. Here are the steps for creating an app:

1. Go to Create App
2. In the create new app form, enter information about your app, then agree to Yelp API Terms of Use and Display Requirements. Then click the Submit button.
3. You will now have an API Key.

Please keep the API Key to yourself since it is the credential for your call to Yelp's API.

Authenticate API calls with the API Key

To authenticate API calls with the API Key, set the `Authorization` HTTP header value as `Bearer API_KEY`.

Refreshing your API Key

If your API Key has become compromised, you can disable it and receive a new one. Here are the steps for refreshing your API Key:

1. Go to Manage App
2. Find the "Refresh My API Key" section, and click the "Refresh My API Key" button.
3. Your new API Key will be displayed in place of the old one on the Manage App page.

Where did my client secret go?

Prior to December 7, 2017 the API used OAuth 2.0 to authenticate requests to the API. In an effort to simplify authentication, starting March 1, 2018 the API no longer uses OAuth 2.0 for requests and moved over to only API Keys.

With OAuth 2.0 the process to authenticate was:

1. Get your Client ID and client secret from the Manage App page.
2. Make a request to the token endpoint with your Client ID and client secret to get an access token.
3. Put the access token inside of the request header as "Authorization: Bearer <YOUR ACCESS TOKEN>" and make requests against the API.

Notice in the documentation, it gives us the specific format "Put the API Key in the request header as "Authorization: Bearer "." This is what we passed in our get request.

Before we do this, let's import your authentication token which you have appropriately stored in **a separate file** from the codealong before.

In [ ]:
```python
import json

#Our previous function for loading our api key file
def get_keys(path):
    with open(path) as f:
        return json.load(f)
```

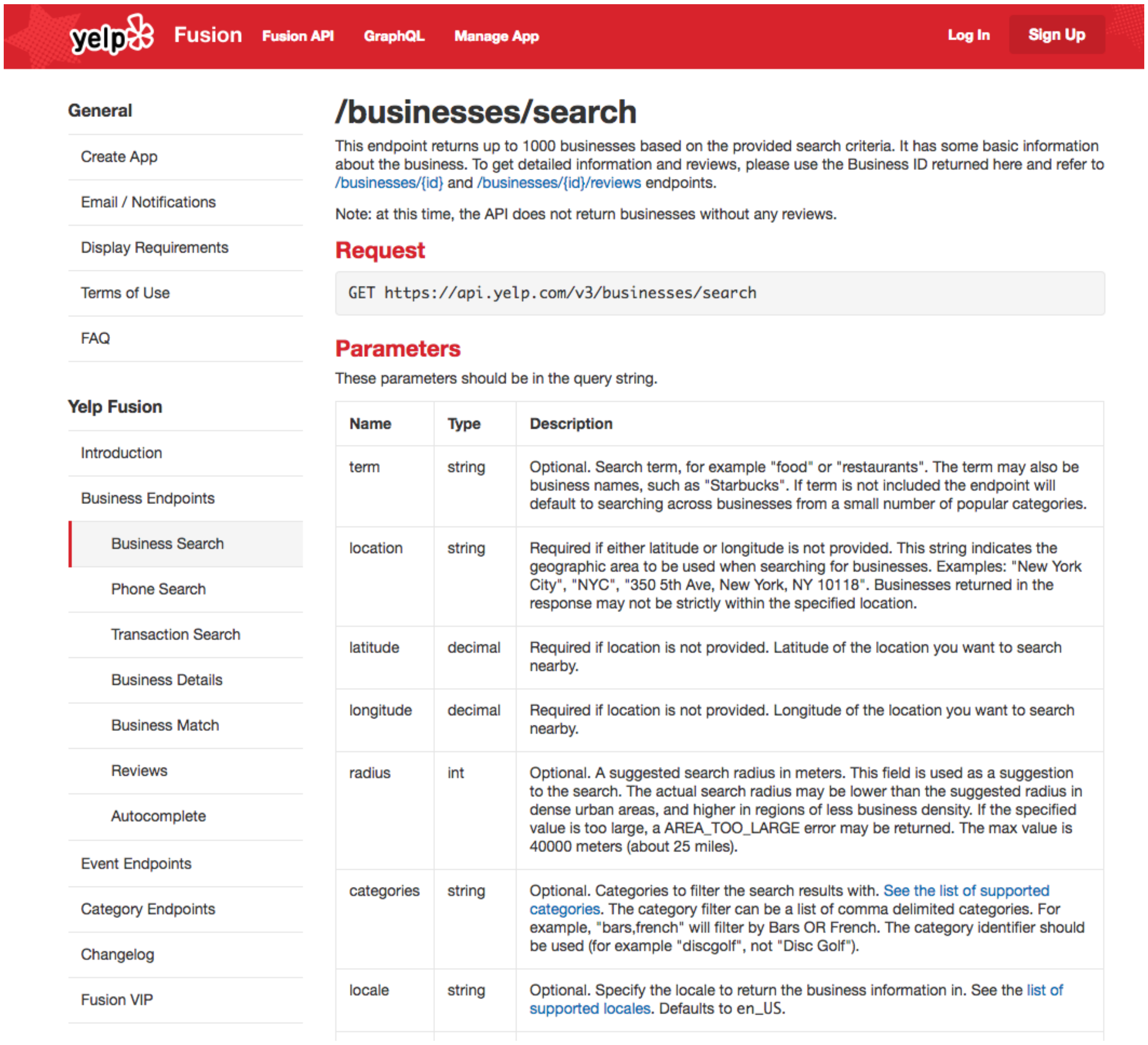
> **Note**: Like before, change the file path below to be your root directory. If you're not sure what your username is, check it with `pwd`
> For example, my current working directory is `/Users/matthew.mitchell/Documents/dsc-using-yelp-api-codealong`
> So the line below would become: `keys = get_keys("/Users/matthew.mitchell/.secret/yelp_api.json")`

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!!!
```

In [2]:
```
import requests
```

In [ ]:
```
url = #This will be our next step
header = {"Authorization" : "Bearer {}".format(api_key)}
response = requests.get(url, header=header)
```

With that, let's take a look at how the rest of our request should be formatted. Go to https://www.yelp.com/developers/documentation/v3/business_search (https://www.yelp.com/developers/documentation/v3/business_search) and take a couple of minutes to look things over.



/businesses/search

This endpoint returns up to 1000 businesses based on the provided search criteria. It has some basic information about the business. To get detailed information and reviews, please use the Business ID returned here and refer to /businesses/{id} and /businesses/{id}/reviews endpoints.

Note: at this time, the API does not return businesses without any reviews.

Request

```
GET https://api.yelp.com/v3/businesses/search
```

Parameters

These parameters should be in the query string.

Name	Type	Description
term	string	Optional. Search term, for example "food" or "restaurants". The term may also be business names, such as "Starbucks". If term is not included the endpoint will default to searching across businesses from a small number of popular categories.
location	string	Required if either latitude or longitude is not provided. This string indicates the geographic area to be used when searching for businesses. Examples: "New York City", "NYC", "350 5th Ave, New York, NY 10118". Businesses returned in the response may not be strictly within the specified location.
latitude	decimal	Required if location is not provided. Latitude of the location you want to search nearby.
longitude	decimal	Required if location is not provided. Longitude of the location you want to search nearby.
radius	int	Optional. A suggested search radius in meters. This field is used as a suggestion to the search. The actual search radius may be lower than the suggested radius in dense urban areas, and higher in regions of less business density. If the specified value is too large, a AREA_TOO_LARGE error may be returned. The max value is 40000 meters (about 25 miles).
categories	string	Optional. Categories to filter the search results with. See the list of supported categories. The category filter can be a list of comma delimited categories. For example, "bars,french" will filter by Bars OR French. The category identifier should be used (for example "discgolf", not "Disc Golf").
locale	string	Optional. Specify the locale to return the business information in. See the list of supported locales. Defaults to en_US.

Notice the first part is the format of the get request! This is the URL we pass into our get request. From there, the available parameters that you can pass are listed. These define your query, some are required while others are optional.

Reviewing our python package we thus have:

In [ ]:
```
url = 'https://api.yelp.com/v3/businesses/search'

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

url_params = {
                'location': 'NYC'
            }
response = requests.get(url, headers=headers, params=url_params)
```

Note that location or latitude and longitude are the only required parameters. That said, you are free to pass as many parameters as you want such as:

In [ ]:
```
url = 'https://api.yelp.com/v3/businesses/search'

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

url_params = {
                'location': 'NYC',
                'term' : 'pizza',
                'limit' : 50,
                'price' : "1,2,3,4",
                'open_now' : True
            }
response = requests.get(url, headers=headers, params=url_params)
```

The final important note is how some of the parameters have alternative formats. For example, `limit` is an integer and `open_now` is a boolean according to the documentation. Following these conventions is essential to receiving valid responses.

Summary

Congratulations! Not only have you seen an API now, you've also practiced sifting through the documentation. The last piece in working with APIs is developing a more solid understanding of JSON files; the data format typically returned by modern APIs. Take some additional time and familiarize yourself with some further aspects of the documentation which you may wish to investigate in the upcoming lab!