lyricsfandom

Release 0.1

May 25, 2020

CHAPTER 1

Getting Started

1.1 Installation

Install *lyricsfandom* package from *PyPi*:

```
pip install lyricsfandom
```

Or from *GitHub*:

```
git clone https://github.com/arthurdjn/scrape-lyricwiki
```

1.2 Usage

You can use the simplified API to look for lyrics and music.

Example:

```
from lyricsfandom import LyricWiki

# Connect to the API
wiki = LyricWiki()
# Search for an artist. `LyricsFandom` is not case sensitive.
artist = wiki.search_artist('london grammar')
artist
```

Output:

```
Artist: London Grammar
```

Then, you can search for albums too.

Example:

```
# Search for an album
album = wiki.search_album('london grammar', 'if you wait')
album
```

Output:

```
London Grammar: Album "If You Wait" (2013), Songs: 17
```

Finally, you can scrape for songs.

Example:

```
# Search for an album
song = wiki.search_song('london grammar', 'strong')
song
```

Output:

```
London Grammar: "Strong" from Album "If You Wait" (2013)
```

. . . and scrape lyrics.

Example:

```
# Search for an album
lyrics = song.get_lyrics()
print(lyrics)
```

Output:

```
Excuse me for a while
While I'm wide eyed
And I'm so damn caught in the middle
I've excused you for a while
While I'm wide eyed
And I'm so down caught in the middle

And a lion, a lion roars
Would you not listen?
If a child, a child cries
Would you not forgive them?

[...]
```

CHAPTER 2

Examples

The requests should be made from the API. Then, you can have access to albums, songs, lyrics.

2.1 LyricsFandom API

```python
from lyricsfandom import LyricWiki

# Connect to the API
wiki = LyricWiki()
# Search for an artist. `LyricsFandom` is not case sensitive.
artist = wiki.search_artist('london grammar')
album = wiki.search_album('london grammar', 'if you wait')
song = wiki.search_song('london grammar', 'strong')
lyrics = song.get_lyrics()
```

You can have access to their attributes with:

```python
# From an Artist
artist_name = artist.artist_name

# From an Album
artist_name = album.artist_name
album_name = album.album_name
album_typz = album.album_typz
album_year = album.album_year

# From a Song
artist_name = song.artist_name
artist_name = song.artist_name
album_name = song.album_name
album_type = song.album_type
album_year = song.album_year
song_name = song.song_name
```

2.2 Access data

Once you have an object instance, you can retrieve data:

```
# From an Artist
artist = wiki.search_artist('london grammar')
albums = artist.get_albums()
songs = artist.get_songs()

# From an Album
album = wiki.search_album('london grammar', 'if you wait')
songs = album.get_songs()
```

Note:

If you want to navigate through albums, songs, you may prefer using `.songs()` or `.albums()` methods, which yields items successively and thus are more optimized as all items are not loaded at directly.

```
# From and Artist
artist = wiki.search_artist('london grammar')
for song in artist.songs():
    lyrics = song.get_lyrics()
    print(lyrics)
    print('\n-----\n')
```

From children classes (Artist –> Album –> Song), you can retrieve data too:

```
# From a Song
song = wiki.search_song('london grammar', 'strong')
album = song.get_album()
artist = song.get_artist()

# From an Album
album = wiki.search_album('london grammar', 'if you wait')
artist = album.get_artist()
```

2.3 Save and export

You can save all classes with the `.to_json()` method. The `'ascii'` argument will transforms all string to ASCII format. If you don't want it, just remove it.

```
# From an Artist
artist = wiki.search_artist('london grammar')
artist_data = artist.to_json(encode='ascii')

# From an Album
album = wiki.search_album('london grammar', 'if you wait')
album_data = album.to_json(encode='ascii')

# From a Song (contains lyrics)
song = wiki.search_song('london grammar', 'strong')
song_data = song.to_json(encode='ascii')
```

CHAPTER 3

lyricsfandom

3.1 lyricsfandom.api

API and other classes to connect on Lyrics Wiki.

class `lyricsfandom.api.`**`LyricWiki`**(*verbose=False*, *sleep=0*, *user=None*)

Main API for *Lyric Wiki* scrapping.

It basically wraps `Artist`, `Album` and `Song` classes.

`get_albums`(*artist_name*, *cover=True*, *other=True*)

Get all albums from an artist.

Parameters

- **`artist_name`** (*`string`*) – name of the artist to get.
- **`cover`** (*`bool`*) – if `True` scrape featuring or covers songs.
- **`other`** (*`bool`*) – if `True` scrape remixes or compilation albums.

Returns list(Album)

`get_discography`(*artist_name*, *cover=True*, *other=True*, *encode=None*)

Get the discography of an artist, in a JSON format.

Note: The returned dictionary is in a nested format.

Parameters

- **`artist_name`** (*`string`*) – name of the artist to get.
- **`cover`** (*`bool`*) – if `True` scrape featuring or covers songs.
- **`other`** (*`bool`*) – if `True` scrape remixes or compilation albums.
- **`encode`** (*`string`*) – encode the string text (ex: `encode='ascii`). Default is None.

Returns dict

get_lyrics(*artist_name*, *cover=False*, *other=False*)
Get all lyrics from an artist.

Parameters

- **artist_name** (*string*) – name of the artist to get.
- **cover** (*bool*) – if `True` scrape featuring or covers songs.
- **other** (*bool*) – if `True` scrape remixes or compilation albums.

Returns lyrics in a JSON format.

Return type dict

search_album(*artist_name*, *album_name*)
Search an album from *Lyric Wiki* server.

Parameters

- **artist_name** (*string*) – name of the artist who made the album.
- **album_name** (*string*) – name of the album.

Returns Album

search_artist(*artist_name*, *cover=False*, *other=False*)
Search an artist from *Lyric Wiki* server.

Parameters

- **artist_name** (*string*) – name of the artist to get.
- **cover** (*bool*) – if `True` scrape featuring or covers songs.
- **other** (*bool*) – if `True` scrape remixes or compilation albums.

Returns Artist

search_song(*artist_name*, *song_name*)
Search a song from *Lyric Wiki* server.

Parameters

- **artist_name** (*string*) – name of the artist who made the song.
- **song_name** (*string*) – name of the song.

Returns Song

set_sleep(*sleep*)
Time before connecting again to a new page.

Parameters **sleep** (*float*) – seconds to wait.

set_user(*user*)
Change the user agent used to connect on internet.

Parameters **user** (*string*) – user agent to use with urllib.request.

set_verbose(*verbose*)
Change the log / display while surfing on internet.

Parameters **verbose** (*bool*) – if `True` will display a log message each time it is connected to a page.

3.2 lyricsfandom.meta

Base classes. They are optional and can be removed for simplicity. However, they provides a better API and sperates Artist / Album / Song in a better way.

class `lyricsfandom.meta.`**`AlbumMeta`**(*artist_name*, *album_name*, *album_year=None*, *album_type=None*)

Defines an Abstract Album from `https://lyrics.fandom.com/wiki/`.

- `album_name`: album of the artist.
- `album_type`: type of album.
- `album_year`: released of the album.

classmethod from_artist(*artist*, *album_name*)

Construct an Artist from an url.

Parameters

- **artist** (`Artist`) – artist to extract the album from.
- **album_name** (*string*) – album name.

classmethod from_url(*url*)

Construct an Album from an url.

Parameters **url** (*string*) – url of the album page.

get_artist()

Retrieve the artist class linked to the album (if it exists). It is usually called when an album has been searched from an `Artist` class. Then, using this function will point to the same `Artist` object.

Returns Artist

register_artist(*artist*)

Manually set the pointer to an `Artist`.

Parameters **artist** (`Artist`) – artist related to the album.

to_json(*encode=None*)

Retrieve the full playlist from an album; in a JSON format.

Returns dict

unregister()

Unlink the album to its artist.

class `lyricsfandom.meta.`**`ArtistMeta`**(*artist_name*)

Defines an Abstract Artist / Band from `https://lyrics.fandom.com/wiki/`.

- `artist_name`: name of the artist.
- `artist_id`: id of the artist.
- `base`: base page of the artist.
- `href`: href page of the artist.
- `url`: url page of the artist.

classmethod from_url(*url*)

Construct an Artist from an url.

Parameters **url** (*string*) – url of the artist page.

get_links()

Retrieve merchandise links from a *Lyric Wiki* page. If the page (and links) exists, it will save it in a private attribute, to avoid loading again and again the same links if the method is called multiple times.

Returns dict

items()

Basic Set-up to iterate through items (albums, songs. . .).

Returns Album or Song

set_links(*value*)

Set manually the `links` attribute.

Parameters **value** (*dict*) – links to change.

to_json(*encode=None*)

Retrieve the full discography from an artist; in a JSON format.

Returns dict

class lyricsfandom.meta.**LyricWikiMeta**

The `LyricWikiMeta` is an abstract class that all object pointing to *Lyric Wiki* web site should inherits. It provide basic set-up to connect and access to *Lyric Wiki* website.

class lyricsfandom.meta.**SongMeta**(*artist_name*, *song_name*, *album_name=None*, *album_year=None*, *album_type=None*)

Defines an Abstract Song from `https://lyrics.fandom.com/`.

- `song_name`: name of the song.
- `song_id`: id of the song.
- `lyrics`: lyrics of the song.

classmethod from_album(*album*, *song_name*)

Construct a Song from an url.

Parameters

- **album** (`Album`) – album to extract the song from.
- **song_name** (*string*) – song name.

classmethod from_artist(*artist*, *song_name*)

Construct an Artist from an url.

Parameters

- **artist** (`Artist`) – artist to extract the album from.
- **song_name** (*string*) – song name.

classmethod from_url(*url*)

Construct a Song from an url.

Parameters **url** (*string*) – url of the lyrics song page.

get_album()

Get the parent album pointing to the song, if it exists.

Returns Album

register_album(*album*)

Link the song to a parent album.

Parameters **album** (`Album`) – album to link to the song.

set_lyrics(*value*)
Manually set the lyrics of the current song.

Parameters **value** (*string*) – new lyrics.

to_json(*encode=None*)
Retrieve the full information / lyrics from a song; in a JSON format.

Returns dict

unregister()
Unlink the song to both artist and album.

3.3 lyricsfandom.connect

A scrapper is used to connect to a website and extract data.

lyricsfandom.connect.**connect**(*url*)
Connect to an URL.

Parameters

- **url** (*string*) – url path
- **sleep** (*float*) – number of seconds to sleep before connection.
- **verbose** (*bool*) – print the url if `True`.

Returns soup

3.4 lyricsfandom.scrape

Functions used to connect, extract, and display data from lyrics fandom website.

These functions are used to scrape data from `HTML` page connection. They are used inside `Artist, Album, Song` classes.

The major part of this functions used a soup parameter, i.e. a `Beautiful Soup Tag` element on a wab page (usually the whole page, not just a `<div>` or other `HTML` elements.

lyricsfandom.scrape.**generate_album_url**(*artist_name*, *album_name*, *album_year*)
Generate a *Lyric Wiki* url from of an album page from its artist and name / year.

Parameters

- **artist_name** (*string*) – name of the Artist.
- **album_name** (*string*) – name of the Album.
- **album_year** (*string*) – year of an Album.

Returns string

Examples::

```
>>> artist_name = 'london grammar'
>>> album_name = 'if you wait'
>>> album_year = 2013
>>> generate_album_url(artist_name, album_name, album_year)
    https://lyrics.fandom.com/wiki/London_Grammar:If_You_Wait_(2013)
```

`lyricsfandom.scrape.`**`generate_artist_url`**`(`*artist_name*`)`

Generate a *Lyric Wiki* url of an artist page from its name.

Parameters **`artist_name`** (*`string`*) – name of the Artist.

Returns string

Examples::

```
>>> artist_name = 'london grammar'
>>> generate_artist_url(artist_name)
    https://lyrics.fandom.com/wiki/London_Grammar
```

`lyricsfandom.scrape.`**`get_artist_info`**`(`*soup*`)`

Get additional information about the artist / band.

Parameters **`soup`** (*`bs4.element.Tag`*) – connection to a wiki artist page.

Returns dict

`lyricsfandom.scrape.`**`get_external_links`**`(`*soup*`)`

Retrieve the different links from a *Lyric Wiki* page. The links returned can be found in the *External Links* page section, and usually references to other platforms (like Last.fm, Amazon, iTunes etc.).

Parameters **`soup`** (*`bs4.element.Tag`*) – connection to the *Lyric Wiki* page.

Returns dict

Examples::

```
>>> # Import packages
>>> import bs4   # for web scrapping
>>> import urllib.request   # to connect
```

```
>>> # Set Up: connect to a lyric wiki page
>>> USER = 'Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.9.0.7) Gecko/
↪2009021910 Firefox/3.0.7'
>>> HEADERS = {'User-Agent': USER}
>>> URL = 'https://lyrics.fandom.com/wiki/London_Grammar:Who_Am_I'
>>> req = urllib.request.Request(URL, headers=HEADERS)
>>> page = urllib.request.urlopen(req)
>>> soup = bs4.BeautifulSoup(page, 'lxml')
```

```
>>> # Retrieve links from the page
>>> get_external_links(soup)
    {'Amazon': ['https://www.amazon.com/exec/obidos/redirect?link_code=ur2&
↪tag=wikia-20&camp=1789&creative=9325&path=https%3A%2F%2Fwww.amazon.com%2Fdp
↪%2FB00J0QJ84E'],
     'Last.fm': ['https://www.last.fm/music/London+Grammar',
      'https://www.last.fm/music/London+Grammar/If+You+Wait'],
     'iTunes': ['https://itunes.apple.com/us/album/695805771'],
     'AllMusic': ['https://www.allmusic.com/album/mw0002559862'],
     'Discogs': ['http://www.discogs.com/master/595953'],
     'MusicBrainz': ['https://musicbrainz.org/release-group/dbf36a9a-df02-
↪41c4-8fa9-5afe599960b0'],
     'Spotify': ['https://open.spotify.com/album/0YTj3vyjZmlfp16S2XGo50']}
```

`lyricsfandom.scrape.`**`get_lyrics`**`(`*soup*`)`

Get lyrics from a *Lyric Wiki* song page.

Returns string

Examples::

```
>>> # Import packages
>>> import bs4  # for web scrapping
>>> import urllib.request   # to connect
```

```
>>> # Set Up: connect to a lyric wiki page
>>> USER = 'Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.9.0.7) Gecko/
↪2009021910 Firefox/3.0.7'
>>> HEADERS = {'User-Agent': USER}
>>> URL = 'https://lyrics.fandom.com/wiki/London_Grammar:Shyer'
>>> req = urllib.request.Request(URL, headers=HEADERS)
>>> page = urllib.request.urlopen(req)
>>> soup = bs4.BeautifulSoup(page, 'lxml')
```

```
>>> # Scrape the lyrics
>>> lyrics = get_lyrics(soup)
>>> print(lyrics)
    I'm feeling shyer and the world gets darker
    Hold yourself a little higher
    Bridge that gap just further
    And all your being
    I'd ask you to give it up
    An ancient feeling love
    So beautifully dressed up
```

Feeling shyer, I'm feeling shyer I'm feeling shyer

Maybe you should call her Deep in the night for her And all your being I'd ask you to give it up I'd ask you to give it up

`lyricsfandom.scrape.`**`scrape_albums`**(*soup*)

Scrape albums tags, usually from the main artist wiki page. This function will successively yield albums.

Note: The function yield `<h2>` tags.

Parameters **`soup`** (*`bs4.element.Tag`*) – artist page connection.

Returns albums tags of an artist page.

Return type yield bs4.element.Tag

Examples::

```
>>> # Import packages
>>> import bs4  # for web scrapping
>>> import urllib.request   # to connect
```

```
>>> # Set Up: connect to a lyric wiki page
>>> USER = 'Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.9.0.7) Gecko/
↪2009021910 Firefox/3.0.7'
>>> HEADERS = {'User-Agent': USER}
```

(continues on next page)

(continued from previous page)

```
>>> URL = 'https://lyrics.fandom.com/wiki/London_Grammar'
>>> req = urllib.request.Request(URL, headers=HEADERS)
>>> page = urllib.request.urlopen(req)
>>> soup = bs4.BeautifulSoup(page, 'lxml')
```

```
>>> # Scrape albums
>>> for album_tag in scrape_albums(soup):
...     print(album_tag.text)
    Strong (2013)
    If You Wait (2013)
    Truth Is a Beautiful Thing (2017)
    Songs on Compilations and Soundtracks
    Additional information
    External links
```

`lyricsfandom.scrape.`**`scrape_songs`**(*album_h2_tag*, *li_tag='ol'*)

Scrape songs from an album. This function should be used to scrape on artist's page. The optional parameter `li_tag` is used to specify whether or not to scrape for released albums (`'ol'` tags) or covers, singles, live etc. (`'ul'` tags). They can be combined using `li_tag=['ol', 'ul']` to scrape among all songs.

Parameters

- **album_h2_tag** (*bs4.element.Tag*) – album tag. Only songs under this tag will be yielded.
- **li_tag** (*string or iterable*) – tags names to scrape songs from.

Returns yield song tags corresponding to the album tag.

Return type yield bs4.element.Tag

Examples::

```
>>> # Import packages
>>> import bs4  # for web scrapping
>>> import urllib.request  # to connect
```

```
>>> # Set Up: connect to a lyric wiki page
>>> USER = 'Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.9.0.7) Gecko/
↪2009021910 Firefox/3.0.7'
>>> HEADERS = {'User-Agent': USER}
>>> URL = 'https://lyrics.fandom.com/wiki/London_Grammar'
>>> req = urllib.request.Request(URL, headers=HEADERS)
>>> page = urllib.request.urlopen(req)
>>> soup = bs4.BeautifulSoup(page, 'lxml')
```

```
>>> # Scrape songs from the first album,  'Strong (2013)' EP.
>>> album_h2_tag = soup.select('h2 .mw-headline')[0].parent
>>> for song_tag in scrape_albums(album_h2_tag):
...     print(song_tag.text)
    Strong
    Feelings
```

```
>>> # Scrape all songs from the artist page
>>> for album_tag in scrape_albums(soup):
>>>     album_h2_tag = album_tag.parent
```

(continues on next page)

(continued from previous page)

```
>>>     for song_tag in scrape_songs(album_h2_tag):
>>>         print(album_h2_tag.text)
>>>         print(song_tag.text)
>>>         print('------------')
    Strong (2013)
    Strong
    Feelings
    ------------
    If You Wait (2013)
    Hey Now
    Stay Awake
    Shyer
    Wasting My Young Years
    Sights
    Strong
    etc. ...
```

3.5 lyricsfandom.utils

Utilities functions.

`lyricsfandom.utils.`**`capitalize`**`(`*string_raw*`)`
Capitalize a string, even if it is between quotes like “, ‘.

Parameters **`string_raw`** (*`string`*) – text to capitalize.

Returns string

`lyricsfandom.utils.`**`name_to_wiki`**`(`*name*`)`
Process artist, album and song’s name.

Parameters **`name`** –

Returns:

`lyricsfandom.utils.`**`name_to_wiki_id`**`(`*name*`)`
Generate a *Lyric Wiki* ID from a name.

Parameters **`name`** (*`string`*) – name of an artist / song.

Returns string

`lyricsfandom.utils.`**`parse_album_header`**`(`*album_header*`)`
Split the album title in half, to retrieve its name an year.

Examples:

```
>>> album_title = 'His Young Heart (2011)'
>>> split_album_title(album_title)
    (His Young Heart, 2011)
```

Parameters **`album_header`** (*`string`*) – album header / title to split

Returns album name and year.

Return type tuple

`lyricsfandom.utils.`**`parse_song_title`**(*song_title*, *artist_name=None*)

Split a song title to retrieve the artist name and song name. Additional argument can be added to better retrieve these names.

Parameters

- **`song_title`** (*`string`*) – song header (or title for the `<a>` element)
- **`artist_name`** (*`string, optional`*) – name of the artist.

Returns tuple

`lyricsfandom.utils.`**`process_lyrics`**(*lyrics*)

Process lyrics.

Parameters **`lyrics`** (*`string`*) – lyrics to tokenize / modify.

Returns string

`lyricsfandom.utils.`**`serialize_dict`**(*dict_raw*)

Serialize a dictionary in ASCII format so it can be saved as a JSON.

Parameters **`dict_raw`** (*`dict`*) –

Returns dict

`lyricsfandom.utils.`**`serialize_list`**(*list_raw*)

Serialize a list in ASCII format, so it can be saved as a JSON.

Parameters **`list_raw`** (*`list`*) –

Returns list

`lyricsfandom.utils.`**`split_header`**(*header*)

Split the header to get the artist name, album, and year.

Examples:

```
>>> album_title = 'Daughter:His Young Heart (2011)'
>>> split_album_title(album_title)
    (Daughter, His Young Heart, 2011)
```

Parameters **`header`** (*`string`*) – album header / title to split

Returns album name and year.

Return type tuple

`lyricsfandom.utils.`**`split_song_header`**(*song_header*)

Split the song title in half, to retrieve its artist an name.

Examples:

```
>>> song_header = 'Daughter:Run Lyrics'
>>> split_song_header(song_header)
    (Daughter, Run)
```

Parameters **`song_header`** (*`string`*) – song header / title to split

Returns artist name and song name.

Return type tuple

CHAPTER 4

lyricsfandom.music

4.1 lyricsfandom.music.artist

Defines an artist from `LyricWiki` server. Extract albums and songs from `https://lyrics.fandom.com/Artist_Name` page.

Examples::

```
>>> # Note that names are not case sensitive
>>> artist = Artist('daughter')
>>> artist
    Artist: Daughter
```

```
>>> # Get all albums (compilation, covers etc. included)
>>> artist.get_albums()
    [Daughter: EP "His Young Heart" (2011), Songs: 4,
     Daughter: EP "The Wild Youth" (2011), Songs: 4,
     Daughter: Album "If You Leave" (2013), Songs: 12,
     Daughter: Album "Not To Disappear" (2016), Songs: 11,
     Daughter: Album "Music From Before The Storm" (2017), Songs: 13,
     Daughter: "Songs On Compilations", Songs: 2,
     Daughter: Single "Other Songs", Songs: 1]
```

```
>>> # Only look for albums / singles released by the artist
>>> artist.get_albums(cover=False, other=False)
    [Daughter: EP "His Young Heart" (2011), Songs: 4,
     Daughter: EP "The Wild Youth" (2011), Songs: 4,
     Daughter: Album "If You Leave" (2013), Songs: 12,
     Daughter: Album "Not To Disappear" (2016), Songs: 11,
     Daughter: Album "Music From Before The Storm" (2017), Songs: 13,
     Daughter: Single "Other Songs", Songs: 1]
```

```
>>> # Idem for get_songs()
>>> # Look for an album / song from the artist
>>> song = artist.search_song('candles')
>>> lyrics = song.get_lyrics()
>>> print(lyrics)
    That boy, take me away, into the night
   Out of the hum of the street lights and into a forest
   I'll do whatever you say to me in the dark
   Scared I'll be torn apart by a wolf in mask of a familiar name on a␣
↪birthday card
```

Blow out all the candles, blow out all the candles “You’re too old to be so shy,” he says to me so I stay the night Just a young heart confusing my mind, but we’re both in silence Wide-eyed, both in silence Wide-eyed, like we’re in a crime scene etc. . . .

```
>>> # Retrieve the artist from a song / album object
>>> song.get_artist()
    Artist: Daughter
```

```
>>> # Get additional information from the artist
>>> artist.get_info()
    {'Years Active': '2010 - present',
     'Band Members': ['Elena Tonra', 'Igor Haefeli', 'Remi Aguilella'],
     'Genres': ['Indie Folk', 'Folk Rock'],
     'Record Labels': ['4AD']}
```

```
>>> # Get merchandise links
>>> artist.get_links()
   {'Amazon': ['https://www.amazon.com/exec/obidos/redirect?link_code=ur2&
↪tag=wikia-20&camp=1789&creative=9325&path=https%3A%2F%2Fwww.amazon.com%2F-%2Fe
↪%2FB001LHN42M'],
    'iTunes': ['https://itunes.apple.com/us/artist/469701923'],
    'AllMusic': ['https://www.allmusic.com/artist/mn0003013627'],
    'Discogs': ['http://www.discogs.com/artist/2218596'],
    'MusicBrainz': ['https://musicbrainz.org/artist/a1ced3e5-476c-4046-bd74-
↪d428f419989b'],
    'Spotify': ['https://open.spotify.com/artist/46CitWgnWrvF9t70C2p1Me'],
    'Bandcamp': ['https://ohdaughter.bandcamp.com/']}
```

```
>>> # Convert the data to JSON
>>> data = artist.to_json(encode='ascii', nested=False)
```

These are the most common functions, but others can be used to modify the data.

class `lyricsfandom.music.artist.`**`Artist`**(*artist_name*)

Defines an Artist / Band from `https://lyrics.fandom.com/wiki/`.

- `artist_name`: name of the artist.
- `base`: base page of Lyric Wiki.
- `href`: href link of the artist.
- `url`: url page of the artist.

`add_album`(*album*, *force=None*)

Add an album to the artist. When adding a new argument, the album artist’s name can be changed to match the parent artist, using `force=True`. If the provided album is the name of an album, it will automatically create an (empty) album and add it to the artist.

Parameters

- **album** (Album or string) – album (or album name) to add to the current artist.
- **force** (bool) – if True, change the album's artist_name attribute to match the artist's name.

Examples::

```
>>> artist = Artist('daughter')
>>> album = Album('daugghter', 'the wild youth')
>>> artist.add_album(album)
>>> artist.get_albums()
    [Daugghter: "The Wild Youth", Songs: 0]
```

```
>>> artist = Artist('daughter')
>>> album = Album('daugghter', 'the wild youth')
>>> artist.add_album(album, force=True)
>>> artist.get_albums()
    [Daughter: "The Wild Youth", Songs: 0]
```

albums(***kwargs*)

Iterate through all Albums made by the artist.

Returns yield ALbum

classmethod from_url(*url*)

Construct an Artist from an url.

Parameters **url** (string) – url.

Returns Artist

Examples::

```
>>> artist = Artist.from_url('https://lyrics.fandom.com/wiki/Daughter')
>>> artist
    Artist: Daughter
```

get_albums(*cover=False*, *other=False*)

Get a list of all albums made by the artist. Keywords arguments can be provided to scrape only from released albums, and reject covers, remix, compilation etc.

Parameters

- **cover** (bool) – if True scrape featuring or covers songs.
- **other** (bool) – if True scrape remixes or compilation albums.

Returns list

get_info()

Retrieve additional information of an Artist (like band members, labels, genres etc.).

Returns dict

Examples::

```
>>> artist = Artist('Daughter')
>>> artist.get_info()
    {'Years Active': '2010 - present',
     'Band Members': ['Elena Tonra', 'Igor Haefeli', 'Remi Aguilella'],
     'Genres': ['Indie Folk', 'Folk Rock'],
     'Record Labels': ['4AD']}
```

get_songs(*cover=False*, *other=False*)
Get a list of all songs made by the artist. Keywords arguments can be provided to scrape only from songs made by the artist, and reject covers etc.

Parameters

- **cover** (*bool*) – if `True` scrape featuring or covers songs.
- **other** (*bool*) – if `True` scrape remixes or compilation albums.

Returns list

items(*cover=True*, *other=True*)
Connect to `LyricWiki` server and scrape albums / songs. Keywords arguments can be provided to scrape only from released albums, and reject covers, remix, compilation etc.

Parameters

- **cover** (*bool*) – if `True` scrape featuring or covers songs.
- **other** (*bool*) – if `True` scrape remixes or compilation albums.

Returns yield Album

search_album(*album_name*)
Search an album from an artist's discography.

Parameters **album_name** (*string*) – name of the album to look for.

Returns Album

search_song(*song_name*)
Search a song from an artist's playlist.

Parameters **song_name** (*string*) – name of the song to look for

Returns Song

songs(***kwargs*)
Iterate through all songs made by the artist.

Returns yield Song

to_json(*encode=None*)
Get the discography of an artist.

Returns list

4.2 lyricsfandom.music.album

Extract lyrics and songs from `https://lyrics.fandom.com/` website.

Examples

```
# 1. Generate an album from scratch
album = Album('Bon Iver', 'For Emma, Forever Ago')
# Scrape songs.
songs = album.get_songs()
# Be careful as this album was created from scratch it is not linked to any␣
↪``Artist`` instance.
# However, there is still the artist's name saved.
album.get_artist()   # None
album.artist_name   # 'Bon Iver'

# 2. Use an album from an artist
artist = Artist('Bon Iver')
album = Album.from_artist(artist, 'For Emmma, Forever Ago')
album.get_artist()   # Artist: 'Bon Iver'
# Or search it from the artist class.
album = artist.search_album('For Emma, Forever Ago')
```

class `lyricsfandom.music.album.`**`Album`**(*artist_name*, *album_name*, *album_type=None*, *album_year=None*)

Defines an Album from `https://lyrics.fandom.com/wiki/`.

- `album_name`: album of the artist.
- `album_type`: type of album.
- `album_year`: released of the album.
- *`songs`*: songs of the album.

`add_song`(*song*, *force=None*)

Add a song to the album. When adding, the song artist's name / album names can be changed to match the parent album, using `force=True`. If the provided song is the name of a song (a string), it will automatically create an (empty) song and add it to the album.

Parameters

- **song** (`Song or string`) – song (or song name) to add to the current album.
- **force** (*bool*) – if `True`, change the song's `artist_name`, `album_name`, `album_year`, `album_type` attribute to match its parent.

Examples::

```
>>> album = Album('daughter', 'the wild youth')
>>> song = Song('daughter', 'youth')
>>> album.add_song(song)
>>> artist.get_albums()
>>> album
    Daughter: "The Wild Youth", Songs: 5
```

classmethod from_artist(*artist*, *album_name*)

Construct an Album from an Artist.

Parameters

- **artist** (`Artist`) – Artist to extract the album from.
- **album_name** (*string*) – name of the album.

Returns Album

classmethod from_url(*url*)

Construct an Album from an url.

Parameters **url** (`string`) – url.

Returns Album

Examples::

```
>>> album = Album.from_url('https://lyrics.fandom.com/wiki/Daughter:His_
↪Young_Heart_(2011)')
>>> album
```

get_songs()

Get a list of all songs made from an album.

Returns list

items()

Connect to `LyricWiki` server and scrape songs.

Returns yield Song

search_song(*song_name*)

Search a song from an album's playlist.

Parameters **song_name** (`string`) – name of the song to look for

Returns Song

set_album_type()

Shortcut to retrieve the type (Single, EP, Album) from an album's playlist.

songs()

Iterate through all songs within the current album.

Returns yield Song

to_json(*encode='ascii'*)

Encode a song in a JSON format, with full description.

Parameters **encode** (`string`) – format style. Recommended: `ASCII`.

Returns dict

4.3 lyricsfandom.music.song

Extract lyrics and songs from `https://lyrics.fandom.com/` website.

class `lyricsfandom.music.song.`**Song**(*artist_name*, *song_name*, *album_name=None*, *album_type=None*, *album_year=None*)

Defines a Song from `https://lyrics.fandom.com/`.

- `song_name`: name of the song.
- `url`: url of the song.

classmethod from_album(*album*, *song_name*)

Construct a Song from an Album.

Parameters

- **`album`** (`Album`) – album to extract the song from.
- **`song_name`** (*`string`*) – name of the song.

Returns Song

`classmethod from_artist`(*artist*, *song_name*)
Construct a Song from an artist.

Parameters

- **`artist`** (`Artist`) – artist to extract the song from.
- **`song_name`** (*`string`*) – name of the song.

Returns Song

`classmethod from_url`(*url*)
Construct a Song from an url.

Parameters **`url`** (*`string`*) – url of the lyrics song page.

`get_lyrics`()
Get lyrics from an URL address.

Returns string

`items`()
Iterate through items (usually it's empty).

Returns None

`to_json`(*encode='ascii'*)
Encode a song in a JSON format, with full description.

Parameters **`encode`** (*`string`*) – format style. Recommended: `ASCII`.

Returns dict

CHAPTER 5

Indices and tables

- genindex
- modindex
- search

Python Module Index

Index