Setup Set directory locations. If working on Google Colab: copy files and install required libraries.

ADS 509 Module 1: APIs and Web Scraping

In [1]:
```
#pip show tweepy
```

In [2]:
```
pip install tweepy==4.10.1
```

```
Requirement already satisfied: tweepy==4.10.1 in c:\users\abana\anaconda3\lib\site-packa
ges (4.10.1)
Requirement already satisfied: requests-oauthlib<2,>=1.2.0 in c:\users\abana\anaconda3\l
ib\site-packages (from tweepy==4.10.1) (1.3.1)
Requirement already satisfied: oauthlib<4,>=3.2.0 in c:\users\abana\anaconda3\lib\site-p
ackages (from tweepy==4.10.1) (3.2.2)
Requirement already satisfied: requests<3,>=2.27.0 in c:\users\abana\anaconda3\lib\site-
packages (from tweepy==4.10.1) (2.28.2)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\abana\anaconda3\lib\site-p
ackages (from requests<3,>=2.27.0->tweepy==4.10.1) (2020.12.5)
Requirement already satisfied: idna<4,>=2.5 in c:\users\abana\anaconda3\lib\site-package
s (from requests<3,>=2.27.0->tweepy==4.10.1) (2.10)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in c:\users\abana\anaconda3\lib\sit
e-packages (from requests<3,>=2.27.0->tweepy==4.10.1) (1.26.4)
Requirement already satisfied: charset-normalizer<4,>=2 in c:\users\abana\anaconda3\lib
\site-packages (from requests<3,>=2.27.0->tweepy==4.10.1) (3.0.1)
Note: you may need to restart the kernel to use updated packages.
```

Twitter API Pull

In [3]:
```
# for the twitter section
import tweepy
import os
import datetime
import re
from pprint import pprint

# for the lyrics scrape section
import requests
import time
from bs4 import BeautifulSoup
from collections import defaultdict, Counter
```

In [4]:
```
import random
import shutil
```

In [5]:
```
#from google.colab import drive
#drive.mount('/content/drive')

!python api_keys.py
```

In [6]:
```
from api_keys import api_key, api_key_secret, bearer_token
```

In [7]:
```
client = tweepy.Client(bearer_token,wait_on_rate_limit=True)
```

testing api with my twitter

Testing the API

In [8]:
```
handle = "abrezzy3"
user_obj = client.get_user(username=handle)
print(user_obj)
followers = client.get_users_followers(
    # Learn about user fields here:
    # https://developer.twitter.com/en/docs/twitter-api/data-dictionary/object-model/us
    user_obj.data.id, user_fields=["created_at","description","location",
                                   "public_metrics"]
)
```

```
Response(data=<User id=180105177 name=AB Negusu username=abrezzy3>, includes={}, errors=
[], meta={})
```

Now let's explore these a bit. We'll start by printing out names, locations, following count, and followers count for these users.

In [9]:
```
#for idx, user in enumerate(followers.data) :
#    print(user.public_metrics)
followers[-1]
```

Out[9]:
```
{'result_count': 100, 'next_token': '0QDD068V2OH1AZZZ'}
```

In [10]:
```
num_to_print = 20

for idx, user in enumerate(followers.data) :
    following_count = user.public_metrics['following_count']
    followers_count = user.public_metrics['followers_count']

    print(f"{user.name} lists '{user.location}' as their location.")
    print(f" Following: {following_count}, Followers: {followers_count}.")
    print()

    if idx >= (num_to_print - 1) :
        break
```

```
Anisi habibi lists 'Ethiopia' as their location.
 Following: 1034, Followers: 189.

😍 lists 'None' as their location.
 Following: 65, Followers: 13.

Estrella lists 'Los Angeles, CA' as their location.
 Following: 82, Followers: 25.

Stax lists 'Barara, Ethiopia' as their location.
 Following: 422, Followers: 57.

ArayaK lists 'None' as their location.
 Following: 105, Followers: 31.
```

```
Erick Montanez lists 'None' as their location.
 Following: 65, Followers: 17.

M E R O N 🌹 lists 'Ethiopia' as their location.
 Following: 79, Followers: 11.

GivingJesusHisMoneysWorth lists '369' as their location.
 Following: 94, Followers: 85.

ᴹ𓂀 lists 'دنفر' as their location.
 Following: 645, Followers: 771.

Kdriver #86549 lists 'None' as their location.
 Following: 63, Followers: 6.

Francisco lists '#TeamPW' as their location.
 Following: 488, Followers: 417.

Eyob lists 'Denver, CO' as their location.
 Following: 891, Followers: 2021.

Jayynets✨ lists 'None' as their location.
 Following: 110, Followers: 88.

Esteria lists 'United States' as their location.
 Following: 8, Followers: 0.

iren lists 'None' as their location.
 Following: 539, Followers: 316.

mohan lists 'New York, USA' as their location.
 Following: 152, Followers: 121.

Dorbor Mulbah lists 'None' as their location.
 Following: 8, Followers: 4.

Lawd Stark lists 'None' as their location.
 Following: 53, Followers: 3.

eeckidpastor lists 'Colorado Rocky Mountain High' as their location.
 Following: 29, Followers: 2.

Wondwosen kefyalew lists 'None' as their location.
 Following: 209, Followers: 36.
```

Let's find the person who follows this handle who has the most followers.

In [11]:

```
max_followers = 0

for idx, user in enumerate(followers.data) :
    followers_count = user.public_metrics['followers_count']

    if followers_count > max_followers :
        max_followers = followers_count
        max_follower_user = user


print(max_follower_user)
print(max_follower_user.public_metrics)
```

```
RealCoryMachado
{'followers_count': 377496, 'following_count': 373960, 'tweet_count': 18465, 'listed_cou
nt': 150}
```

Let's pull some more user fields and take a look at them. The fields can be specified in the user_fields argument.

In [12]:
```
response = client.get_user(id=user_obj.data.id,
                           user_fields=["created_at","description","location",
                                        "entities","name","pinned_tweet_id","profile_ima
                                        "verified","public_metrics"])
```

In [13]:
```
for field, value in response.data.items() :
    print(f"for {field} we have {value}")
```

```
for description we have -MSU Denver Alumni                                  -One smile at a t
ime
for created_at we have 2010-08-18 21:13:37+00:00
for location we have Denver,CO
for id we have 180105177
for verified we have False
for public_metrics we have {'followers_count': 293, 'following_count': 282, 'tweet_coun
t': 11217, 'listed_count': 0}
for name we have AB Negusu
for pinned_tweet_id we have 989137300519964672
for username we have abrezzy3
for profile_image_url we have https://pbs.twimg.com/profile_images/1297939752927158272/G
ofvzPc4_normal.jpg
```

Now a few questions for you about the user object.

Q: How many fields are being returned in the response object?

A: 9 fields

Q: Are any of the fields within the user object non-scalar? (I.e., more complicated than a simple data type like integer, float, string, boolean, etc.)

A: Location, public metric, and profile image, created at.

Q: How many friends, followers, and tweets does this user have?

A:followers_count': 293, 'following_count': 282, 'tweet_count': 11217

Although you won't need it for this assignment, individual tweets can be a rich source of text-based data. To illustrate the concepts, let's look at the last few tweets for this user. You are encouraged to explore the fields that are available about Tweets.

In [14]:
```
response = client.get_users_tweets(user_obj.data.id)

# By default, only the ID and text fields of each Tweet will be returned
for idx, tweet in enumerate(response.data) :
    print(tweet.id)
    print(tweet.text)
    print()
```

```
    if idx > 10 :
        break
```

```
1443445647059795971
RT @TheTraeYoung: Time is valuable. Don’t waste it.

1413693971230957580
@ConnoHaug Dawg!!! I’m weak!!!!! https://t.co/3xIZhTPmVI

1392000041464713218
@tjaramillo5280 https://t.co/VRCdMSE3oh

1351340508220542977
@JayynetsP https://t.co/J3Tq9Z8iGF

1345776911553728515
@MikeTheCompass  with the current climate with sneakers what is your current preference
option of ways to buy sneakers? StockX or GoatApp

1345619077641818112
RT @ThatBoyBah: I left my heart in the streets, so ain’t no sympathy

1333257856632569856
@SwervinJerg 😂😂

1326005916588630016
@serena_bena They flip easily might not want to.

1325117522354466819
RT @Complex: BREAKING: Joe Biden elected the next President of the United States. Kamala
Harris elected the first female Vice President.

@…

1322544449868636160
@Nike  do better with your SNKRS App.  Smh https://t.co/UcFWYoDzsv
```

Pulling Follower Information

In this next section of the assignment, we will pull information about the followers of your two artists. We've seen above how to pull a set of followers using client.get_users_followers. This function has a parameter, max_results, that we can use to change the number of followers that we pull. Unfortunately, we can only pull 1000 followers at a time, which means we will need to handle the pagination of our results.

The return object has the .data field, where the results will be found. It also has .meta, which we use to select the next "page" in the results using the next_token result. I will illustrate the ideas using our user from above.

Rate Limiting

Twitter limits the rates at which we can pull data, as detailed in this guide. We can make 15 user requests per 15 minutes, meaning that we can pull users per hour. I illustrate the handling of rate

limiting below, though whether or not you hit that part of the code depends on your value of handle.

In the below example, I'll pull all the followers, 25 at a time. (We're using 25 to illustrate the idea; when you do this set the value to 1000.)

In [15]:
```
handle_followers = []
pulls = 0
max_pulls = 100
next_token = None

while True :

    followers = client.get_users_followers(
        user_obj.data.id,
        max_results=25, # when you do this for real, set this to 1000!
        pagination_token = next_token,
        user_fields=["created_at","description","location",
                     "entities","name","pinned_tweet_id","profile_image_url",
                     "verified","public_metrics"]
    )
    pulls += 1

    for follower in followers.data :
        follower_row = (follower.id,follower.name,follower.created_at,follower.descript
        handle_followers.append(follower_row)

    if 'next_token' in followers.meta and pulls < max_pulls :
        next_token = followers.meta['next_token']
    else :
        break
```

In [16]:
```
followers.data
```

Out[16]:
```
[<User id=235413026 name=Serena Smith username=serena_bena>,
 <User id=74005909 name=نیکول دیلیسا ❤️ username=love_DeLisa>,
 <User id=423912078 name=Abby username=BlaackGold>,
 <User id=415032966 name=Mona Lena username=LenaKMarie>,
 <User id=176988967 name=D o m i n i q u e 🦋 username=Domdada___>,
 <User id=242903496 name=Brady username=BradyKinsey>,
 <User id=91890203 name=Hafrican Prince 👳‍♂️ username=Kange_West>,
 <User id=56838293 name=Kiera Taylor username=LIKED_by_FEW>,
 <User id=64342615 name=Trillson Mandela ® username=MartnLuthaTweet>,
 <User id=273235962 name=The 4th. 🦊 username=FireWorkz>,
 <User id=346933701 name=Brad username=mountainswagg>,
 <User id=108496840 name=Freezy P username=KojoRalf_Lauren>,
 <User id=189468059 name=A username=amjadwaves>,
 <User id=359699695 name=Dre Weather username=DresWayEveryday>,
 <User id=50921114 name=Jamal username=jamal_abdi>,
 <User id=325043108 name=Papi_Chulo username=1HeLL_IN_HeeLz>,
 <User id=243910487 name=TreezyForPresident username=TreezyMadeIt>,
 <User id=41854943 name=Loser Face Ultra username=imjustaddison>]
```

In [17]:
```
    followers = client.get_users_followers(
        user_obj.data.id,
```

```
        max_results=25, # when you do this for real, set this to 1000!
        pagination_token = next_token,
        user_fields=["created_at","description","location",
                     "entities","name","pinned_tweet_id","profile_image_url",
                     "verified","public_metrics"]
    )
```

Pulling Twitter Data for Your Artists

In [18]:
```
artists = dict()
for handle in ['realkcijojo','SammHenshaw'] :
    user_obj = client.get_user(username=handle,user_fields=["public_metrics"])
    artists[handle] = (user_obj.data.id,
                       handle,
                       user_obj.data.public_metrics['followers_count'])


for artist, data in artists.items() :
    print(f"It would take {data[2]/(1000*15*4):.2f} hours to pull all {data[2]} followe
```

```
It would take 0.66 hours to pull all 39470 followers for realkcijojo.
It would take 0.21 hours to pull all 12439 followers for SammHenshaw.
```

In [19]:
```
for artist, data in artists.items() :
  print(data)
```

```
(199431244, 'realkcijojo', 39470)
(135199936, 'SammHenshaw', 12439)
```

In [20]:
```
0# Make the "twitter" folder here. If you'd like to practice your programming, add func
# that checks to see if the folder exists. If it does, then "unlink" it. Then create a

if not os.path.isdir("twitter") :
    #shutil.rmtree("twitter/")
    os.mkdir("twitter")
```

In this following cells, build on the above code to pull some of the followers and their data for your two artists. As you pull the data, write the follower ids to a file called [artist name]_followers.txt in the "twitter" folder. For instance, for Cher I would create a file named cher_followers.txt. As you pull the data, also store it in an object like a list or a data frame.

In addition to creating a file that only has follower IDs in it, you will create a file that includes user data. From the response object please extract and store the following fields:

screen_name name id location followers_count friends_count description Store the fields with one user per row in a tab-delimited text file with the name [artist name]_follower_data.txt. For instance, for Cher I would create a file named cher_follower_data.txt.

One note: the user's description can have tabs or returns in it, so make sure to clean those out of the description before writing them to the file. I've included some example code to do that below the stub.

In [21]:
```
num_followers_to_pull = 200*1000 # feel free to use this to limit the number of followe
```

Getting artist followers information

In [22]:
```
handles = ['realkcijojo','SammHenshaw']
for handle in handles:
    user_obj = client.get_user(username=handle,user_fields=["public_metrics"])
    artists[handle] = (user_obj.data.id,handle,user_obj.data.public_metrics['followers_
    print(artists[handle])
```

```
(199431244, 'realkcijojo', 39470)
(135199936, 'SammHenshaw', 12439)
```

In [23]:
```
 os. getcwd()
```

Out[23]:
```
'C:\\Users\\Abana\\Downloads'
```

In [24]:
```
# Modify the below code stub to pull the follower IDs and write them to a file.

handles = ['realkcijojo','SammHenshaw']

whitespace_pattern = re.compile(r"\s+")

user_data = dict()
followers_data = dict()

for handle in handles :
    user_data[handle] = [] # will be a list of lists
    followers_data[handle] = [] # will be a simple list of IDs


# Grabs the time when we start making requests to the API
start_time = datetime.datetime.now()

for handle in handles :

    # Create the output file names

    followers_output_file = handle + "_followers.txt"
    user_data_output_file = handle + "_follower_data.txt"

    followers_output_file = handle + "_followers.txt"
    user_data_output_file = handle + "_follower_data.txt"

    # Using tweepy.Paginator (https://docs.tweepy.org/en/latest/v2_pagination.html),
    my_followers = {"id":[],"Artist":[]};
    my_followers_data = {"Artist":[],"User_Following_Artist":[],"Screen_Name":[],"follo
    for handle in handles:
        user_obj = client.get_user(username=handle,user_fields=["public_metrics"])
        followers = client.get_users_followers(
        user_obj.data.id,pagination_token=next_token,user_fields=["created_at", "descri
                                    "profile_image_url", "protected", "public_metrics
                                    "url", "username", "verified", "verified_type",
                                    "withheld"]
```

```
        )
      # following = client.get_users_following(
      # # Learn about user fields here:
      # # https://developer.twitter.com/en/docs/twitter-api/data-dictionary/object-mode



        for idx, user in enumerate(followers.data) :

            following_count = user.public_metrics['following_count']
            followers_count = user.public_metrics['followers_count']
            my_followers["Artist"].append(handle)
            my_followers_data["Artist"].append(handle)
            my_followers_data["User_Following_Artist"].append(user.username)
            my_followers_data["Screen_Name"].append(user.name)
            my_followers_data["following_counts"].append(following_count)
            my_followers_data["followers_counts"].append(followers_count)
            my_followers["id"].append(user.id)
            my_followers_data["id"].append(user.id)
            my_followers_data["URL"].append(user.profile_image_url)
            my_followers_data["Location"].append(user.location)
            my_followers_data["description"].append(user.description)









    # use `get_users_followers` to pull the follower data requested.

    # For each response object, extract the needed fields and store them in a dictionar
    # data frame.
    import pandas as pd

    my_followers_dataframe = pd.DataFrame(my_followers)
    my_followers_data_dataframe = pd.DataFrame(my_followers_data)

    for i in handles:
        abc = my_followers_dataframe[my_followers_dataframe['Artist']==i]
        bcd = my_followers_data_dataframe[my_followers_data_dataframe['Artist']==i]
        file_path= f'C:\\Users\\Abana\\Downloads\\twitter\\{i}_followers.txt'
        with open(file_path, 'w') as f:
            f.write(abc.to_string())
        file_path= f'C:\\Users\\Abana\\Downloads\\twitter\\{i}_followers_data.txt'
        with open(file_path, 'w',errors = 'ignore') as f:
            f.write(bcd.to_string())



# Let's see how long it took to grab all follower IDs
end_time = datetime.datetime.now()
print(end_time - start_time)
```

```
Rate limit exceeded. Sleeping for 759 seconds.
---------------------------------------------------------------------------
UnicodeEncodeError                        Traceback (most recent call last)
<ipython-input-24-e520d659407f> in <module>
     82         file_path= f'C:\\Users\\Abana\\Downloads\\twitter\\{i}_followers_data.tx
t'
```

```
     83         with open(file_path, 'w') as f:
---> 84             f.write(bcd.to_string())
     85
     86

~\anaconda3\lib\encodings\cp1252.py in encode(self, input, final)
     17 class IncrementalEncoder(codecs.IncrementalEncoder):
     18     def encode(self, input, final=False):
---> 19         return codecs.charmap_encode(input,self.errors,encoding_table)[0]
     20
     21 class IncrementalDecoder(codecs.IncrementalDecoder):

UnicodeEncodeError: 'charmap' codec can't encode character '\U0001f49b' in position 176
7: character maps to <undefined>
```

In [27]:

```
tricky_description = """
    Home by Warsan Shire

    no one leaves home unless
    home is the mouth of a shark.
    you only run for the border
    when you see the whole city
    running as well.

"""
# This won't work in a tab-delimited text file.

clean_description = re.sub(r"\s+"," ",tricky_description).strip()
clean_description
```

Out[27]:

```
'Home by Warsan Shire no one leaves home unless home is the mouth of a shark. you only r
un for the border when you see the whole city running as well.'
```

Lyrics Scrape

This section asks you to pull data from the Twitter API and scrape www.AZLyrics.com. In the notebooks where you do that work you are asked to store the data in specific ways.

In [28]:

```
artists = {'realkcijojo':"https://www.azlyrics.com/k/kcijojo.html",
           'SammHenshaw':"https://www.azlyrics.com/s/sammhenshaw.html"}
# we'll use this dictionary to hold both the artist name and the link on AZlyrics
```

Part 1: Finding Links to Songs Lyrics

In [29]:

```
# Let's set up a dictionary of lists to hold our links
lyrics_pages = defaultdict(list)

for artist, artist_page in artists.items() :
    lyrics_pages[artist] = []
    # request the page and sleep
    r = requests.get(artist_page)
    time.sleep(5 + 10*random.random())


    # now extract the links to lyrics pages from this page
    # store the links `lyrics_pages` where the key is the artist and the
    # value is a list of links.
```

```
    soup = BeautifulSoup(r.text)
    for a in soup.find_all('a', href=True):
        if "/lyrics/" in a.attrs['href']:
            link_lyrics_page = "https://www.azlyrics.com" + a.attrs['href']
            lyrics_pages[artist].append(link_lyrics_page)
```

In [30]:
```
for artist, lp in lyrics_pages.items():
    assert(len(set(lp)) > 20)
```

In [31]:
```
# Let's see how long it's going to take to pull these lyrics
#if we're waiting `5 + 10*random.random()` seconds
for artist, links in lyrics_pages.items() :
  time.sleep(5 + 10*random.random())
  print(f"For {artist} we have {len(links)}.")
  print(f"The full pull will take for this artist will take {round(len(links)*10/3600,2
```

```
For realkcijojo we have 65.
The full pull will take for this artist will take 0.18 hours.
For SammHenshaw we have 38.
The full pull will take for this artist will take 0.11 hours.
```

Part 2: Pulling Lyrics

In [32]:
```
def generate_filename_from_link(link) :

    if not link :
        return None

    # drop the http or https and the html
    name = link.replace("https","").replace("http","")
    name = name.replace(".html","")

    name = name.replace("/lyrics/","")

    # Replace useless chareacters with UNDERSCORE
    name = name.replace("://","").replace(".","_").replace("/","_")

    # tack on .txt
    name = name + ".txt"

    return(name)
```

In [33]:
```
# Make the lyrics folder here. If you'd like to practice your programming, add function
# that checks to see if the folder exists. If it does, then use shutil.rmtree to remove

if os.path.isdir("lyrics") :
    shutil.rmtree("lyrics/")

os.mkdir("lyrics")
```

In [37]:
```
url_stub = "https://www.azlyrics.com"
start = time.time()

total_pages = 0
```

```
for artist in lyrics_pages :

    # Use this space to carry out the following steps:

    # 1. Build a subfolder for the artist
    if os.path.isdir(os.path.join("lyrics", artist)) :
        shutil.rmtree(os.path.join("lyrics", artist))
    os.mkdir(os.path.join("lyrics", artist))
    # 2. Iterate over the lyrics pages
    for idx, page in enumerate(lyrics_pages[artist]):
        print(page)
    # 3. Request the lyrics page.
        # Don't forget to add a line like `time.sleep(5 + 10*random.random())`
        # to sleep after making the request
        r = requests.get(page)
        time.sleep(5 + 10*random.random())
        soup = BeautifulSoup(r.text)

    # 4. Extract the title and lyrics from the page.
        title = soup.findAll('b')[1].text
        lyric = soup.find("div", {"class": "col-xs-12 col-lg-8 text-center"}).contents[
        lyric = list(filter(None, lyric))
        lyric = map(lambda s: s.strip(), lyric)
    # 5. Write out the title, two returns ('\n'), and the lyrics. Use `generate_filenam
    #    to generate the filename.

        with open(os.path.join("lyrics", artist, generate_filename_from_link(page)), 'w
            wFile.write(title + '\n\n')
            wFile.write('.'.join(lyric))
        if idx==21:
            break
    # Remember to pull at least 20 songs per artist. It may be fun to pull all the song
```

```
https://www.azlyrics.com/lyrics/kcijojo/hbi.html
https://www.azlyrics.com/lyrics/kcijojo/lastnightsletter.html
https://www.azlyrics.com/lyrics/kcijojo/babycomeback.html
https://www.azlyrics.com/lyrics/kcijojo/justforyourlove.html
https://www.azlyrics.com/lyrics/kcijojo/nowandforever.html
https://www.azlyrics.com/lyrics/kcijojo/dontrushtakeloveslowly.html
https://www.azlyrics.com/lyrics/kcijojo/youbringmeup.html
https://www.azlyrics.com/lyrics/kcijojo/stillwaiting.html
https://www.azlyrics.com/lyrics/kcijojo/loveballad.html
https://www.azlyrics.com/lyrics/kcijojo/howmanytimes.html
https://www.azlyrics.com/lyrics/kcijojo/allmylife.html
https://www.azlyrics.com/lyrics/kcijojo/howcouldyou.html
https://www.azlyrics.com/lyrics/kcijojo/intro.html
https://www.azlyrics.com/lyrics/kcijojo/feefiefoefum.html
https://www.azlyrics.com/lyrics/kcijojo/iwannamakelovetoyou.html
https://www.azlyrics.com/lyrics/kcijojo/iwannagettoknowyou.html
https://www.azlyrics.com/lyrics/kcijojo/hellodarlin.html
https://www.azlyrics.com/lyrics/kcijojo/howlongmusticry.html
https://www.azlyrics.com/lyrics/kcijojo/makinmesaygoodbye.html
https://www.azlyrics.com/lyrics/kcijojo/tellmeitsreal.html
https://www.azlyrics.com/lyrics/kcijojo/life.html
https://www.azlyrics.com/lyrics/kcijojo/girl.html
https://www.azlyrics.com/lyrics/sammhenshaw/temptationintro.html
https://www.azlyrics.com/lyrics/sammhenshaw/autonomyslave.html
https://www.azlyrics.com/lyrics/sammhenshaw/everything.html
https://www.azlyrics.com/lyrics/sammhenshaw/redemption.html
https://www.azlyrics.com/lyrics/sammhenshaw/better.html
https://www.azlyrics.com/lyrics/sammhenshaw/onlywannabewithyouunplugged.html
```

```
https://www.azlyrics.com/lyrics/sammhenshaw/ourlove.html
https://www.azlyrics.com/lyrics/sammhenshaw/thesehands.html
https://www.azlyrics.com/lyrics/sammhenshaw/nightcalls.html
https://www.azlyrics.com/lyrics/sammhenshaw/chances.html
https://www.azlyrics.com/lyrics/sammhenshaw/easy.html
https://www.azlyrics.com/lyrics/sammhenshaw/stillnoalbumintro.html
https://www.azlyrics.com/lyrics/sammhenshaw/thoughtsandprayers.html
https://www.azlyrics.com/lyrics/sammhenshaw/grow.html
https://www.azlyrics.com/lyrics/sammhenshaw/chickenwings.html
https://www.azlyrics.com/lyrics/sammhenshaw/mrintrovert.html
https://www.azlyrics.com/lyrics/sammhenshaw/816.html
https://www.azlyrics.com/lyrics/sammhenshaw/mrintrovertreprise.html
https://www.azlyrics.com/lyrics/sammhenshaw/lovedbyyou.html
https://www.azlyrics.com/lyrics/sammhenshaw/taketime.html
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-37-4bb116e8fdca> in <module>
     24     # 4. Extract the title and lyrics from the page.
     25         title = soup.findAll('b')[1].text
---> 26         lyric = soup.find("div", {"class": "col-xs-12 col-lg-8 text-center"}).c
ontents[14].text.replace('\n\r\n', '').split('\n')
     27         lyric = list(filter(None, lyric))
     28         lyric = map(lambda s: s.strip(), lyric)

~\anaconda3\lib\site-packages\bs4\element.py in __getattr__(self, attr)
    919             return self
    920         else:
--> 921             raise AttributeError(
    922                 "'%s' object has no attribute '%s'" % (
    923                     self.__class__.__name__, attr))

AttributeError: 'NavigableString' object has no attribute 'text'
```

Evaluation

In [38]:

```
# Simple word extractor from Peter Norvig: https://norvig.com/spell-correct.html
def words(text):
    return re.findall(r'\w+', text.lower())
```

In [ ]:

```
twitter_files = os.listdir("twitter")
twitter_files = [f for f in twitter_files if f != ".DS_Store"]
artist_handles = list(set([name.split("_")[0] for name in twitter_files]))

print(f"We see two artist handles: {artist_handles[0]},{artist_handles[1]}.")
```

Checking Twitter Data

In [43]:

```
for artist in artist_handles :
    follower_file = artist + "_followers.txt"
    follower_data_file = artist + "_followers_data.txt"

    ids = open("C:\\Users\\Abana\\Downloads\\twitter\\" + follower_file,'r').readlines(

    print(f"We see {len(ids)-1} in your follower file for {artist}, assuming a header r

    with open("C:\\Users\\Abana\\Downloads\\twitter\\" + follower_data_file,'r') as inf

        # check the headers
        headers = infile.readline().split("\t")
```

```
        print(f"In the follower data file ({follower_data_file}) for {artist}, we have
        print(" : ".join(headers))

        description_words = []
        locations = set()


        for idx, line in enumerate(infile.readlines()) :
            line = line.strip("\n").split("\t")

            try :
                locations.add(line[3])
                description_words.extend(words(line[6]))
            except :
                pass



        print(f"We have {idx+1} data rows for {artist} in the follower data file.")

        print(f"For {artist} we have {len(locations)} unique locations.")

        print(f"For {artist} we have {len(description_words)} words in the descriptions
        print("Here are the five most common words:")
        print(Counter(description_words).most_common(5))


        print("")
        print("-"*40)
        print("")
```

```
We see 94 in your follower file for SammHenshaw, assuming a header row.
In the follower data file (SammHenshaw_followers_data.txt) for SammHenshaw, we have thes
e columns:
          Artist User_Following_Artist                                    Screen_Name  fol
lowing_counts  followers_counts         id                          Location
description
URL

We have 94 data rows for SammHenshaw in the follower data file.
For SammHenshaw we have 0 unique locations.
For SammHenshaw we have 0 words in the descriptions.
Here are the five most common words:
[]

----------------------------------------

We see 100 in your follower file for realkcijojo, assuming a header row.
In the follower data file (realkcijojo_followers_data.txt) for realkcijojo, we have thes
e columns:
         Artist User_Following_Artist                  Screen_Name  following_counts
followers_counts         id                       Location
description
URL

We have 100 data rows for realkcijojo in the follower data file.
For realkcijojo we have 0 unique locations.
For realkcijojo we have 0 words in the descriptions.
Here are the five most common words:
[]
```

```
----------------------------------------
```

Checking Lyrics

In [44]:

```
artist_folders = os.listdir("lyrics/")
artist_folders = [f for f in artist_folders if os.path.isdir("lyrics/" + f)]

for artist in artist_folders :
    artist_files = os.listdir("lyrics/" + artist)
    artist_files = [f for f in artist_files if 'txt' in f or 'csv' in f or 'tsv' in f]

    print(f"For {artist} we have {len(artist_files)} files.")

    artist_words = []

    for f_name in artist_files :
        with open("lyrics/" + artist + "/" + f_name) as infile :
            artist_words.extend(words(infile.read()))


    print(f"For {artist} we have roughly {len(artist_words)} words, {len(set(artist_wor
```

```
For realkcijojo we have 22 files.
For realkcijojo we have roughly 5690 words, 766 are unique.
For SammHenshaw we have 19 files.
For SammHenshaw we have roughly 4872 words, 679 are unique.
```