# ADS 509 Module 1: APIs and Web Scraping

This notebook has three parts. In the first part you will pull data from the Twitter API. In the second, you will scrape lyrics from AZLyrics.com. In the last part, you'll run code that verifies the completeness of your data pull.

For this assignment you have chosen two musical artists who have at least 100,000 Twitter followers and 20 songs with lyrics on AZLyrics.com. In this part of the assignment we pull the some of the user information for the followers of your artist and store them in text files.

## General Assignment Instructions

These instructions are included in every assignment, to remind you of the coding standards for the class. Feel free to delete this cell after reading it.

One sign of mature code is conforming to a style guide. We recommend the Google Python Style Guide. If you use a different style guide, please include a cell with a link.

Your code should be relatively easy-to-read, sensibly commented, and clean. Writing code is a messy process, so please be sure to edit your final submission. Remove any cells that are not needed or parts of cells that contain unnecessary code. Remove inessential `import` statements and make sure that all such statements are moved into the designated cell.

Make use of non-code cells for written commentary. These cells should be grammatical and clearly written. In some of these cells you will have questions to answer. The questions will be marked by a "Q:" and will have a corresponding "A:" spot for you. *Make sure to answer every question marked with a `Q:` for full credit.*

# Twitter API Pull

In [239…
```
# 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 [240…
```
# Use this cell for any import statements you add
#!pip install pandas
import pandas as pd
import numpy as np
import csv
import random
import re
```

We need bring in our API keys. Since API keys should be kept secret, we'll keep them in a file called

`api_keys.py`. This file should be stored in the directory where you store this notebook. The example file is provided for you on Blackboard. The example has API keys that are *not* functional, so you'll need to get Twitter credentials and replace the placeholder keys.

In [241…
```
from api_keys import api_key, api_key_secret, access_token, access_token_secret
```

In [242…
```
auth = tweepy.AppAuthHandler(api_key, api_key_secret)
api = tweepy.API(auth,wait_on_rate_limit=True) # removed the following line ,wait_on_rat

print ('API Host', api.host)
# No longer valid - print ('API Version', api.api_root)
```

```
API Host api.twitter.com
```

# Testing the API

The Twitter APIs are quite rich. Let's play around with some of the features before we dive into this section of the assignment. For our testing, it's convenient to have a small data set to play with. We will seed the code with the handle of John Chandler, one of the instructors in this course. His handle is `@37chandler`. Feel free to use a different handle if you would like to look at someone else's data.

We will write code to explore a few aspects of the API:

1. Pull all the follower IDs for @katymck.
2. Explore the user object, which gives us information about Twitter users.
3. Pull some user objects for the followers.
4. Pull the last few tweets by @katymck.

In [28]:
```
handle = "Lennyj89"

followers = []

for page in tweepy.Cursor(api.get_follower_ids,
                          # This is how we will get around the issue of not being able t
                          # Once the rate limit is hit, we will be  that we must wait 15
                          #wait_on_rate_limit=True,
                          #wait_on_rate_limit_notify=True,
                          #compression=True,
                          screen_name=handle).pages():

    # The page variable comes back as a list, so we have to use .extend rather than .app
    followers.extend(page)


print(f"Here are the first five follower ids for {handle} out of the {len(followers)} to
followers[:5]
```

Out[28]:
```
Here are the first five follower ids for Lennyj89 out of the 58 total.
[338089514,
 1124100202934284289,
 1083552502400933888,
 619587455,
 937826228211470336]
```

We have the follower IDs, which are unique numbers identifying the user, but we'd like to get some more information on these users. Twitter allows us to pull "fully hydrated user objects", which is a fancy way of saying "all the information about the user". Let's look at user object for our starting handle.

In [10]:
```
user = api.get_user(screen_name=handle)
pprint(user._json)
```


```
{'contributors_enabled': False,
 'created_at': 'Fri Feb 11 07:37:40 +0000 2011',
 'default_profile': True,
 'default_profile_image': False,
 'description': '',
 'entities': {'description': {'urls': []}},
 'favourites_count': 31,
 'follow_request_sent': None,
 'followers_count': 58,
 'following': None,
 'friends_count': 220,
 'geo_enabled': False,
 'has_extended_profile': False,
 'id': 250520140,
 'id_str': '250520140',
 'is_translation_enabled': False,
 'is_translator': False,
 'lang': None,
 'listed_count': 0,
 'location': 'Mount Sterling, KY',
 'name': 'Leonard Littleton',
 'notifications': None,
 'profile_background_color': 'C0DEED',
 'profile_background_image_url': 'http://abs.twimg.com/images/themes/theme1/bg.png',
 'profile_background_image_url_https': 'https://abs.twimg.com/images/themes/theme1/bg.pn
g',
 'profile_background_tile': False,
 'profile_image_url': 'http://pbs.twimg.com/profile_images/1241004104/IMG_0107_normal.JP
G',
 'profile_image_url_https': 'https://pbs.twimg.com/profile_images/1241004104/IMG_0107_no
rmal.JPG',
 'profile_link_color': '1DA1F2',
 'profile_location': None,
 'profile_sidebar_border_color': 'C0DEED',
 'profile_sidebar_fill_color': 'DDEEF6',
 'profile_text_color': '333333',
 'profile_use_background_image': True,
 'protected': False,
 'screen_name': 'lennyj89',
 'status': {'contributors': None,
            'coordinates': None,
            'created_at': 'Fri Jul 09 21:42:26 +0000 2021',
            'entities': {'hashtags': [{'indices': [88, 104],
                                       'text': 'SwitchToSleeper'}],
                         'symbols': [],
                         'urls': [],
                         'user_mentions': [{'id': 27098695,
                                            'id_str': '27098695',
                                            'indices': [3, 13],
                                            'name': 'Sleeper',
                                            'screen_name': 'SleeperHQ'}]},
            'favorite_count': 0,
            'favorited': False,
            'geo': None,
            'id': 1413614546992893957,
            'id_str': '1413614546992893957',
            'in_reply_to_screen_name': None,
            'in_reply_to_status_id': None,
            'in_reply_to_status_id_str': None,
            'in_reply_to_user_id': None,
            'in_reply_to_user_id_str': None,
            'is_quote_status': False,
            'lang': 'en',
```

```
            'place': None,
            'retweet_count': 419,
            'retweeted': False,
            'retweeted_status': {'contributors': None,
                                 'coordinates': None,
                                 'created_at': 'Fri Jul 09 14:55:10 +0000 2021',
                                 'entities': {'hashtags': [{'indices': [73, 89],
                                                            'text': 'SwitchToSleeper'}],
                                              'symbols': [],
                                              'urls': [{'display_url': 'twitter.com/i/we
b/status/1…',
                                                        'expanded_url': 'https://twitte
r.com/i/web/status/1413512055580098571',
                                                        'indices': [116, 139],
                                                        'url': 'https://t.co/fshHOlgll
F'}],
                                              'user_mentions': []},
                                 'favorite_count': 582,
                                 'favorited': False,
                                 'geo': None,
                                 'id': 1413512055580098571,
                                 'id_str': '1413512055580098571',
                                 'in_reply_to_screen_name': None,
                                 'in_reply_to_status_id': None,
                                 'in_reply_to_status_id_str': None,
                                 'in_reply_to_user_id': None,
                                 'in_reply_to_user_id_str': None,
                                 'is_quote_status': False,
                                 'lang': 'en',
                                 'place': None,
                                 'retweet_count': 419,
                                 'retweeted': False,
                                 'source': '<a '
                                           'href="https://mobile.twitter.com" '
                                           'rel="nofollow">Twitter Web App</a>',
                                 'text': 'Switch your league to Sleeper and '
                                         'we’ll pay part of your league dues '
                                         '💰💰💰 #SwitchToSleeper\n'
                                         '\n'
                                         '1. retweet and Like for… '
                                         'https://t.co/fshHOlgllF',
                                 'truncated': True},
            'source': '<a href="https://mobile.twitter.com" '
                      'rel="nofollow">Twitter Web App</a>',
            'text': 'RT @SleeperHQ: Switch your league to Sleeper and we’ll '
                    'pay part of your league dues 💰💰💰 #SwitchToSleeper\n'
                    '\n'
                    '1. retweet and Like for eligibili…',
            'truncated': False},
 'statuses_count': 26,
 'time_zone': None,
 'translator_type': 'none',
 'url': None,
 'utc_offset': None,
 'verified': False,
 'withheld_in_countries': []}
```

Now a few questions for you about the user object.

Q: How many fields are being returned in the _json portion of the user object?

A: It seems 44 fields are being returned in the user object.

---

Q: Are any of the fields within the user object non-scaler? TK correct term

A: Yes. The status field seems to be a non-scalar object.

---

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

A: This user has 220 friends, 58 followers, 31 favorites and 26 statuses.

We can map the follower IDs onto screen names by accessing the screen_name key within the user object. Modify the code below to also print out how many people the follower is following and how many followers they have.

In [29]:
```
ids_to_lookup = followers[:10]

for user_obj in api.lookup_users(user_id=ids_to_lookup) :
    print(f"{handle} is followed by {user_obj.screen_name}")

    # Add code here to print out friends and followers of `handle`
    print(f"{user_obj.screen_name} is followed by {user_obj.followers_count} follower(s)
```

```
Lennyj89 is followed by Doteocinco
Doteocinco is followed by 132 follower(s) and following 296 people.
Lennyj89 is followed by acoldiron0
acoldiron0 is followed by 5 follower(s) and following 14 people.
Lennyj89 is followed by Crump48864926
Crump48864926 is followed by 4 follower(s) and following 59 people.
Lennyj89 is followed by BrittKnap
BrittKnap is followed by 86 follower(s) and following 281 people.
Lennyj89 is followed by CadyPerdue
CadyPerdue is followed by 8 follower(s) and following 86 people.
Lennyj89 is followed by yesmulu
yesmulu is followed by 40 follower(s) and following 853 people.
Lennyj89 is followed by muluyes
muluyes is followed by 323 follower(s) and following 3860 people.
Lennyj89 is followed by GeorgeTThompso1
GeorgeTThompso1 is followed by 417 follower(s) and following 1584 people.
Lennyj89 is followed by Hmohamed619
Hmohamed619 is followed by 5 follower(s) and following 18 people.
Lennyj89 is followed by dewey_copley
dewey_copley is followed by 288 follower(s) and following 522 people.
```

Although you won't need it for this assignment, individual tweets (called "statuses" in the API) 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 `status` object and marvel in the richness of the data that is available.

In [21]:
```
tweet_count = 0

for status in tweepy.Cursor(api.user_timeline, id=handle).items():
    tweet_count += 1

    print(f"The tweet was tweeted at {status.created_at}.")
    print(f"The original tweet has been retweeted {status.retweet_count} times.")

    clean_status = status.text
    clean_status = clean_status.replace("\n"," ")

    print(f"{clean_status}")
    print("\n"*2)

    if tweet_count > 10 :
        break
```

```
Unexpected parameter: id
```

The tweet was tweeted at 2021-07-09 21:42:26+00:00.
The original tweet has been retweeted 419 times.
RT @SleeperHQ: Switch your league to Sleeper and we'll pay part of your league dues 💰💰💰 #SwitchToSleeper  1. retweet and Like for eligibili…

The tweet was tweeted at 2018-12-17 19:11:29+00:00.
The original tweet has been retweeted 0 times.
@MarkIngram22 @wil_lutz5 Need a combined 45 fantasy points from you guys tonight! Help a bro out??

The tweet was tweeted at 2018-12-06 14:05:01+00:00.
The original tweet has been retweeted 0 times.
@MikeTagliereNFL @bobbyfantasypro Guys, need some help with start/sit.  Need 3 of 5.  M. Ingram, J. Mixon, S. Miche… https://t.co/6npGRy0ogy

The tweet was tweeted at 2018-08-27 01:50:11+00:00.
The original tweet has been retweeted 0 times.
@MikeTagliereNFL @bobbyfantasypro My bench is r. freeman, m. Ingram, m. Lynch, s. Watkins, n. Agholor and d. Parker… https://t.co/equ5Le9ziE

The tweet was tweeted at 2018-08-27 01:36:04+00:00.
The original tweet has been retweeted 0 times.
@bobbyfantasypro @MikeTagliereNFL Could you guys give some advice on later round draft strategies when in a smaller… https://t.co/kA61ilCrBm

The tweet was tweeted at 2017-12-08 19:56:54+00:00.
The original tweet has been retweeted 0 times.
@rotobuzzguy Yeah, a friend and I were debating whether it was a sound strategy or a crappy thing to do...I was lea… https://t.co/qEoiW6zCuk

The tweet was tweeted at 2017-12-08 17:50:42+00:00.
The original tweet has been retweeted 0 times.
@rotobuzzguy Thoughts on playoff teams benching their lineups in order to adjust playoff matchups? Is it an acceptable strategy?

The tweet was tweeted at 2017-12-05 16:34:29+00:00.
The original tweet has been retweeted 0 times.
@rotobuzzguy Top WRs RBs available according to https://t.co/PU1HrevA3A https://t.co/At0a7h1JFt

The tweet was tweeted at 2017-12-05 16:34:03+00:00.
The original tweet has been retweeted 0 times.
@rotobuzzguy Bench https://t.co/Q50oen1izW

The tweet was tweeted at 2017-12-05 16:26:00+00:00.
The original tweet has been retweeted 0 times.
@rotobuzzguy I'm in #desperationmode.  Need a win to get in.  12-team standard.  Any of these pieces you'd consider… https://t.co/42AYU3Ubf1

```
The tweet was tweeted at 2017-11-30 16:37:02+00:00.
The original tweet has been retweeted 0 times.
@MikeTagliereNFL Thoughts on Diggs if Trufant suits up?
```

# Pulling Follower Information

In this next section of the assignment, we will pull information about the followers of your two artists. We must first get the follower IDs, then we will be able to "hydrate" the IDs, pulling the user objects for them. Once we have those user objects we will extract some fields that we can use in future analyses.

The Twitter API only allows users to make 15 requests per 15 minutes when pulling followers. Each request allows you to gather 5000 follower ids. Tweepy will grab the 15 requests quickly then wait 15 minutes, rather than slowly pull the requests over the time period. Before we start grabbing follower IDs, let's first just check how long it would take to pull all of the followers. To do this we use the `followers_count` item from the user object.

In [269…
```
# I'm putting the handles in a list to iterate through below
handles = ['mtrench','NateWantsToBtl']

# This will iterate through each Twitter handle that we're collecting from
for screen_name in handles:

    # Tells Tweepy we want information on the handle we're collecting from
    # The next line specifies which information we want, which in this case is the numbe
    user = api.get_user(screen_name=screen_name)
    followers_count = user.followers_count

    # Let's see roughly how long it will take to grab all the follower IDs.
    print(f'''
    @{screen_name} has {followers_count} followers.
    That will take roughly {followers_count/(5000*15*4):.2f} hours to pull the followers
    ''')
```

```
    @mtrench has 191564 followers.
    That will take roughly 0.64 hours to pull the followers.


    @NateWantsToBtl has 185140 followers.
    That will take roughly 0.62 hours to pull the followers.
```

In [286…
```
basepath = 'C:\\Users\\lenny\\'
```

As we pull data for each artist we will write their data to a folder called "twitter", so we will make that folder if needed.

In [7]:
```
# Make the "twitter" folder here. If you'd like to practice your programming, add functi
# that checks to see if the folder exists. If it does, then "unlink" it. Then create a n

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

In this following cells, use the `api.followers_ids` (and the `tweepy.Cursor` functionality) to pull some of the followers 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 [7]:
```
num_followers_to_pull = 60*1000 # feel free to use this to limit the number of followers
```

In [270…
```
#Set the root directory to the new twitter file created

root=os.getcwd()
directory='twitter'

filepath=os.path.join(root, directory)
```

In [9]:
```
# Modify the below code stub to pull the follower IDs and write them to a file.

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

for handle in handles :

    output_file = handle + "_followers.txt"


    # Pull and store the follower IDs
    handle_followers = []
    for page in tweepy.Cursor(api.get_follower_ids,
                              screen_name=handle).pages():
        handle_followers.extend(page)

    # Write the IDs to the output file in the `twitter` folder.
    completeName=os.path.join(filepath, output_file)
    f = open(completeName, "a")
    for id in handle_followers[:-1]:
        f.write(str(id)+'\t')
    f.write(str(handle_followers[-1]))
    f.close()

    # If you've pulled num_followers_to_pull, feel free to break out paged twitter API r



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

```
Rate limit reached. Sleeping for: 896
Rate limit reached. Sleeping for: 896
Rate limit reached. Sleeping for: 896
Rate limit reached. Sleeping for: 895
Rate limit reached. Sleeping for: 895
1:15:03.018775
```

In [243…
```
# Import data from the files to utilize the twitter ids
mtFile=os.path.join(filepath, "mtrench_followers.txt")

with open(mtFile) as mt_file:
    reader = csv.reader(mt_file, delimiter="\t")
    mt = list(reader)
```

In [244…
```
# Import data from the files to utilize the twitter ids
nwtbFile=os.path.join(filepath, "NateWantsToBtl_followers.txt")
```

```
with open(nwtbFile) as nwtb_file:
    reader = csv.reader(nwtb_file, delimiter="\t")
    nwtb = list(reader)
```

Now that you have your follower ids, gather some information that we can use in future assignments on them. Using the `lookup_users` function, pull the user objects for your followers. These requests are limited to 900 per 15 minutes, but you can request 100 users at a time. At 90,000 users per 15 minutes, the rate limiter on pulls might be bandwidth rather than API limits.

Extract the following fields from the user object:

- screen_name
- name
- id
- location
- followers_count
- friends_count
- description

These can all be accessed via these names in the object. 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`.

In [245…
```
def chunks(lst, n):
    """Yield successive n-sized chunks from lst."""
    for i in range(0, len(lst), n):
        yield lst[i:i + n]
```

In [246…
```
#use the chunks method to split the followers of artist 1 into batches of 100.
mt_temp = mt[0]
mt_100 = list(chunks(mt_temp, 100))
```

In [258…
```
#use the chunks method to split the followers of artist 2 into batches of 100.
nwtb_temp = nwtb[0]
nwtb_100 = list(chunks(nwtb_temp, 100))
```

In [260…
```
# in this cell, do the following

# 1. Set up a data frame or dictionary to hold the user information
# 2. Use the `lookup_users` api function to pull sets of 100 users at a time
start = time.time()

mt_followers = []
nwtb_followers = []

for i in range(0,900):
    for users in api.lookup_users(user_id=mt_100[i]):
        mt_followers.append(users)
for i in range(900,1800):
    for users in api.lookup_users(user_id=mt_100[i]):
        mt_followers.append(users)
for i in range(1800, len(mt_100)):
    for users in api.lookup_users(user_id=mt_100[i]):
        mt_followers.append(users)

for i in range(0,900):
    for users in api.lookup_users(user_id=nwtb_100[i]):
        nwtb_followers.append(users)
```

```
for i in range(900,1800):
    for users in api.lookup_users(user_id=nwtb_100[i]):
        nwtb_followers.append(users)
for i in range(1800, len(nwtb_100)):
    for users in api.lookup_users(user_id=nwtb_100[i]):
        nwtb_followers.append(users)

print(time.time() - start)
```

```
3089.0448491573334
```

In [304…

```
# 3. Store the listed fields in your data frame or dictionary.
# 4. Write the user information in tab-delimited form to the follower data text file.

#create a dataframe storing the user information for marianas trench
mt_df = pd.DataFrame(data=[(user.screen_name, user.name, user.id, user.location,
                            user.followers_count, user.friends_count, user.description) fo
                     columns=['Screen Name', 'Name', 'ID', 'Location', 'Follower Count',

#create a dataframe storing the user information for nate wants to battle
nwtb_df = pd.DataFrame(data=[(user.screen_name, user.name, user.id, user.location,
                            user.followers_count, user.friends_count, user.description) fo
                     columns=['Screen Name', 'Name', 'ID', 'Location', 'Follower Count',

dataframes = {"mtrench": mt_df, "NateWantsToBtl": nwtb_df}

for artist, df in dataframes.items() :
    output_follower_file = artist + "_followers_data.txt"
    completeFFName=os.path.join(basepath+directory, output_follower_file)
    df.to_csv(completeFFName)
```

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. Here's an example of how you might do this.

In [ ]:

```
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)
clean_description
```

---

# 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 [72]:

```
artists = {'marianas trench':"https://www.azlyrics.com/m/marianastrench.html",
           'nate wants to battle':"https://www.azlyrics.com/n/natewantstobattle.html"}
base_url = 'https://www.azlyrics.com'
# we'll use this dictionary to hold both the artist name and the link on AZlyrics
```

# A Note on Rate Limiting

The lyrics site, www.azlyrics.com, does not have an explicit maximum on number of requests in any one time, but in our testing it appears that too many requests in too short a time will cause the site to stop returning lyrics pages. (Entertainingly, the page that gets returned seems to only have the song title to a Tom Jones song.)

Whenever you call `requests.get` to retrieve a page, put a `time.sleep(5 + 10*random.random())` on the next line. This will help you not to get blocked. If you *do* get blocked, which you can identify if the returned pages are not correct, just request a lyrics page through your browser. You'll be asked to perform a CAPTCHA and then your requests should start working again.

# Part 1: Finding Links to Songs Lyrics

That general artist page has a list of all songs for that artist with links to the individual song pages.

Q: Take a look at the `robots.txt` page on www.azlyrics.com. (You can read more about these pages here.) Is the scraping we are about to do allowed or disallowed by this page? How do you know?

A: The scraping is allowed since we are interested in the lyrics. We are not allowed to scrape the lyricsdb.

In [ ]:
```
#define a method to add key/value pairs to dictionary
def add_values_in_dict(sample_dict, key, list_of_values):
    ''' Append multiple values to a key in
        the given dictionary '''
    if key not in sample_dict:
        sample_dict[key] = list()
    sample_dict[key].extend(list_of_values)
    return sample_dict
```

In [103…
```
# Let's set up a dictionary of lists to hold our links
lyrics_pages = defaultdict(list)

for artist, artist_page in artists.items() :
    # 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, "html.parser")
    #create a temporary list to story links
    temp_list = []
    #extract the links and append them to the temp_list array
    for link in soup.find_all('a', href=re.compile('^/lyrics/')):
        temp_list.append(base_url + link.get('href'))

    #use the user-defined method to add the key/value pairs to lyrics_pages
    add_values_in_dict(lyrics_pages, artist, temp_list)
```

Let's make sure we have enough lyrics pages to scrape.

In [129…
```
for artist, lp in lyrics_pages.items() :
    assert(len(set(lp)) > 20)
```

In [130…
```
# 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() :
    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,
```

```
For marianas trench we have 76.
The full pull will take for this artist will take 0.21 hours.
For nate wants to battle we have 324.
The full pull will take for this artist will take 0.9 hours.
```

# Part 2: Pulling Lyrics

Now that we have the links to our lyrics pages, let's go scrape them! Here are the steps for this part.

1. Create an empty folder in our repo called "lyrics".
2. Iterate over the artists in `lyrics_pages`.
3. Create a subfolder in lyrics with the artist's name. For instance, if the artist was Cher you'd have `lyrics/cher/` in your repo.
4. Iterate over the pages.
5. Request the page and extract the lyrics from the returned HTML file using BeautifulSoup.
6. Use the function below, `generate_filename_from_url`, to create a filename based on the lyrics page, then write the lyrics to a text file with that name.

In [179…

```
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 = link.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 [175…

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

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

In [193…

```
#Set the root directory to the new lyrics folder created

root=os.getcwd()
new_directory='lyrics'

lyricspath=os.path.join(root, new_directory)
```

In [226…

```
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
    # 2. Iterate over the lyrics pages
    # 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
    # 4. Extract the title and lyrics from the page.
    # 5. Write out the title, two returns ('\n'), and the lyrics. Use `generate_filename
    #    to generate the filename.

    # Remember to pull at least 20 songs per artist. It may be fun to pull all the songs
    for link in lyrics_pages[artist] :
        #generate the file name for later use
        output_lyric_file = generate_filename_from_link(link)

        # Pull and store the title and lyrics after submitting the request
        sl = requests.get(link)
        time.sleep(5 + 10*random.random())

        #beautify the response
        souplyric = BeautifulSoup(sl.text, "html.parser")

        #save the title and lyrics to an object
        title = souplyric.find('div' , class_ = 'col-xs-12 col-lg-8 text-center').find_a
        lyrics = souplyric.find('div' , class_ = 'col-xs-12 col-lg-8 text-center').find_

        #create the artist folder if it doesn't exist and cwd to new folder
        if os.getcwd() != lyricspath:
            os.chdir(lyricspath)

        if not os.path.isdir(artist) :
            #shutil.rmtree("lyrics/")
            os.mkdir(artist)
        newpath = os.path.join(lyricspath, artist)

        # Write the Title and Lyrics to the output file in the `lyrics/artist` folder.
        finalName=os.path.join(newpath, output_lyric_file)
        f = open(finalName, "w", encoding="UTF-8")
        f.write(title + lyrics)
        f.close()

        #increment the total_pages counter
        total_pages+=1
```

In [227…
```
print(f"Total run time was {round((time.time() - start)/3600,2)} hours.")
```

```
Total run time was 1.17 hours.
```

---

# Evaluation

This assignment asks you to pull data from the Twitter API and scrape www.AZLyrics.com. After you have finished the above sections , run all the cells in this notebook. Print this to PDF and submit it, per the instructions.

In [290…
```
# Simple word extractor from Peter Norvig: https://norvig.com/spell-correct.html
def words(text):
    return re.findall(r'\w+', text.lower())
```

## Checking Twitter Data

The output from your Twitter API pull should be two files per artist, stored in files with formats like `cher_followers.txt` (a list of all follower IDs you pulled) and `cher_followers_data.txt`. These files should be in a folder named `twitter` within the repository directory. This code summarizes the information at a high level to help the instructor evaluate your work.

In [300…

```
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]} and {artist_handles[3]}.")
```

```
We see two artist handles: mtrench and NateWantsToBtl.
```

In [305…

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

    ids = open("twitter/" + follower_file,'r').readlines()

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

    with open("twitter/" + follower_data_file,'r') as infile :

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

        print(f"In the follower data file ({follower_data_file}) for {artist}, we have t
        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 0 in your follower file for mtrench, assuming a header row.
```

```
---------------------------------------------------------------------------
UnicodeDecodeError                        Traceback (most recent call last)
Input In [305], in <cell line: 1>()
      7 print(f"We see {len(ids)-1} in your follower file for {artist}, assuming a heade
r row.")
      9 with open("twitter/" + follower_data_file,'r') as infile :
     10
     11     # check the headers
---> 12     headers = infile.readline().split("\t")
     14     print(f"In the follower data file ({follower_data_file}) for {artist}, we ha
ve these columns:")
     15     print(" : ".join(headers))

File ~\anaconda3\envs\my-env-for-ads509\lib\encodings\cp1252.py:23, in IncrementalDecode
r.decode(self, input, final)
     22 def decode(self, input, final=False):
---> 23     return codecs.charmap_decode(input,self.errors,decoding_table)[0]

UnicodeDecodeError: 'charmap' codec can't decode byte 0x8d in position 262: character ma
ps to <undefined>
```

# Checking Lyrics

The output from your lyrics scrape should be stored in files located in this path from the directory: `/lyrics/[Artist Name]/[filename from URL]`. This code summarizes the information at a high level to help the instructor evaluate your work.

In [306…

```
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_word
```

```
For marianas trench we have 71 files.
For marianas trench we have roughly 26064 words, 1683 are unique.
For nate wants to battle we have 314 files.
For nate wants to battle we have roughly 97094 words, 4866 are unique.
```

In [ ]: