Group Name: AC
Group Members: Amy Chivavibul, Alexis Cirenza
Github Link: https://github.com/amychiv/SI206-FinalProject

Original Goals

In our initial project proposal, we planned to use the Spotify API to gather information on Amy's liked songs to generate a list of her top 100 most-played songs and count how many of them appear in "Google's Top Hummed Songs 2020" from the Billboard website using BeautifulSoup.

Goals Achieved

We used the Spotify and iTunes APIs, as well as the Seventeen website to get information on 60 popular TikTok songs. We used BeautifulSoup and regex to scrape artist names from the Seventeen article, and used those artists names as search queries on the Spotify and iTunes APIs. With the Spotify API, we were able to gather data on the top five songs for each artist. More specifically, we got each song's name, popularity, danceability, energy, liveness, and tempo. We calculated the average popularity of each artist using the Spotify popularity ratings of their top five songs, and found the top ten most popular artists based on their averages. We were also able to calculate correlations between danceability and popularity, liveness and popularity, tempo and popularity, and energy and popularity.

With the iTunes API, we also got the top five songs for each artist. However, these songs differed from the songs generated from the Spotify API, so we were able to count how many songs the iTunes API and Spotify API results had in common.

With the data we collected from both APIs and the Seventeen website, we successfully generated tables without duplicate strings by assigning each artist name an integer number, artist_id.

We were also able to create several interesting visualizations with the data we collected: a bar graph of the top ten artists and their average popularity, and four scatterplots that show danceability vs. popularity, liveness vs. popularity, tempo vs. popularity, and energy vs. popularity for each song result from the Spotify API.

Problems

Problem 1: The first problem we faced when beginning our project was discovering that "Google's Top Hummed Songs 2020," the original website we wanted to scrape data from, was no longer available unless we were Pro Subscribers to Billboard. As using data from this webpage was one of our original goals, this meant we needed to find a new webpage to gather data from, as well as reevaluate our initial goals for calculations.

Solution: We brainstormed and researched different websites that had a similar idea. In other words, we looked for websites that had some sort of "top" list that we could use in order to gather information to use with our API's. This led us to finding the Seventeen website, which we

thought would be interesting to collect data from being that TikTok is a very popular platform and many songs today become popular because they go viral on TikTok.

Problem 2: Another problem we faced was being unable to gather data from Amy's personal Spotify account. This was because we started using spotipy to access the Spotify API rather than accessing the API directly.

Solution: We used spotipy to gather different data than we initially intended. We were able to use the Spotify API to gather the top 5 songs that come up when searching an artist's name and for each of these songs we were able to gather audio features about them such as, danceability, energy, liveness, and tempo. This new information gathered inspired us to make more visualizations and do different calculations than we originally planned. For example, we were able to determine the top 10 artists based on who had the highest average popularity.

Problem 3: Another problem we faced was that the format of the song and artist names on the Seventeen website was not the same for each song. For example, some songs were formatted "Song Title by Artist Name," while others were formatted "Song Title - Artist Name."

Solution: Our solution for this problem was using regex. We used regex so that we could capture the artist that followed either 'by' or '--' or '-.'

Problem 4: One of the artists (5ifty3) in our list had a song name that was offensive, which we didn't want to include in our database.

Solution: We decided to remove the artist from our list using .remove()

Problem 5: Not all of the artists we initially loop through appear on the Spotify_Data table. This problem began after assigning each artist an id number so that there would be no duplicate strings. This is because the artists in the search queries only include the first artist's name, not the artists that come after "and" or "feat." Therefore, when inserting data into the Spotify_Data table using artist_ids rather than artist_names (if artist_tup[1] == artist_name:) the artist names in both cases are not exactly identical, leading to not all artists being included in the table.

Solutions: We tried to clean the artist's names and compare the two strings so that only the first few words had to match, however, this still caused errors in our code. After going to office hours, we came to the conclusion that it was okay if not all of the artists were included in the data table because we only needed a minimum of 100 rows, which would mean not all of the artists would be included anyways, and without the missing artists we still had at least 100 rows of data in our table.

Calculations

We had three different calculations. We calculated the average popularity for each artist using the popularity ratings from the Spotify API and used a bar chart to visualize average popularities for the ten most popular artists:

average_popularity.csv

```
TOP TEN TIKTOK ARTISTS AND AVERAGE POPULARITY ON SPOTIFY
Artist Name,Average Popularity
Olivia Rodrigo,90.125
Doja Cat,89.0
Lil Nas X,86.5
Drake,86.4
Pop Smoke,82.0
Britney Spears,77.83333333333333
Kid Cudi,77.5
Don Toliver,76.33333333333333
Kesha,73.0
Saweetie,69.25

```

Our next calculation was a count of how many top songs the iTunes and Spotify APIs had in common using the same artists as search queries:

common_songs.csv

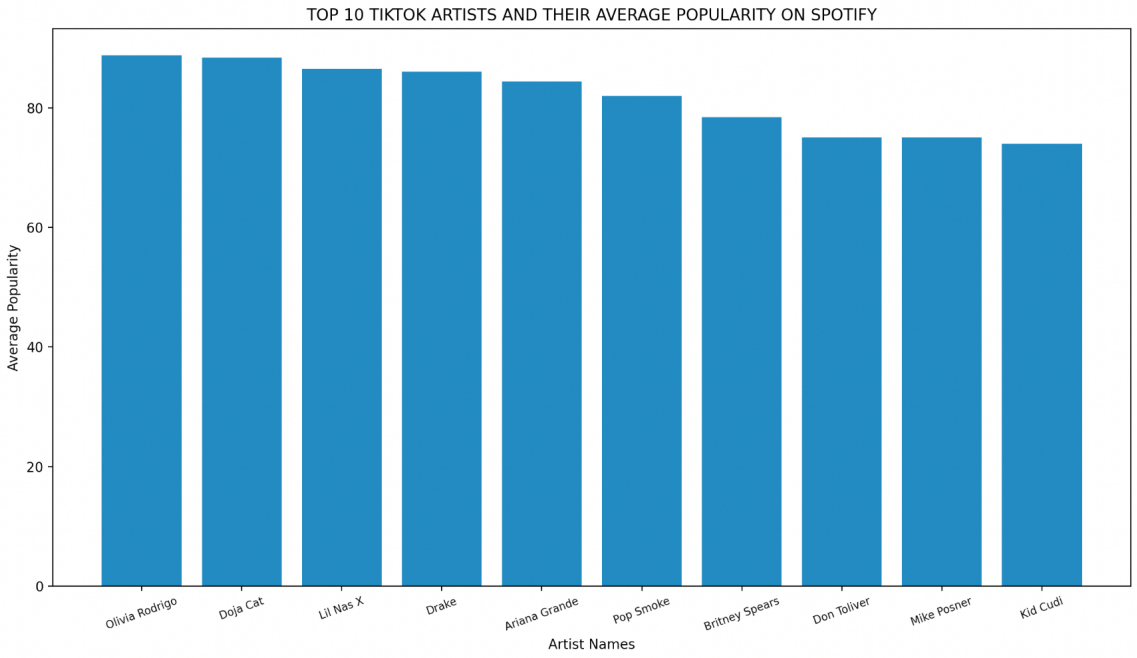
```
SONGS THAT APPEAR IN BOTH SPOTIFY AND ITUNES SEARCHES
Song Name,Artist Name
INDUSTRY BABY,Lil Nas X
Best Friend (feat. Doja Cat),Saweetie
good 4 u,Olivia Rodrigo
For The Night (feat. Lil Baby & DaBaby),Pop Smoke
After Party,Don Toliver
Number of songs that appear in both: 5
```

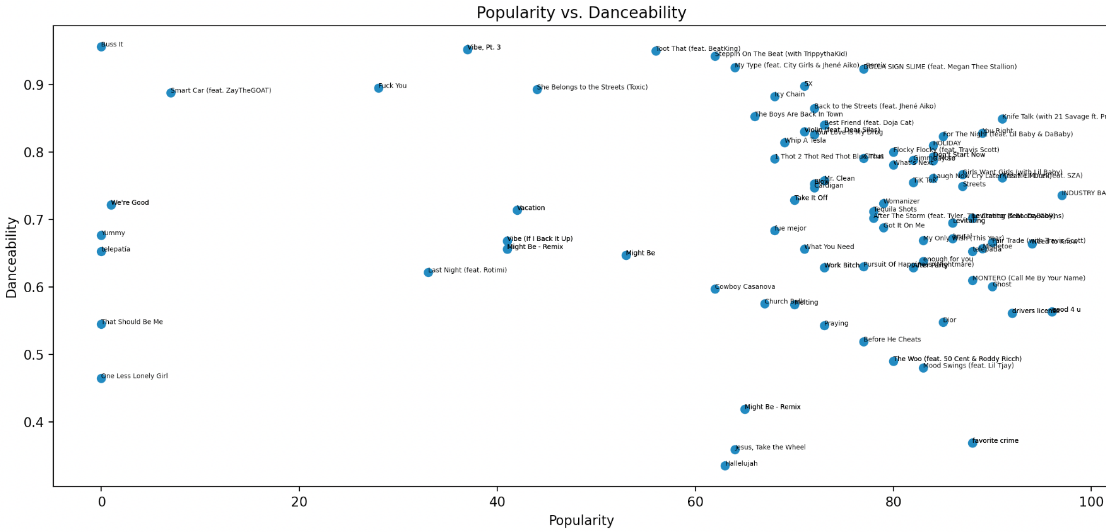
Our final calculation were the correlation coefficients between popularity and danceability, popularity and energy, popularity and liveness, and popularity and tempo - which we visualized with four different scatterplots:

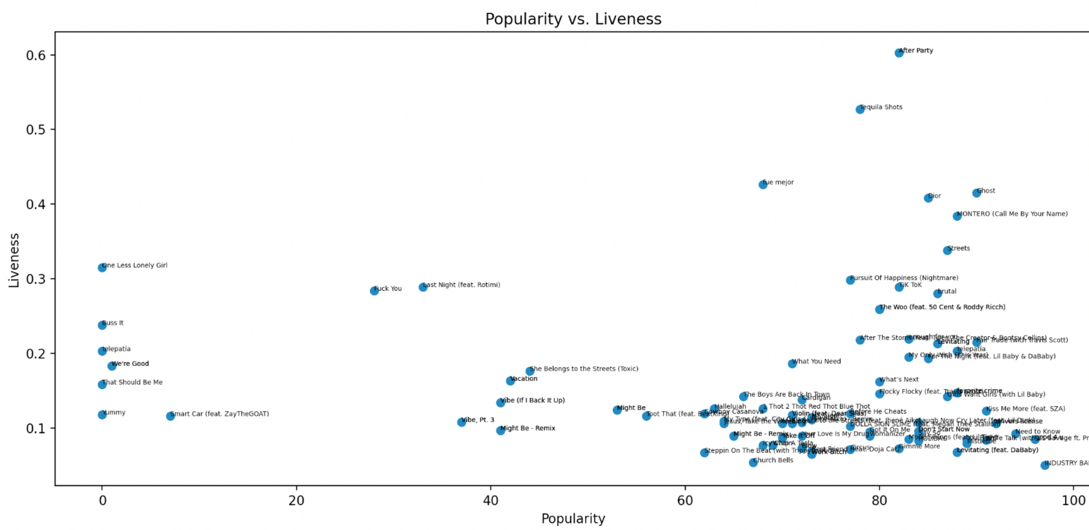
correlations.txt

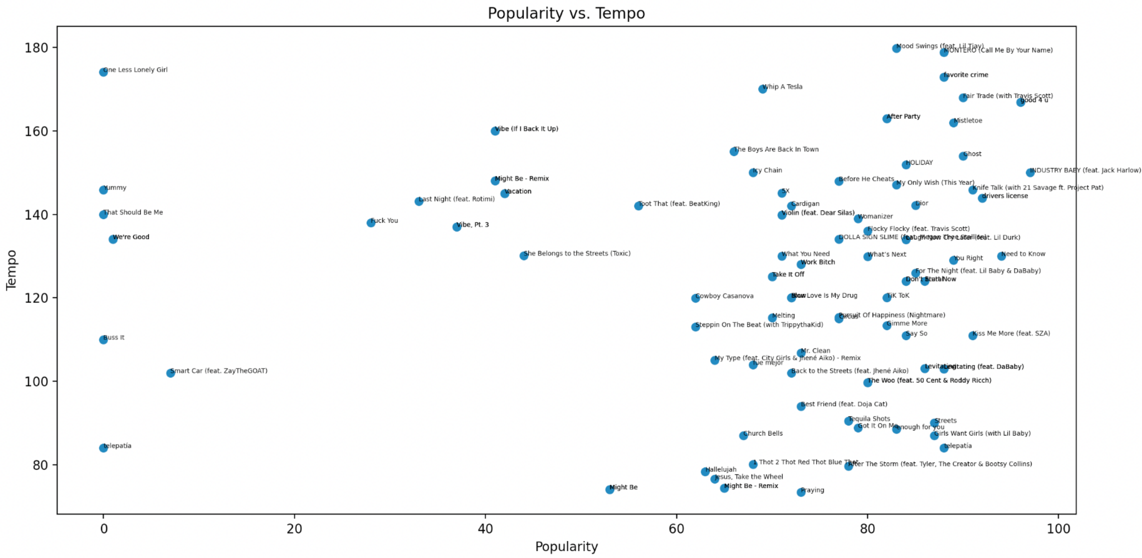
```
1   CORRELATION COEFFICIENTS
2   Popularity vs. Danceability: -0.12275628577783662
3   Popularity vs. Energy: 0.028029130747308954
4   Popularity vs. Liveness: -0.01377931866354649
5   Popularity vs. Tempo: 0.005824417228360462
6
```

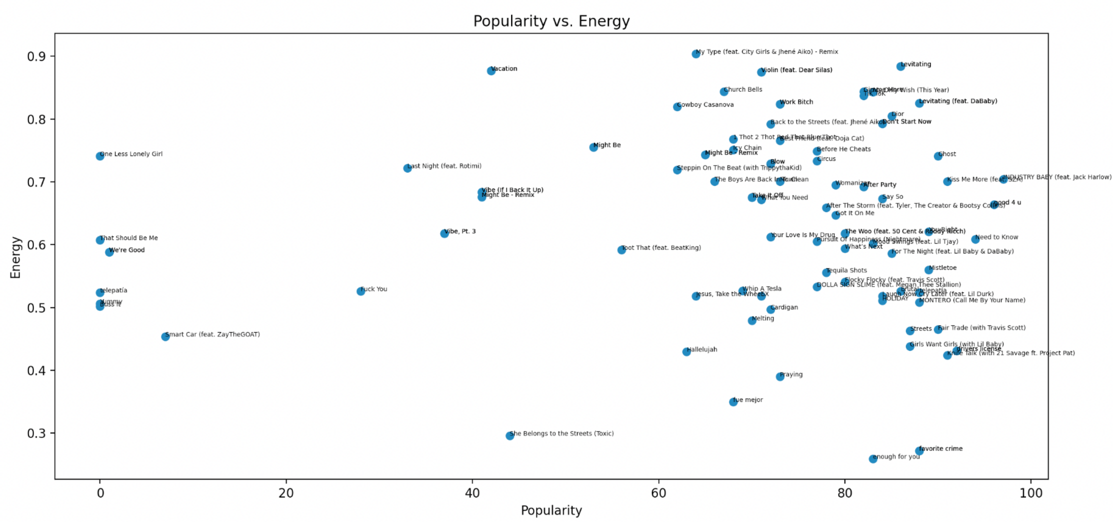
Visualizations




Popularity vs. Danceability
Danceability
Popularity
0.4
0.5
0.6
0.7
0.8
0.9
0
20
40
60
80
100


Popularity vs. Liveness
Liveness
Popularity
0.1
0.2
0.3
0.4
0.5
0.6
0
20
40
60
80
100


Popularity vs. Tempo
Tempo
Popularity
180
160
140
120
100
80
0
20
40
60
80
100


Popularity vs. Energy
Energy
Popularity
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0
20
40
60
80
100

Instructions for Running Code

Our files should be run in the following order:

spotify.py

Run this file until you get at least 100 songs in the Spotify_Data table in Music.db. This may take over 5 times. You will know when the file has run when “Done” is printed - this file takes some time to run each time.

itunes.py

Run this file 4+ times to get at least 100 songs in the iTunes table in Music.db. You will know when the file has run when “done” is printed.

calculations.py

Run this code once to generate three output files, average_popularity.csv, common_songs.csv, and correlations.txt. This code will also generate 5 different visualizations: one bar chart, and four scatterplots.

Documentation

spotify.py

```python

def get_all_artists():
    '''Takes no parameters. Returns list of unique artists from the Seventeen website using BeautifulSoup. Uses regex to ensure
    only the artist name is in the returned list.'''

def get_artists_cleaned(l):
    '''Takes in list of artists that get_all_artists() returns. Returns the same list of artists, but without the '(feat. featured_artist)'
    part of the artist name to be used in our search queries.'''

def get_track_data(artist_list):
    '''Takes in a list of artists from get_artists_cleaned(). Iterates through each artist in this list and uses the artist name as a search
    query with the Spotify API. Returns a list of tuples in the format: (track name, track id, artist name, track popularity)'''

def get_audio_features(track_id):
    '''Takes in a song's track_id, access the song's audio features from the Spotify API and returns a tuple of its ratings in the
    format: (dancability, energy, liveness, tempo)'''

def setUpDatabase(db_name):
    '''Takes in the database Music.db as a parameter, sets up the database, and returns cur and conn.'''

def setUpSpotifyTable(artist_list, cur, conn):
    '''Takes in artist_list (which is cleaned), cur, and conn as parameter. Uses each artist in artist_list to access data from
    get_track_data() and get_audio_features(), which is inserted into the Spotify table in the Music.db. This function limits results
    count to 25.'''

def setUpArtistTable(artist_list, cur, conn):
    '''Takes in artist_list (which is cleaned), cur, and conn as parameters. Assigns each artist name an artist_id, which is then inserted
    into the Spotify_Artist table'''

def main():
    '''Takes in no parameters and calls all of the functions above.'''
```

itunes.py

```
def itunes_search(searchterm):
    '''Takes in a search term as a parameter and returns its search results from the iTunes API'''

def get_all_artists():
    '''Takes no parameters. Returns list of unique artists from the Seventeen website using BeautifulSoup. Uses regex to ensure
    only the artist name is in the returned list.'''

def get_artists_cleaned(l):
    '''Takes in list of artists that get_all_artists() returns. Returns the same list of artists, but without the '(feat. featured_artist)'
    part of the artist name to be used in our search queries.'''

def get_top_5_tracks(artist):
    '''Takes in a artist name (cleaned) as a search query with the iTunes API and returns the first five unique song names that appear in
    the search results.'''

def setUpDatabase(db_name):
    '''Takes in the database Music.db as a parameter, sets up the database, and returns cur and conn.'''

def set_up_itunes_table(artist_list, cur, conn):
    '''Takes in artist_list (which is cleaned), cur, and conn as parameters. Inserts song name, artist name, and search rank from
    get_top_5_tracks() into the iTunes table. Limits how many songs are added to the table to 25.'''

def set_up_itunes_artist_table(artist_list, cur, conn):
    '''Takes in artist_list (which is cleaned), cur, and conn as parameters. Assigns each artist name an artist_id, which is then inserted
    into the iTunes_artist table'''

def main():
    '''Takes in no parameters and calls all of the functions above.'''
```

calculations.py

```
def get_top_10_artists_ids_and_popularity(cur, conn):
    '''Takes in cur and conn, and returns a list of tuples with artist_id and their average popularity. Only top 10 are returned.
    Selects all of the artist_ids from the Spotify_Data table and then loops through these ids in order to get the popularity of each of the artist's songs.
    The popularities are then averaged for each artist, appended to a list, and sorted in descending order.'''
def get_top_track(artist_id, cur, conn):
    '''Takes in artist_id, cur, conn, and returns a tuple of the artist_id and the corresponding artist's top track.'''
def join_artists(tup, cur, conn):
    '''Takes in a tuple (with artist_ids and track names), cur, conn and returns a tuple of the corresponding artist names and tracks.
    This is how the artist_id is used as a key between the different tables.'''
def compare_with_itunes(l2, cur, conn):
    '''Takes in a list of tuples (with artist name and tracks), cur, conn and returns a list of the songs that appear in both the iTunes and Spotify search.
    Selects the track name from iTunes table and compares it to each of the track names in the list which come from Spotify.
    If the song contains "(feat." only the beginning of the track names are compared.
    If the song does not contain feat. compare the full track name.
    After the track names are compared if they are the same, that track is appended to the list of songs that both Spotify and iTunes have in common.'''
def get_top_10_artist_names_and_pop(l, cur, conn):
    '''Takes in a list of tuples, cur, conn and returns a list of tuples with the artist names and popularity.'''
def get_itunes_songs_and_artists(l, cur, conn):
    '''Takes in a list of tuples, cur, conn and returns a list of tuples with the track name and corresponding artist's name.'''
def itunes_csv_out(l, file_name, cur, conn):
    '''Takes in a list, a file named "common_songs.csv", cur, conn and outputs text to a csv file.
    Calls the get_itunes_songs_and_artists function in order to collect the list of tuples of artists and songs that appeared in both iTunes and Spotify.
    Also keeps count of how many songs are in this list.
    Writes these names and tracks to a csv file, as well as the count of the number of songs that appear.'''
def csv_out(l, file_name, cur, conn):
    '''Takes in a list, a csv file named "average_popularity.csv", cur, conn and outputs text to a csv file.
    Calls the get_top_10_artist_names_and_pop function to get a list of tuples of the top 10 artist names and their average popularity.
    Writes each tuple to the csv file.'''
def bar_chart_visualization(l, cur, conn):
    '''Takes in a list, cur, conn and shows a visualization of a bar chart of the top 10 artist's average popularity.
    Calls the get_top_10_artist_names_and_pop function to get the data for the bar chart.'''
def scatterplot_visualization(cur, conn):
    '''Takes in cur, conn and returns the correlation coefficient of Popularity vs. Dance.
    Also creates a scatterplot of Popularity vs. Danceability'''
def scatterplot_energy_visualization(cur, conn):
    '''Takes in cur, conn and returns the correlation coefficient of Popularity vs. Energy.
    Also creates a scatterplot of Popularity vs. Energy.'''
```

```
38  def scatterplot_liveness_visualization(cur, conn):
39      '''Takes in cur, conn and returns the correlation coefficient of Popularity vs. Liveness.
40      Also creates a scatterplot of Popularity vs. Liveness'''
41  def scatterplot_tempo_visualization(cur, conn):
42      '''Takes in cur, conn and returns the correlation coefficient for Popularity vs. Tempo.
43      Also creates a scatterplot of Popularity vs. Tempo.'''
44  def write_correlations(filename, cur, conn):
45      '''Takes in a file named "correlations.txt", cur, conn and outputs to a txt file.
46      Calls the four scatterplot functions to display the scatterplots and get the correlations. These correlations are written to the txt file.'''
47  def main():
48      '''Takes in no parameters and calls the functions above in order to process the calculations using the "Music.db" database,
49      output the results of the calculations onto different files, as well as produce visualizations.'''
50
51
52
```

Resources Used

Date	Issue Description	Location of Resource	Result
11/30	Had trouble accessing Spotify API directly, so tried using spotipy library to access the Spotify API	https://spotipy.readthedocs.io/en/2.19.0/	Ended up using spotipy library to access the Spotify API instead
11/30	Was having trouble understanding how to use spotipy with the Spotify API	https://betterprogramming.pub/how-to-extract-any-artists-data-using-spotify-s-api-python-and-spotipy-4c079401bc37	Learned how to create a Spotify developer account, and how to use client ID and client secret to gain access to data
11/30	Needed to see what was being matched.	https://regex101.com/	Was successful in capturing all of the artist names.
12/2	Needed to figure out how to access information with an artist name as a search query	https://medium.com/@maxtingle/getting-started-with-spotifys-api-spotipy-197c3dc6353b	Found example code that we were able to adopt in our own project
12/7	Needed to remove an artist with an offensive song title	https://note.nkmk.me/en/python-list-clear-pop-remove-del/	Removed the artist using .remove()
12/8	Need documentation for the iTunes API	https://developer.apple.com/library/archive/documentation/AudioVideo/Conceptual/iTuneSearchAPI/S	Was able to access iTunes API using artist names as queries

		earching.html#//apple_ref/doc/uid/TP40017632-CH5-SW1	
12/6	Was getting an error message and didn't know what it meant.	https://stackoverflow.com/questions/16856647/sqlite3-programmingerror-incorrect-number-of-bindings-supplied-the-current-sta	Added a comma to make a tuple.
12/12	Wanted to calculate the correlation of Popularity vs. the audio features (4 of our visualizations).	https://realpython.com/numpy-scipy-pandas-correlation-python/	Used the lists that we use to create visualizations to calculate the correlation of each and output it to a txt file.
12/12	Wanted to add labels (song names) to the data points on our scatterplots	https://stackoverflow.com/questions/14432557/matplotlib-scatter-plot-with-different-text-at-each-data-point	Used enumerate() and .annotate() to add labels to the data points
12/12	Artist names overlapped in our bar chart, so we wanted to rotate the axis labels	https://www.kite.com/python/answers/how-to-rotate-axis-labels-in-matplotlib-in-python	Used plt.xticks(rotation=20) to alter rotation
12/12	Artist names were not aligning with the bar as we would have liked them to, and there was a lot of overlap when we annotated each point on the scatter plot.	https://stackoverflow.com/questions/12444716/how-do-i-set-the-figure-title-and-axes-labels-font-size-in-matplotlib	Changed the size of the font for the annotations, as well as the artist names. plt.xticks(fontsize=8) `ax1.annotate(label, (popularity_list[i], danceability_list[i]), fontsize = 5)`
12/12	Trying to label our axes and title our graph.	https://www.w3schools.com/python/matplotlib_labels.asp	Used to .title() and .xlabel() and .ylabel() to title each of our visualizations and label all of the axes.