Contents

[Help on Spotify in module spotipy.client object: 5](#_Toc82859163)

[Spotify(auth=None, requests\_session=True, client\_credentials\_manager=None, oauth\_manager=None, auth\_manager=None, proxies=None, requests\_timeout=5, status\_forcelist=None, retries=3, status\_retries=3, backoff\_factor=0.3, language=None) 5](#_Toc82859164)

[Example usage: 5](#_Toc82859165)

[Methods defined here: 5](#_Toc82859166)

[add\_to\_queue(self, uri, device\_id=None) 7](#_Toc82859167)

[album(self, album\_id) 7](#_Toc82859168)

[album\_tracks(self, album\_id, limit=50, offset=0, market=None) 7](#_Toc82859169)

[albums(self, albums) 8](#_Toc82859170)

[artist(self, artist\_id) 8](#_Toc82859171)

[artist\_albums(self, artist\_id, album\_type=None, country=None, limit=20, offset=0) 8](#_Toc82859172)

[artist\_related\_artists(self, artist\_id) 8](#_Toc82859173)

[artist\_top\_tracks(self, artist\_id, country='US') 9](#_Toc82859174)

[artists(self, artists) 9](#_Toc82859175)

[audio\_analysis(self, track\_id) 9](#_Toc82859176)

[audio\_features(self, tracks=[]) 9](#_Toc82859177)

[available\_markets(self) 9](#_Toc82859178)

[categories(self, country=None, locale=None, limit=20, offset=0) 10](#_Toc82859179)

[category(self, category\_id, country=None, locale=None) 10](#_Toc82859180)

[category\_playlists(self, category\_id=None, country=None, limit=20, offset=0) 10](#_Toc82859181)

[current\_playback(self, market=None, additional\_types=None) 11](#_Toc82859182)

[current\_user(self) 11](#_Toc82859183)

[current\_user\_follow\_playlist(self, playlist\_id) 11](#_Toc82859184)

[current\_user\_followed\_artists(self, limit=20, after=None) 12](#_Toc82859185)

[current\_user\_following\_artists(self, ids=None) 12](#_Toc82859186)

[current\_user\_following\_users(self, ids=None) 12](#_Toc82859187)

[current\_user\_playing\_track(self) 12](#_Toc82859188)

[current\_user\_playlists(self, limit=50, offset=0) 12](#_Toc82859189)

[current\_user\_recently\_played(self, limit=50, after=None, before=None) 13](#_Toc82859190)

[current\_user\_saved\_albums(self, limit=20, offset=0, market=None) 13](#_Toc82859191)

[current\_user\_saved\_albums\_add(self, albums=[]) 13](#_Toc82859192)

[current\_user\_saved\_albums\_contains(self, albums=[]) 14](#_Toc82859193)

[current\_user\_saved\_albums\_delete(self, albums=[]) 14](#_Toc82859194)

[current\_user\_saved\_episodes(self, limit=20, offset=0, market=None) 14](#_Toc82859195)

[current\_user\_saved\_episodes\_add(self, episodes=None) 14](#_Toc82859196)

[current\_user\_saved\_episodes\_contains(self, episodes=None) 15](#_Toc82859197)

[current\_user\_saved\_episodes\_delete(self, episodes=None) 15](#_Toc82859198)

[current\_user\_saved\_shows(self, limit=20, offset=0, market=None) 15](#_Toc82859199)

[current\_user\_saved\_shows\_add(self, shows=[]) 15](#_Toc82859200)

[current\_user\_saved\_shows\_contains(self, shows=[]) 16](#_Toc82859201)

[current\_user\_saved\_shows\_delete(self, shows=[]) 16](#_Toc82859202)

[current\_user\_saved\_tracks(self, limit=20, offset=0, market=None) 16](#_Toc82859203)

[current\_user\_saved\_tracks\_add(self, tracks=None) 16](#_Toc82859204)

[current\_user\_saved\_tracks\_contains(self, tracks=None) 17](#_Toc82859205)

[current\_user\_saved\_tracks\_delete(self, tracks=None) 17](#_Toc82859206)

[current\_user\_top\_artists(self, limit=20, offset=0, time\_range='medium\_term') 17](#_Toc82859207)

[current\_user\_top\_tracks(self, limit=20, offset=0, time\_range='medium\_term') 17](#_Toc82859208)

[current\_user\_unfollow\_playlist(self, playlist\_id) 18](#_Toc82859209)

[currently\_playing(self, market=None, additional\_types=None) 18](#_Toc82859210)

[devices(self) 18](#_Toc82859211)

[episode(self, episode\_id, market=None) 18](#_Toc82859212)

[episodes(self, episodes, market=None) 19](#_Toc82859213)

[featured\_playlists(self, locale=None, country=None, timestamp=None, limit=20, offset=0) 19](#_Toc82859214)

[me(self) 20](#_Toc82859215)

[new\_releases(self, country=None, limit=20, offset=0) 20](#_Toc82859216)

[next(self, result) 20](#_Toc82859217)

[next\_track(self, device\_id=None) 20](#_Toc82859218)

[pause\_playback(self, device\_id=None) 21](#_Toc82859219)

[playlist(self, playlist\_id, fields=None, market=None, additional\_types=('track',)) 21](#_Toc82859220)

[playlist\_add\_items(self, playlist\_id, items, position=None) 21](#_Toc82859221)

[playlist\_change\_details(self, playlist\_id, name=None, public=None, collaborative=None, description=None) 22](#_Toc82859222)

[playlist\_cover\_image(self, playlist\_id) 22](#_Toc82859223)

[playlist\_is\_following(self, playlist\_id, user\_ids) 22](#_Toc82859224)

[playlist\_items(self, playlist\_id, fields=None, limit=100, offset=0, market=None, additional\_types=('track', 'episode')) 22](#_Toc82859225)

[playlist\_remove\_all\_occurrences\_of\_items(self, playlist\_id, items, snapshot\_id=None) 23](#_Toc82859226)

[playlist\_remove\_specific\_occurrences\_of\_items(self, playlist\_id, items, snapshot\_id=None) 23](#_Toc82859227)

[playlist\_reorder\_items(self, playlist\_id, range\_start, insert\_before, range\_length=1, snapshot\_id=None) 24](#_Toc82859228)

[playlist\_replace\_items(self, playlist\_id, items) 24](#_Toc82859229)

[playlist\_tracks(self, playlist\_id, fields=None, limit=100, offset=0, market=None, additional\_types=('track',)) 24](#_Toc82859230)

[playlist\_upload\_cover\_image(self, playlist\_id, image\_b64) 25](#_Toc82859231)

[previous(self, result) 25](#_Toc82859232)

[previous\_track(self, device\_id=None) 25](#_Toc82859233)

[recommendation\_genre\_seeds(self) 25](#_Toc82859234)

[recommendations(self, seed\_artists=None, seed\_genres=None, seed\_tracks=None, limit=20, country=None, \*\*kwargs) 25](#_Toc82859235)

[repeat(self, state, device\_id=None) 26](#_Toc82859236)

[search(self, q, limit=10, offset=0, type='track', market=None) 26](#_Toc82859237)

[search\_markets(self, q, limit=10, offset=0, type='track', markets=None, total=None) 27](#_Toc82859238)

[seek\_track(self, position\_ms, device\_id=None) 28](#_Toc82859239)

[set\_auth(self, auth) 28](#_Toc82859240)

[show(self, show\_id, market=None) 28](#_Toc82859241)

[show\_episodes(self, show\_id, limit=50, offset=0, market=None) 28](#_Toc82859242)

[shows(self, shows, market=None) 29](#_Toc82859243)

[shuffle(self, state, device\_id=None) 29](#_Toc82859244)

[start\_playback(self, device\_id=None, context\_uri=None, uris=None, offset=None, position\_ms=None) 29](#_Toc82859245)

[track(self, track\_id, market=None) 30](#_Toc82859246)

[tracks(self, tracks, market=None) 30](#_Toc82859247)

[transfer\_playback(self, device\_id, force\_play=True) 31](#_Toc82859248)

[user(self, user) 31](#_Toc82859249)

[user\_follow\_artists(self, ids=[]) 31](#_Toc82859250)

[user\_follow\_users(self, ids=[]) 31](#_Toc82859251)

[user\_playlist(self, user, playlist\_id=None, fields=None, market=None) 31](#_Toc82859252)

[user\_playlist\_add\_tracks(self, user, playlist\_id, tracks, position=None) 31](#_Toc82859253)

[user\_playlist\_change\_details(self, user, playlist\_id, name=None, public=None, collaborative=None, description=None) 32](#_Toc82859254)

[user\_playlist\_create(self, user, name, public=True, collaborative=False, description='') 32](#_Toc82859255)

[user\_playlist\_follow\_playlist(self, playlist\_owner\_id, playlist\_id) 32](#_Toc82859256)

[user\_playlist\_is\_following(self, playlist\_owner\_id, playlist\_id, user\_ids) 32](#_Toc82859257)

[user\_playlist\_remove\_all\_occurrences\_of\_tracks(self, user, playlist\_id, tracks, snapshot\_id=None) 33](#_Toc82859258)

[user\_playlist\_remove\_specific\_occurrences\_of\_tracks(self, user, playlist\_id, tracks, snapshot\_id=None) 33](#_Toc82859259)

[user\_playlist\_reorder\_tracks(self, user, playlist\_id, range\_start, insert\_before, range\_length=1, snapshot\_id=None) 33](#_Toc82859260)

[user\_playlist\_replace\_tracks(self, user, playlist\_id, tracks) 34](#_Toc82859261)

[user\_playlist\_tracks(self, user=None, playlist\_id=None, fields=None, limit=100, offset=0, market=None) 34](#_Toc82859262)

[user\_playlist\_unfollow(self, user, playlist\_id) 34](#_Toc82859263)

[user\_playlists(self, user, limit=50, offset=0) 34](#_Toc82859264)

[user\_unfollow\_artists(self, ids=[]) 35](#_Toc82859265)

[user\_unfollow\_users(self, ids=[]) 35](#_Toc82859266)

[volume(self, volume\_percent, device\_id=None) 35](#_Toc82859267)

# Help on Spotify in module spotipy.client object:

class Spotify(builtins.object)

## Spotify(auth=None, requests\_session=True, client\_credentials\_manager=None, oauth\_manager=None, auth\_manager=None, proxies=None, requests\_timeout=5, status\_forcelist=None, retries=3, status\_retries=3, backoff\_factor=0.3, language=None)

### Example usage:

import spotipy

urn = 'spotify:artist:3jOstUTkEu2JkjvRdBA5Gu'

sp = spotipy.Spotify()

artist = sp.artist(urn)

print(artist)

user = sp.user('plamere')

print(user)

## Methods defined here:

\_\_del\_\_(self)

Make sure the connection (pool) gets closed

\_\_init\_\_(self, auth=None, requests\_session=True, client\_credentials\_manager=None, oauth\_manager=None, auth\_manager=None, proxies=None, requests\_timeout=5, status\_forcelist=None, retries=3, status\_retries=3, backoff\_factor=0.3, language=None)

Creates a Spotify API client.

:param auth: An access token (optional)

:param requests\_session:

A Requests session object or a truthy value to create one.

A falsy value disables sessions.

It should generally be a good idea to keep sessions enabled

for performance reasons (connection pooling).

:param client\_credentials\_manager:

SpotifyClientCredentials object

:param oauth\_manager:

SpotifyOAuth object

:param auth\_manager:

SpotifyOauth, SpotifyClientCredentials,

or SpotifyImplicitGrant object

:param proxies:

Definition of proxies (optional).

See Requests doc <https://2.python-requests.org/en/master/user/advanced/#proxies>

:param requests\_timeout:

Tell Requests to stop waiting for a response after a given

number of seconds

:param status\_forcelist:

Tell requests what type of status codes retries should occur on

:param retries:

Total number of retries to allow

:param status\_retries:

Number of times to retry on bad status codes

:param backoff\_factor:

A backoff factor to apply between attempts after the second try

See urllib3 <https://urllib3.readthedocs.io/en/latest/reference/urllib3.util.html>

:param language:

The language parameter advertises what language the user prefers to see.

See ISO-639 language code: <https://www.loc.gov/standards/iso639-2/php/code_list.php>

## add\_to\_queue(self, uri, device\_id=None)

Adds a song to the end of a user's queue

If device A is currently playing music and you try to add to the queue

and pass in the id for device B, you will get a

'Player command failed: Restriction violated' error

I therefore reccomend leaving device\_id as None so that the active device is targeted

:param uri: song uri, id, or url

:param device\_id:

the id of a Spotify device.

If None, then the active device is used.

## album(self, album\_id)

returns a single album given the album's ID, URIs or URL

Parameters:

- album\_id - the album ID, URI or URL

## album\_tracks(self, album\_id, limit=50, offset=0, market=None)

Get Spotify catalog information about an album's tracks

Parameters:

- album\_id - the album ID, URI or URL

- limit - the number of items to return

- offset - the index of the first item to return

- market - an ISO 3166-1 alpha-2 country code.

## albums(self, albums)

returns a list of albums given the album IDs, URIs, or URLs

Parameters:

- albums - a list of album IDs, URIs or URLs

## artist(self, artist\_id)

returns a single artist given the artist's ID, URI or URL

Parameters:

- artist\_id - an artist ID, URI or URL

## artist\_albums(self, artist\_id, album\_type=None, country=None, limit=20, offset=0)

Get Spotify catalog information about an artist's albums

Parameters:

- artist\_id - the artist ID, URI or URL

- album\_type - 'album', 'single', 'appears\_on', 'compilation'

- country - limit the response to one particular country.

- limit - the number of albums to return

- offset - the index of the first album to return

## artist\_related\_artists(self, artist\_id)

Get Spotify catalog information about artists similar to an

identified artist. Similarity is based on analysis of the

Spotify community's listening history.

Parameters:

- artist\_id - the artist ID, URI or URL

## artist\_top\_tracks(self, artist\_id, country='US')

Get Spotify catalog information about an artist's top 10 tracks

by country.

Parameters:

- artist\_id - the artist ID, URI or URL

- country - limit the response to one particular country.

## artists(self, artists)

returns a list of artists given the artist IDs, URIs, or URLs

Parameters:

- artists - a list of artist IDs, URIs or URLs

## audio\_analysis(self, track\_id)

Get audio analysis for a track based upon its Spotify ID

Parameters:

- track\_id - a track URI, URL or ID

## audio\_features(self, tracks=[])

Get audio features for one or multiple tracks based upon their Spotify IDs

Parameters:

- tracks - a list of track URIs, URLs or IDs, maximum: 100 ids

## available\_markets(self)

Get the list of markets where Spotify is available.

Returns a list of the countries in which Spotify is available, identified by their

ISO 3166-1 alpha-2 country code with additional country codes for special territories.

## categories(self, country=None, locale=None, limit=20, offset=0)

Get a list of categories

Parameters:

- country - An ISO 3166-1 alpha-2 country code.

- locale - The desired language, consisting of an ISO 639

language code and an ISO 3166-1 alpha-2 country code, joined

by an underscore.

- limit - The maximum number of items to return. Default: 20.

Minimum: 1. Maximum: 50

- offset - The index of the first item to return. Default: 0

(the first object). Use with limit to get the next set of

items.

## category(self, category\_id, country=None, locale=None)

Get info about a category

Parameters:

- category\_id - The Spotify category ID for the category.

- country - An ISO 3166-1 alpha-2 country code.

- locale - The desired language, consisting of an ISO 639

language code and an ISO 3166-1 alpha-2 country code, joined

by an underscore.

## category\_playlists(self, category\_id=None, country=None, limit=20, offset=0)

Get a list of playlists for a specific Spotify category

Parameters:

- category\_id - The Spotify category ID for the category.

- country - An ISO 3166-1 alpha-2 country code.

- limit - The maximum number of items to return. Default: 20.

Minimum: 1. Maximum: 50

- offset - The index of the first item to return. Default: 0

(the first object). Use with limit to get the next set of

items.

## current\_playback(self, market=None, additional\_types=None)

Get information about user's current playback.

Parameters:

- market - an ISO 3166-1 alpha-2 country code.

- additional\_types - `episode` to get podcast track information

## current\_user(self)

Get detailed profile information about the current user.

An alias for the 'me' method.

## current\_user\_follow\_playlist(self, playlist\_id)

Add the current authenticated user as a follower of a playlist.

Parameters:

- playlist\_id - the id of the playlist

## current\_user\_followed\_artists(self, limit=20, after=None)

Gets a list of the artists followed by the current authorized user

Parameters:

- limit - the number of artists to return

- after - the last artist ID retrieved from the previous

request

## current\_user\_following\_artists(self, ids=None)

Check if the current user is following certain artists

Returns list of booleans respective to ids

Parameters:

- ids - a list of artist URIs, URLs or IDs

## current\_user\_following\_users(self, ids=None)

Check if the current user is following certain artists

Returns list of booleans respective to ids

Parameters:

- ids - a list of user URIs, URLs or IDs

## current\_user\_playing\_track(self)

Get information about the current users currently playing track.

## current\_user\_playlists(self, limit=50, offset=0)

Get current user playlists without required getting his profile

Parameters:

- limit - the number of items to return

- offset - the index of the first item to return

## current\_user\_recently\_played(self, limit=50, after=None, before=None)

Get the current user's recently played tracks

Parameters:

- limit - the number of entities to return

- after - unix timestamp in milliseconds. Returns all items

after (but not including) this cursor position.

Cannot be used if before is specified.

- before - unix timestamp in milliseconds. Returns all items

before (but not including) this cursor position.

Cannot be used if after is specified

## current\_user\_saved\_albums(self, limit=20, offset=0, market=None)

Gets a list of the albums saved in the current authorized user's

"Your Music" library

Parameters:

- limit - the number of albums to return (MAX\_LIMIT=50)

- offset - the index of the first album to return

- market - an ISO 3166-1 alpha-2 country code.

## current\_user\_saved\_albums\_add(self, albums=[])

Add one or more albums to the current user's

"Your Music" library.

Parameters:

- albums - a list of album URIs, URLs or IDs

## current\_user\_saved\_albums\_contains(self, albums=[])

Check if one or more albums is already saved in

the current Spotify user’s “Your Music” library.

Parameters:

- albums - a list of album URIs, URLs or IDs

## current\_user\_saved\_albums\_delete(self, albums=[])

Remove one or more albums from the current user's

"Your Music" library.

Parameters:

- albums - a list of album URIs, URLs or IDs

## current\_user\_saved\_episodes(self, limit=20, offset=0, market=None)

Gets a list of the episodes saved in the current authorized user's

"Your Music" library

Parameters:

- limit - the number of episodes to return

- offset - the index of the first episode to return

- market - an ISO 3166-1 alpha-2 country code

## current\_user\_saved\_episodes\_add(self, episodes=None)

Add one or more episodes to the current user's

"Your Music" library.

Parameters:

- episodes - a list of episode URIs, URLs or IDs

## current\_user\_saved\_episodes\_contains(self, episodes=None)

Check if one or more episodes is already saved in

the current Spotify user’s “Your Music” library.

Parameters:

- episodes - a list of episode URIs, URLs or IDs

## current\_user\_saved\_episodes\_delete(self, episodes=None)

Remove one or more episodes from the current user's

"Your Music" library.

Parameters:

- episodes - a list of episode URIs, URLs or IDs

## current\_user\_saved\_shows(self, limit=20, offset=0, market=None)

Gets a list of the shows saved in the current authorized user's

"Your Music" library

Parameters:

- limit - the number of shows to return

- offset - the index of the first show to return

- market - an ISO 3166-1 alpha-2 country code

## current\_user\_saved\_shows\_add(self, shows=[])

Add one or more albums to the current user's

"Your Music" library.

Parameters:

- shows - a list of show URIs, URLs or IDs

## current\_user\_saved\_shows\_contains(self, shows=[])

Check if one or more shows is already saved in

the current Spotify user’s “Your Music” library.

Parameters:

- shows - a list of show URIs, URLs or IDs

## current\_user\_saved\_shows\_delete(self, shows=[])

Remove one or more shows from the current user's

"Your Music" library.

Parameters:

- shows - a list of show URIs, URLs or IDs

## current\_user\_saved\_tracks(self, limit=20, offset=0, market=None)

Gets a list of the tracks saved in the current authorized user's

"Your Music" library

Parameters:

- limit - the number of tracks to return

- offset - the index of the first track to return

- market - an ISO 3166-1 alpha-2 country code

## current\_user\_saved\_tracks\_add(self, tracks=None)

Add one or more tracks to the current user's

"Your Music" library.

Parameters:

- tracks - a list of track URIs, URLs or IDs

## current\_user\_saved\_tracks\_contains(self, tracks=None)

Check if one or more tracks is already saved in

the current Spotify user’s “Your Music” library.

Parameters:

- tracks - a list of track URIs, URLs or IDs

## current\_user\_saved\_tracks\_delete(self, tracks=None)

Remove one or more tracks from the current user's

"Your Music" library.

Parameters:

- tracks - a list of track URIs, URLs or IDs

## current\_user\_top\_artists(self, limit=20, offset=0, time\_range='medium\_term')

Get the current user's top artists

Parameters:

- limit - the number of entities to return

- offset - the index of the first entity to return

- time\_range - Over what time frame are the affinities computed

Valid-values: short\_term, medium\_term, long\_term

## current\_user\_top\_tracks(self, limit=20, offset=0, time\_range='medium\_term')

Get the current user's top tracks

Parameters:

- limit - the number of entities to return

- offset - the index of the first entity to return

- time\_range - Over what time frame are the affinities computed

Valid-values: short\_term, medium\_term, long\_term

## current\_user\_unfollow\_playlist(self, playlist\_id)

Unfollows (deletes) a playlist for the current authenticated

user

Parameters:

- name - the name of the playlist

## currently\_playing(self, market=None, additional\_types=None)

Get user's currently playing track.

Parameters:

- market - an ISO 3166-1 alpha-2 country code.

- additional\_types - `episode` to get podcast track information

## devices(self)

Get a list of user's available devices.

## episode(self, episode\_id, market=None)

returns a single episode given the episode's ID, URIs or URL

Parameters:

- episode\_id - the episode ID, URI or URL

- market - an ISO 3166-1 alpha-2 country code.

The episode must be available in the given market.

If user-based authorization is in use, the user's country

takes precedence. If neither market nor user country are

provided, the content is considered unavailable for the client.

## episodes(self, episodes, market=None)

returns a list of episodes given the episode IDs, URIs, or URLs

Parameters:

- episodes - a list of episode IDs, URIs or URLs

- market - an ISO 3166-1 alpha-2 country code.

Only episodes available in the given market will be returned.

If user-based authorization is in use, the user's country

takes precedence. If neither market nor user country are

provided, the content is considered unavailable for the client.

## featured\_playlists(self, locale=None, country=None, timestamp=None, limit=20, offset=0)

Get a list of Spotify featured playlists

Parameters:

- locale - The desired language, consisting of a lowercase ISO

639 language code and an uppercase ISO 3166-1 alpha-2 country

code, joined by an underscore.

- country - An ISO 3166-1 alpha-2 country code.

- timestamp - A timestamp in ISO 8601 format:

yyyy-MM-ddTHH:mm:ss. Use this parameter to specify the user's

local time to get results tailored for that specific date and

time in the day

- limit - The maximum number of items to return. Default: 20.

Minimum: 1. Maximum: 50

- offset - The index of the first item to return. Default: 0

(the first object). Use with limit to get the next set of

items.

## me(self)

Get detailed profile information about the current user.

An alias for the 'current\_user' method.

## new\_releases(self, country=None, limit=20, offset=0)

Get a list of new album releases featured in Spotify

Parameters:

- country - An ISO 3166-1 alpha-2 country code.

- limit - The maximum number of items to return. Default: 20.

Minimum: 1. Maximum: 50

- offset - The index of the first item to return. Default: 0

(the first object). Use with limit to get the next set of

items.

## next(self, result)

returns the next result given a paged result

Parameters:

- result - a previously returned paged result

## next\_track(self, device\_id=None)

Skip user's playback to next track.

Parameters:

- device\_id - device target for playback

## pause\_playback(self, device\_id=None)

Pause user's playback.

Parameters:

- device\_id - device target for playback

## playlist(self, playlist\_id, fields=None, market=None, additional\_types=('track',))

Gets playlist by id.

Parameters:

- playlist - the id of the playlist

- fields - which fields to return

- market - An ISO 3166-1 alpha-2 country code or the

string from\_token.

- additional\_types - list of item types to return.

valid types are: track and episode

## playlist\_add\_items(self, playlist\_id, items, position=None)

Adds tracks/episodes to a playlist

Parameters:

- playlist\_id - the id of the playlist

- items - a list of track/episode URIs, URLs or IDs

- position - the position to add the tracks

## playlist\_change\_details(self, playlist\_id, name=None, public=None, collaborative=None, description=None)

Changes a playlist's name and/or public/private state

Parameters:

- playlist\_id - the id of the playlist

- name - optional name of the playlist

- public - optional is the playlist public

- collaborative - optional is the playlist collaborative

- description - optional description of the playlist

## playlist\_cover\_image(self, playlist\_id)

Get cover of a playlist.

Parameters:

- playlist\_id - the id of the playlist

## playlist\_is\_following(self, playlist\_id, user\_ids)

Check to see if the given users are following the given playlist

Parameters:

- playlist\_id - the id of the playlist

- user\_ids - the ids of the users that you want to check to see

if they follow the playlist. Maximum: 5 ids.

## playlist\_items(self, playlist\_id, fields=None, limit=100, offset=0, market=None, additional\_types=('track', 'episode'))

Get full details of the tracks and episodes of a playlist.

Parameters:

- playlist\_id - the id of the playlist

- fields - which fields to return

- limit - the maximum number of tracks to return

- offset - the index of the first track to return

- market - an ISO 3166-1 alpha-2 country code.

- additional\_types - list of item types to return.

valid types are: track and episode

## playlist\_remove\_all\_occurrences\_of\_items(self, playlist\_id, items, snapshot\_id=None)

Removes all occurrences of the given tracks from the given playlist

Parameters:

- playlist\_id - the id of the playlist

- items - list of track/episode ids to remove from the playlist

- snapshot\_id - optional id of the playlist snapshot

## playlist\_remove\_specific\_occurrences\_of\_items(self, playlist\_id, items, snapshot\_id=None)

Removes all occurrences of the given tracks from the given playlist

Parameters:

- playlist\_id - the id of the playlist

- items - an array of objects containing Spotify URIs of the

tracks/episodes to remove with their current positions in

the playlist. For example:

[ { "uri":"4iV5W9uYEdYUVa79Axb7Rh", "positions":[2] },

{ "uri":"1301WleyT98MSxVHPZCA6M", "positions":[7] } ]

- snapshot\_id - optional id of the playlist snapshot

## playlist\_reorder\_items(self, playlist\_id, range\_start, insert\_before, range\_length=1, snapshot\_id=None)

Reorder tracks in a playlist

Parameters:

- playlist\_id - the id of the playlist

- range\_start - the position of the first track to be reordered

- range\_length - optional the number of tracks to be reordered

(default: 1)

- insert\_before - the position where the tracks should be

inserted

- snapshot\_id - optional playlist's snapshot ID

## playlist\_replace\_items(self, playlist\_id, items)

Replace all tracks/episodes in a playlist

Parameters:

- playlist\_id - the id of the playlist

- items - list of track/episode ids to comprise playlist

## playlist\_tracks(self, playlist\_id, fields=None, limit=100, offset=0, market=None, additional\_types=('track',))

Get full details of the tracks of a playlist.

Parameters:

- playlist\_id - the id of the playlist

- fields - which fields to return

- limit - the maximum number of tracks to return

- offset - the index of the first track to return

- market - an ISO 3166-1 alpha-2 country code.

- additional\_types - list of item types to return.

valid types are: track and episode

## playlist\_upload\_cover\_image(self, playlist\_id, image\_b64)

Replace the image used to represent a specific playlist

Parameters:

- playlist\_id - the id of the playlist

- image\_b64 - image data as a Base64 encoded JPEG image string

(maximum payload size is 256 KB)

## previous(self, result)

returns the previous result given a paged result

Parameters:

- result - a previously returned paged result

## previous\_track(self, device\_id=None)

Skip user's playback to previous track.

Parameters:

- device\_id - device target for playback

## recommendation\_genre\_seeds(self)

Get a list of genres available for the recommendations function.

## recommendations(self, seed\_artists=None, seed\_genres=None, seed\_tracks=None, limit=20, country=None, \*\*kwargs)

Get a list of recommended tracks for one to five seeds.

(at least one of `seed\_artists`, `seed\_tracks` and `seed\_genres`

are needed)

Parameters:

- seed\_artists - a list of artist IDs, URIs or URLs

- seed\_tracks - a list of track IDs, URIs or URLs

- seed\_genres - a list of genre names. Available genres for

recommendations can be found by calling

recommendation\_genre\_seeds

- country - An ISO 3166-1 alpha-2 country code. If provided,

all results will be playable in this country.

- limit - The maximum number of items to return. Default: 20.

Minimum: 1. Maximum: 100

- min/max/target\_<attribute> - For the tuneable track

attributes listed in the documentation, these values

provide filters and targeting on results.

## repeat(self, state, device\_id=None)

Set repeat mode for playback.

Parameters:

- state - `track`, `context`, or `off`

- device\_id - device target for playback

## search(self, q, limit=10, offset=0, type='track', market=None)

searches for an item

Parameters:

- q - the search query (see how to write a query in the

official documentation <https://developer.spotify.com/documentation/web-api/reference/search/)> # noqa

- limit - the number of items to return (min = 1, default = 10, max = 50). The limit is applied

within each type, not on the total response.

- offset - the index of the first item to return

- type - the types of items to return. One or more of 'artist', 'album',

'track', 'playlist', 'show', and 'episode'. If multiple types are desired,

pass in a comma separated string; e.g., 'track,album,episode'.

- market - An ISO 3166-1 alpha-2 country code or the string

from\_token.

## search\_markets(self, q, limit=10, offset=0, type='track', markets=None, total=None)

(experimental) Searches multiple markets for an item

Parameters:

- q - the search query (see how to write a query in the

official documentation <https://developer.spotify.com/documentation/web-api/reference/search/)> # noqa

- limit - the number of items to return (min = 1, default = 10, max = 50). If a search is to be done on multiple

markets, then this limit is applied to each market. (e.g. search US, CA, MX each with a limit of 10).

- offset - the index of the first item to return

- type - the types of items to return. One or more of 'artist', 'album',

'track', 'playlist', 'show', or 'episode'. If multiple types are desired, pass in a comma separated string.

- markets - A list of ISO 3166-1 alpha-2 country codes. Search all country markets by default.

- total - the total number of results to return if multiple markets are supplied in the search.

If multiple types are specified, this only applies to the first type.

## seek\_track(self, position\_ms, device\_id=None)

Seek to position in current track.

Parameters:

- position\_ms - position in milliseconds to seek to

- device\_id - device target for playback

## set\_auth(self, auth)

## show(self, show\_id, market=None)

returns a single show given the show's ID, URIs or URL

Parameters:

- show\_id - the show ID, URI or URL

- market - an ISO 3166-1 alpha-2 country code.

The show must be available in the given market.

If user-based authorization is in use, the user's country

takes precedence. If neither market nor user country are

provided, the content is considered unavailable for the client.

## show\_episodes(self, show\_id, limit=50, offset=0, market=None)

Get Spotify catalog information about a show's episodes

Parameters:

- show\_id - the show ID, URI or URL

- limit - the number of items to return

- offset - the index of the first item to return

- market - an ISO 3166-1 alpha-2 country code.

Only episodes available in the given market will be returned.

If user-based authorization is in use, the user's country

takes precedence. If neither market nor user country are

provided, the content is considered unavailable for the client.

## shows(self, shows, market=None)

returns a list of shows given the show IDs, URIs, or URLs

Parameters:

- shows - a list of show IDs, URIs or URLs

- market - an ISO 3166-1 alpha-2 country code.

Only shows available in the given market will be returned.

If user-based authorization is in use, the user's country

takes precedence. If neither market nor user country are

provided, the content is considered unavailable for the client.

## shuffle(self, state, device\_id=None)

Toggle playback shuffling.

Parameters:

- state - true or false

- device\_id - device target for playback

## start\_playback(self, device\_id=None, context\_uri=None, uris=None, offset=None, position\_ms=None)

Start or resume user's playback.

Provide a `context\_uri` to start playback or a album,

artist, or playlist.

Provide a `uris` list to start playback of one or more

tracks.

Provide `offset` as {"position": <int>} or {"uri": "<track uri>"}

to start playback at a particular offset.

Parameters:

- device\_id - device target for playback

- context\_uri - spotify context uri to play

- uris - spotify track uris

- offset - offset into context by index or track

- position\_ms - (optional) indicates from what position to start playback.

Must be a positive number. Passing in a position that is

greater than the length of the track will cause the player to

start playing the next song.

## track(self, track\_id, market=None)

returns a single track given the track's ID, URI or URL

Parameters:

- track\_id - a spotify URI, URL or ID

- market - an ISO 3166-1 alpha-2 country code.

## tracks(self, tracks, market=None)

returns a list of tracks given a list of track IDs, URIs, or URLs

Parameters:

- tracks - a list of spotify URIs, URLs or IDs. Maximum: 50 IDs.

- market - an ISO 3166-1 alpha-2 country code.

## transfer\_playback(self, device\_id, force\_play=True)

Transfer playback to another device.

Note that the API accepts a list of device ids, but only

actually supports one.

Parameters:

- device\_id - transfer playback to this device

- force\_play - true: after transfer, play. false:

keep current state.

## user(self, user)

Gets basic profile information about a Spotify User

Parameters:

- user - the id of the usr

## user\_follow\_artists(self, ids=[])

Follow one or more artists

Parameters:

- ids - a list of artist IDs

## user\_follow\_users(self, ids=[])

Follow one or more users

Parameters:

- ids - a list of user IDs

## user\_playlist(self, user, playlist\_id=None, fields=None, market=None)

## user\_playlist\_add\_tracks(self, user, playlist\_id, tracks, position=None)

## user\_playlist\_change\_details(self, user, playlist\_id, name=None, public=None, collaborative=None, description=None)

## user\_playlist\_create(self, user, name, public=True, collaborative=False, description='')

Creates a playlist for a user

Parameters:

- user - the id of the user

- name - the name of the playlist

- public - is the created playlist public

- collaborative - is the created playlist collaborative

- description - the description of the playlist

## user\_playlist\_follow\_playlist(self, playlist\_owner\_id, playlist\_id)

Add the current authenticated user as a follower of a playlist.

Parameters:

- playlist\_owner\_id - the user id of the playlist owner

- playlist\_id - the id of the playlist

## user\_playlist\_is\_following(self, playlist\_owner\_id, playlist\_id, user\_ids)

Check to see if the given users are following the given playlist

Parameters:

- playlist\_owner\_id - the user id of the playlist owner

- playlist\_id - the id of the playlist

- user\_ids - the ids of the users that you want to check to see

if they follow the playlist. Maximum: 5 ids.

## user\_playlist\_remove\_all\_occurrences\_of\_tracks(self, user, playlist\_id, tracks, snapshot\_id=None)

Removes all occurrences of the given tracks from the given playlist

Parameters:

- user - the id of the user

- playlist\_id - the id of the playlist

- tracks - the list of track ids to remove from the playlist

- snapshot\_id - optional id of the playlist snapshot

## user\_playlist\_remove\_specific\_occurrences\_of\_tracks(self, user, playlist\_id, tracks, snapshot\_id=None)

Removes all occurrences of the given tracks from the given playlist

Parameters:

- user - the id of the user

- playlist\_id - the id of the playlist

- tracks - an array of objects containing Spotify URIs of the

tracks to remove with their current positions in the

playlist. For example:

[ { "uri":"4iV5W9uYEdYUVa79Axb7Rh", "positions":[2] },

{ "uri":"1301WleyT98MSxVHPZCA6M", "positions":[7] } ]

- snapshot\_id - optional id of the playlist snapshot

## user\_playlist\_reorder\_tracks(self, user, playlist\_id, range\_start, insert\_before, range\_length=1, snapshot\_id=None)

Reorder tracks in a playlist

Parameters:

- user - the id of the user

- playlist\_id - the id of the playlist

- range\_start - the position of the first track to be reordered

- range\_length - optional the number of tracks to be reordered (default: 1)

- insert\_before - the position where the tracks should be inserted

- snapshot\_id - optional playlist's snapshot ID

## user\_playlist\_replace\_tracks(self, user, playlist\_id, tracks)

Replace all tracks in a playlist

Parameters:

- user - the id of the user

- playlist\_id - the id of the playlist

- tracks - the list of track ids to add to the playlist

## user\_playlist\_tracks(self, user=None, playlist\_id=None, fields=None, limit=100, offset=0, market=None)

## user\_playlist\_unfollow(self, user, playlist\_id)

Unfollows (deletes) a playlist for a user

Parameters:

- user - the id of the user

- name - the name of the playlist

## user\_playlists(self, user, limit=50, offset=0)

Gets playlists of a user

Parameters:

- user - the id of the usr

- limit - the number of items to return

- offset - the index of the first item to return

## user\_unfollow\_artists(self, ids=[])

Unfollow one or more artists

Parameters:

- ids - a list of artist IDs

## user\_unfollow\_users(self, ids=[])

Unfollow one or more users

Parameters:

- ids - a list of user IDs

## volume(self, volume\_percent, device\_id=None)

Set playback volume.

Parameters:

- volume\_percent - volume between 0 and 100

- device\_id - device target for playback

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

Data descriptors defined here:

\_\_dict\_\_

dictionary for instance variables (if defined)

\_\_weakref\_\_

list of weak references to the object (if defined)

auth\_manager

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

Data and other attributes defined here:

country\_codes = ['AD', 'AR', 'AU', 'AT', 'BE', 'BO', 'BR', 'BG', 'CA',...

default\_retry\_codes = (429, 500, 502, 503, 504)

max\_retries = 3