

Spotify Data Extraction With Python: A Guide

Contents

Chapter 1. Background.....	1
Chapter 2. Install the Required Tools.....	2
Chapter 3. Import Packages and Libraries.....	6
Chapter 4. Create a List of Album Names and Album URIs.....	7
Chapter 5. Extract All the Tracks From the Album List.....	9
Chapter 6. Extract Audio Features.....	11
Chapter 7. Transfrom the Dictionary into a Dataframe.....	13

Chapter 1. Background

Description:I am an avid Spotify user. My Spotify wrapped 2019 states that I spent 35,095 minutes listening to music on Spotify. This project originally started as an exploratory data analysis I did during my free time after having a discussion with a fellow Lorde fan about which Lorde songs we liked/disliked the most. I wanted to compare our subjective rankings to the actual data. As my interest in technical writing grew, I have decided to create a technical instruction of my data analysis project to demonstrate my technical authoring skills. This DITA guide was made using OxygenXML author.

The tools I used to make this guide:

- Oxygen XML Author - for creating, structuring, and publishing the document.

Chapter 2. Install the Required Tools

Here you can find the instructions to install the tools needed for this task. Skip this part if you have installed all the required tools.

Python

This is the first tool that you have to install. There are many ways to install Python to your machine. One way is to use the Python official installer.

1. Go to Python's [Official Website](#) .
2. Go to downloads page, and select the version appropriate for your machine.
3. Once the file has been downloaded, click on the .exe file to open the installer.
4. Click *continue* to proceed.
5. The next page will take you to the software license. Click *agree*.
6. Select the appropriate installation location.
7. Close the installer once it has finished installing Python.


Jupyter Notebook

Jupyter Notebook is a lightweight environment to run and test your code.

To install Jupyter Notebook:

1. Head to your computer's command-line
 - For Windows users:
 - Click on the search bar beside the Windows logo.
 - Type "Command Prompt".
 - For Mac Users:
 - Press F4 on your keyboard.
 - You will see a search bar on top of your screen.
 - Type "Terminal".
2. On the command prompt, run this line:

```
pip install jupyter
```

 **Note:** To make sure that your pip is up-to-date run this line on command-line

```
pip install --upgrade pip
```

3. To run Jupyter Notebook, run this line on command-line


```
jupyter notebook
```

Spotify Developer Account

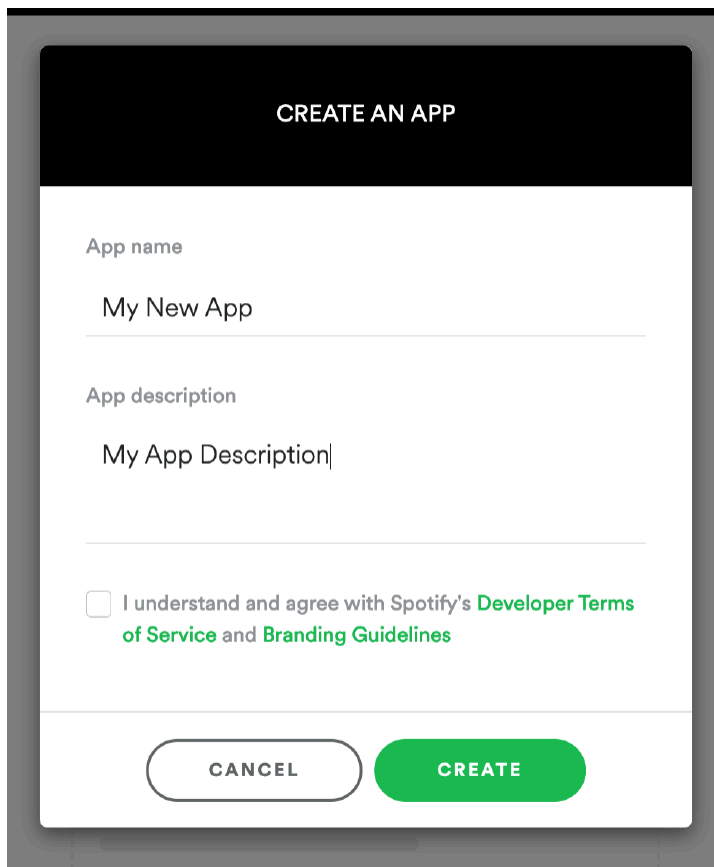
We will be using Spotify API to get our data. To get access to the Spotify API, we have to create a Spotify Developer account.

To create a Spotify Developer account:

1. Head to <https://developer.spotify.com/dashboard/>
2. Sign in with your Spotify account login info. If you do not have a Spotify account, Sign up for a Spotify account

 **Note:** You do not need a premium Spotify account to create a Spotify Developer account

3. Accept Spotify's Terms of Service
4. Next, click *Create an app*
5. Fill in your app name and a brief description of your app.



The screenshot shows the 'CREATE AN APP' form on the Spotify Developer Dashboard. The form has a black header with the text 'CREATE AN APP' in white. Below the header, there are two text input fields: 'App name' and 'App description'. The 'App name' field contains the text 'My New App' and the 'App description' field contains the text 'My App Description'. Below these fields, there is a checkbox labeled 'I understand and agree with Spotify's Developer Terms of Service and Branding Guidelines'. The checkbox is currently unchecked. At the bottom of the form, there are two buttons: a 'CANCEL' button and a 'CREATE' button. The 'CREATE' button is green and the 'CANCEL' button is white with a black border.

6. The website will take you to your dashboard containing your **Client ID**.
7. Click *Show client secret* to view your **Client Secret**.
8. Now that you have both Client ID and Client Secret, you can make request through Spotify API.

NumPy

Numpy is a scientific computing python package. We will use NumPy to work with arrays.

To install NumPy:

1. Head to your computer's command-line.
 - For Windows users:
 - Click on the search bar beside the Windows logo
 - Type "Command Prompt"
 - For Mac Users:
 - Press F4 on your keyboard
 - You will see a search bar on top of your screen
 - Type "Terminal"
2. Run this line:

```
pip install numpy
```

Pandas

Pandas is a python package to manipulate data.

To install Pandas:

1. Head to your computer's command-line.
 - For Windows users:
 - Click on the search bar beside the Windows logo
 - Type "Command Prompt"
 - For Mac Users:
 - Press F4 on your keyboard
 - You will see a search bar on top of your screen
 - Type "Terminal"
2. Run this line:

```
pip install pandas
```

Spotipy

Spotipy is a python library for the Spotify Web API.

To install Spotipy:

1. Head to your computer's command-line.

- For Windows users:
 - Click on the search bar beside the Windows logo
 - Type "Command Prompt"
- For Mac Users:
 - Press F4 on your keyboard
 - You will see a search bar on top of your screen
 - Type "Terminal"


2. Run this line

```
pip install spotipy
```


Chapter 3. Import Packages and Libraries

The first thing to do is to import required libraries and packages to our code.

1. Open Jupyter Notebook
2. We need to import all the packages and libraries needed for our code

 **Note:** See our installation guide (Step 0) to install the tools needed for this task.

```
import spotipy  
  
from spotipy.oauth2 import SpotifyClientCredentials  
  
pandas as pd  
  
numpy as np
```

3. Run the cell by pressing Shift + Enter/ Return or by pressing the  Run icon
4. Running this cell will not yield any output. As long as there is no error message, you can proceed to the next step.

Chapter 4. Create a List of Album Names and Album URIs

We are going to create a list of albums and album URIs. On this guide, we will be using Lorde's albums *Melodrama* and *Pure Heroine* as our examples.

1. Declare variables for Spotify API authentication.

We are going to declare the spotify Client ID and Client Secret Variables . Refer to your Client ID and Client Secret on your Spotify Developer Account dashboard and create these variables:

```
client_id = 'YOUR-CLIENT-ID'
```

```
client_secret = 'YOUR-CLIENT-SECRET'
```

 **Note:** Replace the texts between the parentheses with your Client ID and Client Secret.

Next, we are going to create these two variables below to authenticate requests to Spotify API.

```
client_credentials = SpotifyClientCredentials(client_id, client_secret)
```

```
sp = spotipy.Spotify(client_credentials_manager = client_credentials)
```

Run the cell to make sure no error occurs.

2. Create a list of album URIs to be included in the dataframe

Uniform Resource Indicator (URI) is a link that directs you to a spotify resource (artist/song/album/playlist, etc). Each resource has its own unique URI.

To access a spotify URI:

- Open Spotify and head to your album/artist/playlist of choice
- Click on the three dots under the album name
- Go to "Share" and place your cursor on "Copy Album Link"
- Press the Option/Alt key on your keyboard while your cursor is placed on "Copy Album Link"
- Click "Copy Album URI"

Create a list to store the URIs of the albums we want to include in our dataframe. Place the URI in a parenthesis. Separate each URI with a comma. For example:

```
album_uris = ['spotify:album:6rnzvZhe3PA57xKcKLrtJ6', 'spotify:album:4oCGmYsAQOWt2ACWtpNU']
```

3. Use Spotipy's **albums** function to get the album data from spotify. We are passing our list of album URIs to this function.

```
get_album = sp.albums(albums_uris)
```

4. Create empty lists to store album names and album URIs. We will append the data from Spotify to these lists.

```
lorde_album_names = []
```

```
lorde_album_uris = []
```

5. Create a loop that will go over each albums in the album URIs list and append the name of the album and the uri to the empty list made previously.

```
for i in range(len(get_album['albums'])):
    lorde_album_names.append(get_album['albums'][i]['name'])
    lorde_album_uris.append(get_album['albums'][i]['uri'])
```

The spotify data comes in JSON (JavaScript Object Notation). Inside the JSON data, the album name is stored under the key “name”. To get the name, we use the function **get_album['albums'][i]['name']**, which will be appended to the lorde_album_names. We repeat this process for the URIs.



Note: To see albums data from spotify, run this code below. The data from Spotify is stored in JSON format and you can see the values contained each album.

```
sp.albums(album_uris)
```

6. Run the cell to check if it works.

Chapter 5. Extract All the Tracks From the Album List

The goal of this step is to create a dictionary containing all of our albums, including each tracks and the relevant data that it contains. To do this, we will create a function that will extract and store each tracks from a list of albums.

1. Create an empty dictionary. This dictionary will be the place to store the album and album tracks.

```
lorde_albums={}
```

2. Create a function that, given an album URI, will extract and store each tracks, and its relevant data (album, track number, id, name, and uri) contained in the album to the dictionary we made in the previous step.

- a. Define a function **lorde_songs** that takes in one parameter 'album'.
- b. Inside the function, create lists that will store each track's album, track number, id, name and URI.
- c. Create an empty dictionary that stores all the lists from **step b**. Place this empty dictionary on the line after we define the function.

```
def lorde_songs(album):  
    lorde_albums[album] = {}  
    lorde_albums[album]['album'] = []  
    lorde_albums[album]['track_number'] = []  
    lorde_albums[album]['id'] = []  
    lorde_albums[album]['name'] = []  
    lorde_albums[album]['uri'] = []
```

- d. Using Spotify's **album_tracks** function, we will get the catalog of the album's track. Create a variable **tracks** to contain the **album_tracks** function.

Album_tracks gets the catalog of albums track, and it has 4 parameters. We will only use **album_id** for this task.

You can name the parameter passed to tracks to anything, because the function we are creating will be passed later to a loop that goes through a list of URIs.

```
tracks = sp.album_tracks(i)
```

- e. Next, create a for-loop that will go over a list of album and extracts the **album track number, id, name uri** to the empty list we made previously.

```
for t in range(len(tracks['items'])):  
    lorde_albums[album]['album'].append(lorde_album_names[album_count])  
    lorde_albums[album]['track_number'].append(tracks['items'][t]['track_number'])
```

```
lorde_albums[album]['id'].append(tracks['items'][t]['id'])


lorde_albums[album]['name'].append(tracks['items'][t]['name'])

lorde_albums[album]['uri'].append(tracks['items'][t]['uri'])
```

If we look at the spotify album JSON data , we can see that the information we want to extract is contained in an array with "items" as its key. Therefore, we pass **['items']** to the loop so we can get access to the data contained within it. We also want to append the values extracted, so we use **append** to add the values to the list we made in the previous step. We do this to every values we want to extract from each track. Now our function is complete.

3. Notice that in the loop inside our previously written function, we mention **album_count**. In the loop, **album_count** works so that our function knows which album from the album names list to extract the data from. For example, this snippet of code below appends the name of the first item in our album names list.

```
lorde_albums[album]['album'].append(lorde_album_names[0])
```

 **Note:** Because of zero-based indexing, the sequence starts at 0 instead of 1.

Since we want to reach all the albums in the album names list, we have to increase **album_count** by 1 for every iteration of the loop.

But first, we have to initiate the variable **album_count**, which contains the number 0.

```
album_count = 0
```

4. Now, we create the loop that will apply our function to each URI in the album URI list.

```
for i in lorde_album_uris:

    lorde_songs(i)

    album_count += 1
```

The variable we use in this function has to match the parameter name we pass to **tracks** (see step 2d). After each iteration of the loop, the value of album count increases by one.

5. Run the cell to make sure the code works.


Chapter 6. Extract Audio Features

The goal of this step is to add the audio features of each tracks to our dictionary.

SpotifyAPI allows us to access audio features embedded in each track.

We will be extracting 10 audio features :

- Acousticness
- Danceability
- Energy
- Instrumentalness
- Liveness
- Loudness
- Speechiness
- Tempo
- Valence
- Popularity

 **Note:** See <https://developer.spotify.com/documentation/web-api/reference/#category-tracks> for more information regarding the audio features

1. First, define a function **song_features** . **Song_features** takes in one parameter, album.
2. Create empty lists of all the features we want to extract.

```
def song_features(album):  
  
    lorde_albums[album]['acousticness'] = []  
  
    lorde_albums[album]['danceability'] = []  
  
    lorde_albums[album]['energy'] = []  
  
    lorde_albums[album]['instrumentalness'] = []  
  
    lorde_albums[album]['liveness'] = []  
  
    lorde_albums[album]['loudness'] = []  
  
    lorde_albums[album]['speechiness'] = []  
  
    lorde_albums[album]['tempo'] = []  
  
    lorde_albums[album]['valence'] = []  
  
    lorde_albums[album]['popularity'] = []
```

3. Inside the function, create a loop that goes through the URIs contained in the dictionary we made in the previous step.
4. To extract the features, we use to Spotipy's `audio_features` and `track` function.

```
for track in lorde_albums[album]['uri']:
    features = sp.audio_features(track)
    popularity = sp.track(track)
```

5. Inside the loop, we append each features to the appropriate list we made previously.

```
lorde_albums[album]['acousticness'].append(features[0]['acousticness'])
lorde_albums[album]['danceability'].append(features[0]['danceability'])
lorde_albums[album]['energy'].append(features[0]['energy'])
lorde_albums[album]['instrumentalness'].append(features[0]['instrumentalness'])
lorde_albums[album]['liveness'].append(features[0]['liveness'])
lorde_albums[album]['loudness'].append(features[0]['loudness'])
lorde_albums[album]['speechiness'].append(features[0]['speechiness'])
lorde_albums[album]['tempo'].append(features[0]['tempo'])
lorde_albums[album]['valence'].append(features[0]['valence'])
lorde_albums[album]['popularity'].append(popularity['popularity'])
```

6. Now that we have a function, we use a loop to apply the function to each tracks in our dictionary that contains all albums and tracks.

```
for i in lorde_albums:
    song_features(i)
```

7. Run the cell to make sure it works.

Chapter 7. Transfrom the Dictionary into a Dataframe

The last step is to transform our dictionary into a dataframe.

Notice that even though our dictionary contains all information needed, it is not well organized. We want to be able to see the name of each tracks and its associated features. Thus, we need to re-structure our dictionary.

1. Start by creating an empty dictionary

```
lorde_df={}
```

2. Create empty lists of the column names we want to have in our dataframe

```
lorde_df['name'] = []  
lorde_df['album'] = []  
lorde_df['track_number'] = []  
lorde_df['id'] = []  
lorde_df['uri'] = []  
lorde_df['acousticness'] = []  
lorde_df['danceability'] = []  
lorde_df['energy'] = []  
lorde_df['instrumentalness'] = []  
lorde_df['liveness'] = []  
lorde_df['loudness'] = []  
lorde_df['speechiness'] = []  
lorde_df['tempo'] = []  
lorde_df['valence'] = []  
lorde_df['popularity'] = []
```

3. Create a nested loop that will go through each albums and features in lorde albums

```
for album in lorde_albums:  
    for feature in lorde_albums[album]:  
        lorde_df[feature].extend(lorde_albums[album][feature])
```

4. Transform the dictionary into a dataframe and arrange the index to start from 1 instead of 0

```
df = pd.DataFrame.from_dict(lorde_df)  
df.index = np.arange(1, len(df) + 1)
```

5. Run this code to see the dataframe

```
df
```

6. To export the dataframe to a .csv, use pandas **to_csv** function

```
df.to_csv('lorde_discography.csv')
```

7. Congratulations! You just created a dataframe containing your chosen artists' spotify data!