CS181 Final Project

Spotify API and Billboard Webscraping

Alex Tubbs and Hieu Nguyen Notebook 1

In this notebook, we are creating a url that can be shared with classmates. If they accept and allow us access, they will receive a code which they'll give to us, and then we can exchange that code for a token. With that token, we will then be able to access a specified amount of data about that user and their Spotify account, and use this to study listening patterns of our class.

In [1]:
```
import json
import requests
```

In [2]:
```
with open("creds.json", "r") as file:
    creds = json.load(file)
```

In [3]:
```
protocol = "https"
location = "accounts.spotify.com"
auth_resource = "/authorize"

authurl_fmt = "{}://{}{}"
authurl = authurl_fmt.format(protocol, location, auth_resource)
print(authurl)
```

```
https://accounts.spotify.com/authorize (https://accounts.spotify.com/authorize)
```

Spotify's documentation was super helpful for this first notebook, laying out the exact url we need with the parameters and what needed to be in each of those parameters. Because of this we really didn't have many issues with creating the url. The one issue we had was instead of having spaces between the scopes we tried to use +'s to make it more like a url, but it worked once we realized it needed to be a space separated list and changed it to spaces.

In [4]:
```
urlquery = {}
urlquery['client_id'] = creds['spotify']['appid']
urlquery['redirect_uri'] = creds['spotify']['redirect_uri']
urlquery['response_type'] = 'code'
urlquery['state'] = '12345678910'
urlquery['scope'] = "user-read-recently-played user-follow-read user-top-read user-library-read"
```

We decided on these scopes because they all are read only, so we couldn't accidentally change anything on anyone's account. This also gave us a wide range of data to collect since we weren't positive on the questions that we wanted to ask, so we would have access to different aspects of people's listening, whether it be what they're listening to recently, what they've listened to the most, or the songs and artists they've saved and followed, which could indicate preferences.

In [5]:
```
session = requests.Session()
p = requests.Request('GET', authurl, params=urlquery).prepare()
print(p.url)
```

```
https://accounts.spotify.com/authorize?client_id=dd7d22f635cc461fb583a5ad33eeba1c&redirect_uri=https%3
A%2F%2Flocalhost%2Fcallback%2F&response_type=code&state=12345678910&scope=user-read-recently-played+use
r-follow-read+user-top-read+user-library-read (https://accounts.spotify.com/authorize?client_id=dd7d22f
635cc461fb583a5ad33eeba1c&redirect_uri=https%3A%2F%2Flocalhost%2Fcallback%2F&response_type=code&state=1
2345678910&scope=user-read-recently-played+user-follow-read+user-top-read+user-library-read)
```

We then gave the url to the class through notebowl, and got a decent number of responses. Unfortunately we were only able to get 6 people including us, because a couple codes timed out before we could get to them since they were sent after the class period was over. Once we went to use the refresh token and obtain new access tokens, another couple users didn't work, so that left us with 6 people. This definitely turned out to be a bit of an issue once we attempted to start collecting data, because having more people would have just helped us find trends easier.

In [6]:
```
def getCodeDict():
    codemap = {}
    uservalue = input("Enter a User: ")
    while uservalue != "":
        codevalue = input("Enter Code for {}: ".format(uservalue))
        codemap[uservalue] = codevalue

        uservalue = input("Enter a User: ")
    return codemap
codemap = getCodeDict()
```

```
Enter a User: Alex
Enter Code for Alex: code
Enter a User:
```

In [7]:
```
def getToken(codemap):
    protocol = "https"
    location = "accounts.spotify.com"
    auth_resource = "/api/token"
    access_fmt = "{}://{}{}"
    accessurl = access_fmt.format(protocol, location, auth_resource)

    dataD = {}
    dataD['client_id'] = creds['spotify']['appid']
    dataD['client_secret'] = creds['spotify']['appsecret']
    dataD['redirect_uri'] = creds['spotify']['redirect_uri']
    dataD['grant_type'] = "authorization_code"
    resp = requests.post(accessurl, data=dataD)

    tokenmap = {}
    for user, code in codemap.items():
        dataD['code'] = code
        resp = requests.post(accessurl, data = dataD)
        if resp.status_code == 200:
            retval = resp.json()
            if 'access_token' in retval:
                tokenmap[user] = [retval['refresh_token'], retval['access_token']]
            else:
                print('No access token found in result:', str(retval))
        else:
            print("HTTP error on exhange of code for token", str(resp.status_code), str(resp.text))

    return tokenmap

tokenmap = getToken(codemap)
```

```
HTTP error on exhange of code for token 400 {"error":"invalid_grant","error_description":"Invalid autho
rization code"}
```

We used only a slightly different version to collect multiple tokens than the only you gave to us. We changed the data dictionary according to the specifications in Spotify's Documentation, so that it had all of the necessary information in the post body to communicate with Spotify. We also changed the original function so that we collected both the refresh token and the access token in the tokenmap, after realizing that we needed both since the token was only good for an hour. We changed it to collect both when we only received Hieu's access token, and realized afterwards that we had no way to get to his refresh token, which was necessary for when we wanted to do data collection. This was easily fixable because Hieu could just request a new code. Luckily we didn't need to get more codes from the class, since we changed it in time to get the refresh token from everyone else.