**Spotify Examples Part-1**

**Example 1:** This example shows how to scrape the names of all albums Rihanna has from the Spotify and saves the results into a csv file.

Requirements:

1. You need to create an free app via the Spotify Developer tool. The app is a placeholder for the client id and client secret which we use to scrape data from the Spotify in the examples. <https://developer.spotify.com/dashboard>
2. <https://pypi.org/project/spotipy/> pip3 install spotipy
3. pip3 install pandas

Original Tutorial: <https://github.com/varunsatish/Projects/blob/master/A%20Guide%20to%20Data%20Scraping%20from%20Spotify%20(BeautifulSoup%20-%20Spotipy).ipynb>

import spotipy

import pandas

sp = spotipy.Spotify()

#I am creating an empty dictionary here

output = {"AlbumName":[]}

#in order to gain accesss to spotify API we needed to authorise client

from spotipy.oauth2 import SpotifyClientCredentials

#Put your own client id and secret below

client\_credentials\_manager = SpotifyClientCredentials(client\_id='Your App Client ID Goes Here', client\_secret='Your App Client Secret Goes Here')

sp = spotipy.Spotify(client\_credentials\_manager=client\_credentials\_manager)

#creates a list of all of Rihanna's albums

#https://open.spotify.com/artist/5pKCCKE2ajJHZ9KAiaK11H

rihanna = '5pKCCKE2ajJHZ9KAiaK11H'

results = sp.artist\_albums(rihanna, album\_type = 'album')

albums = results['items']

while results['next']:

results = sp.next(results)

albums.extend(results['items'])

for album in albums:

print(album['name'])

output['AlbumName'].append(album['name'])

results = pandas.DataFrame(output)

results.to\_csv('example1results.csv', index=True, index\_label="Index")

print("done")

**Example 2:** This example shows how to scrape artist, song, and album information from the Spotify Top Songs Global list. You can use any Spotify list in this example. The script saves the results to a csv file.

Original Tutorial: <https://towardsdatascience.com/extracting-song-data-from-the-spotify-api-using-python-b1e79388d50>

#https://towardsdatascience.com/extracting-song-data-from-the-spotify-api-using-python-b1e79388d50

#Search the top songs from the spotify global list

import spotipy

from spotipy.oauth2 import SpotifyClientCredentials

import pandas

#Put your App’s Client ID and Secret Below

client\_credentials\_manager = SpotifyClientCredentials(client\_id='Your App Client ID Goes Here', client\_secret='Your App Client Scret Goes Here')

sp = spotipy.Spotify(client\_credentials\_manager=client\_credentials\_manager)

#I am creating an empty dictionary here

output = {"ArtistName":[],"ArtistPopularity":[],"ArtistGenres":[],"SongName":[],'SongPopularity':[],'AlbumName':[]}

playlist\_link = "https://open.spotify.com/playlist/37i9dQZEVXbNG2KDcFcKOF?si=77d8f5cd51cd478d&nd=1"

playlist\_URI = playlist\_link.split("/")[-1].split("?")[0]

track\_uris = [x["track"]["uri"] for x in sp.playlist\_tracks(playlist\_URI)["items"]]

for track in sp.playlist\_tracks(playlist\_URI)["items"]:

# URI

track\_uri = track["track"]["uri"]

# Track name

track\_name = track["track"]["name"]

#print(track\_name)

output['SongName'].append(track\_name)

# Main Artist

artist\_uri = track["track"]["artists"][0]["uri"]

artist\_info = sp.artist(artist\_uri)

# Name, popularity, genre

artist\_name = track["track"]["artists"][0]["name"]

artist\_pop = artist\_info["popularity"]

output['ArtistPopularity'].append(artist\_pop )

artist\_genres = artist\_info["genres"]

#print(artist\_genres)

output['ArtistGenres'].append(artist\_genres)

#print(artist\_name)

output['ArtistName'].append(artist\_name)

#print(artist\_pop)

# Album

album = track["track"]["album"]["name"]

#print(album)

output['AlbumName'].append(album)

# Popularity of the track

track\_pop = track["track"]["popularity"]

# print(track\_pop)

output['SongPopularity'].append(track\_pop )

results = pandas.DataFrame(output)

results.to\_csv('example2results.csv', index=True, index\_label="Index")

print("done")

**Example 3:** This script scrapes the names of all songs and their associated album names of an artist. You can provide multiple artist names as an input for the script. The script saves the results into a csv file.

Original Tutorial: <https://www.kaggle.com/code/pavansanagapati/spotify-music-api-data-extraction-part1>

#https://www.kaggle.com/code/pavansanagapati/spotify-music-api-data-extraction-part1

import spotipy

from spotipy.oauth2 import SpotifyClientCredentials

import pandas

#Put your App’s Client ID and Secret Below

client\_id = "Your App Client ID Goes Here"

client\_secret = "Your app Client Secret Goes Here"

client\_credentials\_manager = SpotifyClientCredentials(client\_id=client\_id, client\_secret=client\_secret)

sp = spotipy.Spotify(client\_credentials\_manager=client\_credentials\_manager)

output = {'SongName':[],"AlbumName":[]}

#You can put multiple artists below

name = ['Rihanna']

result = sp.search(name)

result['tracks']['items'][1]['artists']

# Extract Artist's uri

artists\_uris = result['tracks']['items'][0]['artists'][0]['uri']

# Pull all of the artist's albums

artist\_albums = sp.artist\_albums(artists\_uris, album\_type='album')

# Store artist's albums' names' and uris in separate lists

artist\_album\_names = []

artist\_album\_uris = []

for i in range(len(artist\_albums['items'])):

artist\_album\_names.append(artist\_albums['items'][i]['name'])

artist\_album\_uris.append(artist\_albums['items'][i]['uri'])

#print(artist\_album\_names)

#print(artist\_album\_uris)

# Keep names and uris in same order to keep track of duplicate albums

#Extract all the songs from every album

def album\_songs(uri):

album = uri

spotify\_albums[album] = {}

#Create keys-values of empty lists inside nested dictionary for album

spotify\_albums[album]['album'] = []

spotify\_albums[album]['track\_number'] = []

spotify\_albums[album]['id'] = []

spotify\_albums[album]['name'] = []

spotify\_albums[album]['uri'] = []

#pull data on album tracks

tracks = sp.album\_tracks(album)

for n in range(len(tracks['items'])):

spotify\_albums[album]['album'].append(artist\_album\_names[album\_count])

spotify\_albums[album]['track\_number'].append(tracks['items'][n]['track\_number'])

spotify\_albums[album]['id'].append(tracks['items'][n]['id'])

spotify\_albums[album]['name'].append(tracks['items'][n]['name'])

spotify\_albums[album]['uri'].append(tracks['items'][n]['uri'])

#print(tracks['items'][n]['name'])

output['SongName'].append(tracks['items'][n]['name'])

#print(artist\_album\_names[album\_count])

output['AlbumName'].append(artist\_album\_names[album\_count])

spotify\_albums = {}

album\_count = 0

for i in artist\_album\_uris: #each album

album\_songs(i)

#print(str(artist\_album\_names[album\_count]) + " album songs has been added to spotify\_albums dictionary")

album\_count+=1 #Updates album count once all tracks have been added

results = pandas.DataFrame(output)

results.to\_csv('example3results.csv', index=True, index\_label="Index")

print("done")

**Example 4:** The script in this example scrapes more details about songs (e.g., tempo, danceability, acoustic…) together popularity values organized by album names. The script follows the same tutorial from the previous example.

Original Tutorial: <https://www.kaggle.com/code/pavansanagapati/spotify-music-api-data-extraction-part1>

#https://www.kaggle.com/code/pavansanagapati/spotify-music-api-data-extraction-part1

import spotipy

from spotipy.oauth2 import SpotifyClientCredentials

#Put Your App’s Client ID and Secret Below

client\_id = "Your App Client ID Goes Here"

client\_secret = "Your App Client Secret Goes Here"

client\_credentials\_manager = SpotifyClientCredentials(client\_id=client\_id, client\_secret=client\_secret)

sp = spotipy.Spotify(client\_credentials\_manager=client\_credentials\_manager)

#You can put multiple artists below

name = ['Rihanna']

result = sp.search(name)

result['tracks']['items'][1]['artists']

# Extract Artist's uri

artists\_uris = result['tracks']['items'][0]['artists'][0]['uri']

# Pull all of the artist's albums

artist\_albums = sp.artist\_albums(artists\_uris, album\_type='album')

# Store artist's albums' names' and uris in separate lists

artist\_album\_names = []

artist\_album\_uris = []

for i in range(len(artist\_albums['items'])):

artist\_album\_names.append(artist\_albums['items'][i]['name'])

artist\_album\_uris.append(artist\_albums['items'][i]['uri'])

#print(artist\_album\_names)

#print(artist\_album\_uris)

# Keep names and uris in same order to keep track of duplicate albums

#Extract all the songs from every album

def album\_songs(uri):

album = uri

spotify\_albums[album] = {}

#Create keys-values of empty lists inside nested dictionary for album

spotify\_albums[album]['album'] = []

spotify\_albums[album]['track\_number'] = []

spotify\_albums[album]['id'] = []

spotify\_albums[album]['name'] = []

spotify\_albums[album]['uri'] = []

#pull data on album tracks

tracks = sp.album\_tracks(album)

for n in range(len(tracks['items'])):

spotify\_albums[album]['album'].append(artist\_album\_names[album\_count])

spotify\_albums[album]['track\_number'].append(tracks['items'][n]['track\_number'])

spotify\_albums[album]['id'].append(tracks['items'][n]['id'])

spotify\_albums[album]['name'].append(tracks['items'][n]['name'])

spotify\_albums[album]['uri'].append(tracks['items'][n]['uri'])

spotify\_albums = {}

album\_count = 0

for i in artist\_album\_uris: #each album

album\_songs(i)

#print(str(artist\_album\_names[album\_count]) + " album songs has been added to spotify\_albums dictionary")

album\_count+=1 #Updates album count once all tracks have been added

#Extract audio features for each song

def audio\_features(album):

# Add new key-values to store audio features

spotify\_albums[album]['acousticness'] = []

spotify\_albums[album]['danceability'] = []

spotify\_albums[album]['energy'] = []

spotify\_albums[album]['instrumentalness'] = []

spotify\_albums[album]['liveness'] = []

spotify\_albums[album]['loudness'] = []

spotify\_albums[album]['speechiness'] = []

spotify\_albums[album]['tempo'] = []

spotify\_albums[album]['valence'] = []

spotify\_albums[album]['popularity'] = []

track\_count = 0

for track in spotify\_albums[album]['uri']:

# pull audio features per track

features = sp.audio\_features(track)

# Append to relevant key-value

spotify\_albums[album]['acousticness'].append(features[0]['acousticness'])

spotify\_albums[album]['danceability'].append(features[0]['danceability'])

spotify\_albums[album]['energy'].append(features[0]['energy'])

spotify\_albums[album]['instrumentalness'].append(features[0]['instrumentalness'])

spotify\_albums[album]['liveness'].append(features[0]['liveness'])

spotify\_albums[album]['loudness'].append(features[0]['loudness'])

spotify\_albums[album]['speechiness'].append(features[0]['speechiness'])

spotify\_albums[album]['tempo'].append(features[0]['tempo'])

spotify\_albums[album]['valence'].append(features[0]['valence'])

# popularity is stored elsewhere

pop = sp.track(track)

spotify\_albums[album]['popularity'].append(pop['popularity'])

track\_count += 1

#Now let loop through albums extracting the audio features.For this we will add a random delay every few albums to avoid sending too many requests at Spotify's API.

import time

import numpy as np

sleep\_min = 2

sleep\_max = 5

start\_time = time.time()

request\_count = 0

for i in spotify\_albums:

audio\_features(i)

request\_count+=1

if request\_count % 5 == 0:

print(str(request\_count) + " playlists completed")

time.sleep(np.random.uniform(sleep\_min, sleep\_max))

print('Loop #: {}'.format(request\_count))

print('Elapsed Time: {} seconds'.format(time.time() - start\_time))

#add the above spotify music data to a new dataframe

dic\_df = {}

dic\_df['album'] = []

dic\_df['track\_number'] = []

dic\_df['id'] = []

dic\_df['name'] = []

dic\_df['uri'] = []

dic\_df['acousticness'] = []

dic\_df['danceability'] = []

dic\_df['energy'] = []

dic\_df['instrumentalness'] = []

dic\_df['liveness'] = []

dic\_df['loudness'] = []

dic\_df['speechiness'] = []

dic\_df['tempo'] = []

dic\_df['valence'] = []

dic\_df['popularity'] = []

for album in spotify\_albums:

for feature in spotify\_albums[album]:

dic\_df[feature].extend(spotify\_albums[album][feature])

print(len(dic\_df['album']))

import pandas as pd

dataframe = pd.DataFrame.from\_dict(dic\_df)

#Spotify has a duplicate issue which can be addressed by removing all but the most popular songs.

print(len(dataframe))

final\_df = dataframe.sort\_values('popularity', ascending=False).drop\_duplicates('name').sort\_index()

print(len(final\_df))

final\_df.to\_csv("example4results.csv",index=True, index\_label="Index")

print("done")

**Example 5:** The next set of examples focus on scraping song lyrics. This example uses the python azapi library.

pip3 install azapi

<https://pypi.org/project/azapi/>

Original Tutorial: <https://github.com/elmoiv/azapi>

This example demonstrates how to scrape a song’s lyrics by a singer.

import azapi

API = azapi.AZlyrics('google', accuracy=0.5)

API.artist = 'Rihanna'

API.title = 'Diamonds'

API.getLyrics(save=True, ext='lrc')

print(API.title, API.artist)

print(API.lyrics)

**Example 6:** This example elaborates the previous example to scrape multiple songs’ lyrics from one singer. Pay attention to how I am making a loop here. The script saves the lyrics to a CSV file. Do not scrape many songs with this API otherwise, the service provider temporarily blocks your IP address!

import azapi

import pandas

API = azapi.AZlyrics('google', accuracy=0.5)

API.artist = 'Rihanna'

songlist=['Diamonds','Born Again']

output={"Song":[],"Lyrics":[]}

for song in songlist:

API.title = song

print("Song:",API.title,"Singer:",API.artist)

output['Song'].append(API.title)

API.getLyrics(save=True, ext='lrc')

print(API.lyrics)

output['Lyrics'].append(API.lyrics)

results = pandas.DataFrame(output)

results.to\_csv('example6results.csv', index=True, index\_label="Index")

print("done")

**Example 7:** This is just for your learning purposes. I built this script to scrape the lyrics of all of Rihanna’s songs. I got the song names from the example 3 results csv file. **DO NOT use this because**, the API will temporarily block your IP address for several hours. Check the logic of what I am doing in the source code. Particularly the if statement to prevent errors.

FYI: For learning oriented purposes, try giving some sleep timer like in example 4 and see if you can get the results without your IP address getting blocked.

import pandas

#Create an empty dataframe

input\_data\_in\_dataframe = pandas.DataFrame()

#Put the contents of the csv file into the dataframe

#FYI example 3 results csv file is about Rihanna

input\_data\_in\_dataframe = pandas.read\_csv("example3results.csv")

SongNames= input\_data\_in\_dataframe['SongName']

SongNamesList=list(SongNames)

#print(SongNamesList)

#I am now going to use this list that contains all song names of a singer to find each song's lyrics

import azapi

API = azapi.AZlyrics('google', accuracy=0.5)

#Since example 3 results is about Rihanna, I keep the artist name same below. Otherwise, change the artist name accordingly

API.artist = 'Rihanna'

#print(SongNamesList)

#Clean Song Names to prevent errors when searching for lyrics

separator = ' -'

cleanedsonglist=[]

for song in SongNamesList:

cleansong = song.split(separator, 1)[0]

cleanedsonglist.append(cleansong)

#print(cleanedsonglist)

output={"Song":[],"Lyrics":[]}

for song in (cleanedsonglist):

API.title = song

if (API.getLyrics(save=True, ext='lrc') !=''):

print("Song:", API.title)

print(API.lyrics)

output["Song"].append(API.title)

output["Lyrics"].append(API.lyrics)

results = pandas.DataFrame(output)

results.to\_csv('example7results.csv', index=True, index\_label="Index")

print("done")

**Example 8:** In order to prevent our IP address getting blocked, this example uses an access token from the lyrics genius.

pip3 install lyricsgenius

Original Tutorial:

<https://github.com/johnwmillr/LyricsGenius>

<https://medium.com/geekculture/easily-obtain-song-lyrical-information-in-python-5b2c85d4f589>

To create an access token, use

<https://genius.com/api-clients>

import lyricsgenius

import pandas

#Put your access token from Genius Below

genius = lyricsgenius.Genius(‘Your Genius Access Token Goes Here’)

artist = genius.search\_artist("Rihanna", max\_songs=3, sort="title", include\_features=True)

#print(artist.songs)

#song = artist.song("Diamonds")

#print(song.lyrics)

output={"Song":[],"Lyrics":[]}

songslist=['Work','Diamonds','The Monster','Wild Thoughts']

for listitem in (songslist):

print("Song:", listitem)

song=artist.song(listitem)

if (song!=''):

print(song.lyrics)

output['Song'].append(song)

#The first line in the output looks weird so I am deleting it below

cleanedlyrics = song.lyrics.split("\n",1)[1]

output['Lyrics'].append(cleanedlyrics)

results = pandas.DataFrame(output)

results.to\_csv('example8results.csv', index=True, index\_label="Index")

print("done")

**Example 9:** This example is similar to Example 7. In order to prevent our IP Address getting blocked, the script is using a client token from the lyrics genius. The problem is that when we scrape too many lyrics, the API gives a “time out” error and shuts down the script.

import pandas

#Create an empty dataframe

input\_data\_in\_dataframe = pandas.DataFrame()

#Put the contents of the csv file into the dataframe

#FYI example 3 results csv file is about Rihanna

input\_data\_in\_dataframe = pandas.read\_csv("example3results.csv")

SongNames= input\_data\_in\_dataframe['SongName']

SongNamesList=list(SongNames)

separator = ' -'

cleanedsonglist=[]

for song in SongNamesList:

cleansong = song.split(separator, 1)[0]

cleanedsonglist.append(cleansong)

import lyricsgenius

#Put your access token from Genius Below

genius = lyricsgenius.Genius(‘Your Genius Access Token Goes Here’)

artist = genius.search\_artist("Rihanna", max\_songs=3, sort="title", include\_features=True)

#print(artist.songs)

#song = artist.song("Diamonds")

#print(song.lyrics)

output={"Song":[],"Lyrics":[]}

for listitem in (cleanedsonglist):

print("Song:", listitem)

song=artist.song(listitem)

if (song is not None):

if(song.lyrics is not None):

print(song.lyrics)

output['Song'].append(song)

#The first line in the output looks weird so I am deleting it below

cleanedlyrics = song.lyrics.split("\n",1)[1]

output['Lyrics'].append(cleanedlyrics)

results = pandas.DataFrame(output)

results.to\_csv('example9results.csv', index=True, index\_label="Index")

print("done")