scrapereads

Release 0.0.2

May 23, 2020

Getting Started

CHAPTER 1

Getting Started

1.1 Installation

Install *scrapereads* package from *PyPi*:

```
pip install scrapereads
```

Or from *GitHub*:

```
git clone https://github.com/arthurdjn/scrape-goodreads
cd scrape-goodreads
pip install .
```

1.2 Usage

You can use the simplified API to look for books and quotes.

Example:

```
from scrapereads import GoodReads

# Connect to the API
goodreads = GoodReads()
# Search for an author, by its id.
AUTHOR_ID = 3389
author = goodreads.search_author(3389)
author
```

Output:

```
Author: Stephen King
```

Then, you can search for book(s) too. Use `top_k=NUMBER` to look for an amount of books.

Example:

```
# Search for books
# Will look for the first 5 books
books = goodreads.search_books(AUTHOR_ID, top_k=5)
books
```

Output:

```
[Stephen King: "The Shining", 387 editions,
 Stephen King: "It", 313 editions,
 Stephen King: "The Stand", 230 editions,
 Stephen King: "Misery", 263 editions,
 Stephen King: "Carrie", 315 editions]
```

Finally, you can scrape for quotes.

Example:

```
# Search for quotes
quotes = goodreads.search_quotes(AUTHOR_ID, top_k=5)

for quote in quotes:
    print(quote)
    print()
```

Output:

```
“Books are a uniquely portable magic.”
-- Stephen King, from "On Writing: A Memoir Of The Craft"
  Likes: 16225, Tags: books, magic, reading

“If you don't have time to read, you don't have the time (or the tools) to write.␣
↪Simple as that.”
-- Stephen King
  Likes: 12565, Tags: reading, writing

“Get busy living or get busy dying.”
-- Stephen King, from "Different Seasons"
  Likes: 9014, Tags: life

“Books are the perfect entertainment: no commercials, no batteries, hours of␣
↪enjoyment for each dollar spent. What I wonder is why everybody doesn't carry a␣
↪book around for those inevitable dead spots in life.”
-- Stephen King
  Likes: 8667, Tags: books

“When his life was ruined, his family killed, his farm destroyed, Job knelt down on␣
↪the ground and yelled up to the heavens, "Why god? Why me?" and the thundering␣
↪voice of God answered, There's just something about you that pisses me off.”
-- Stephen King, from "Storm Of The Century"
  Likes: 7686, Tags: god, humor, religion
```

CHAPTER 2

Examples

The requests should be made from the API. Then, you can have access to authors, books, quotes.

2.1 GoodReads API

The API uses `id` to search authors, books or quotes.

```
from scrapereads import GoodReads

# Connect to the API
goodreads = GoodReads()
# Examples
# Author id: 3389    -> Stephen King
#            1077326 -> J. K. Rolling
AUTHOR_ID = 3389
author = goodreads.search_author(AUTHOR_ID)
```

You can have access to their attributes with:

```
# From an Author
author_name = author.author_name
author_id = author.author_id

# From a Book
author_name = book.author_name
author_id = book.author_id
book_name = book.book_name
book_id = book.book_id
edition = book.edition
year = book.year
ratings = book.ratings

# From a Quote
```

(continues on next page)

(continued from previous page)

```
author_name = quote.author_name
author_id = quote.author_id
quote_id = quote.quote_id
text = quote.text
tags = quote.tags
likes = quote.likes
```

2.2 Access data

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

```
# From an Author
AUTHOR_ID = 3389
author = goodreads.search_author(AUTHOR_ID)
books = author.get_books(top_k=None)
quotes = author.get_quotes(top_k=None)

# From a Book
BOOK_ID = 3048970
book = goodreads.search_book(AUTHOR_ID, BOOK_ID)
quotes = book.get_quotes()
```

Note:

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

```
# From and author
author = goodreads.search_author(AUTHOR_ID)
for quote in author.quotes():
    # quote is a Quote object
    print(quote)
```

From children classes (author –> book –> quote), you can retrieve data too:

```
author = goodreads.search_author(AUTHOR_ID)
quotes = author.get_quotes(top_k=5)
books = author.get_books(top_k=5)

# From a quote
quote = quotes[0]
book = quote.get_book()
author = quote.get_author()

# From an book
book = books[0]
author = book.get_author()
```

2.3 Cache System

scrapereads uses a cache system, that add dynamically data to an object while scraping. The main advantage, is that you don't have to scrape twice. The downside however, is that cache data is first loaded, meaning that it won't scrape

online if the cache is not empty. Turn this behavior off by setting `cache=False`:

```
author = goodreads.search_author(AUTHOR_ID)
quotes = author.get_quotes(top_k=5)
# WARNING: here, it will look for books saved in the cache (books added while␣
↪scraping quotes)
books = author.get_books(top_k=5)
# Turn it off, and search for books independently of the quotes (it will connect and␣
↪scrape on goodreads.com)
books = author.get_books(top_k=5, cache=False)
# Search for quotes regardless of previously added quotes (it will connect and scrape␣
↪on goodreads.com)
quotes = author.get_quotes(top_k=5, cache=False)
```

2.4 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 author
author = goodreads.search_author(AUTHOR_ID)
author_data = author.to_json(encode='ascii')
# Or directly
author_data = goodreads.get_author(AUTHOR_ID, encode='ascii')

# From an book
book = goodreads.search_book(AUTHOR_ID, BOOK_ID)
book_data = book.to_json(encode='ascii')
# Or directly
book_data = goodreads.get_books(AUTHOR_ID, top_k=NUMBER, encode=None)

# From a quote
quote_data = quote.to_json(encode='ascii')
# Or directly
quote_data = goodreads.get_quotes(AUTHOR_ID, top_k=NUMBER, encode=None)
```

CHAPTER 3

scrapereads

3.1 scrapereads.api

Simple API to connect and extract data from `Good Reads` servers.

class `scrapereads.api.`**GoodReads**(*verbose=False*, *sleep=0*, *user=None*)

Main API for *Good Reads* scrapping.

It basically wraps `Author`, `Book` and `Quote` classes.

static get_author(*author_id*, *encode=None*)

Get an author in a JSON format.

Parameters

- **author_id** (*string*) – name of the author.
- **encode** (*string*) – encode to ASCII format or not.

Returns dict

static get_books(*author_id*, *top_k=10*)

Get all books in a JSON format from an author.

Parameters

- **author_id** (*string*) – name of the author to get.
- **top_k** (*int*) – number of books to retrieve.

Returns list(dict)

static get_quotes(*author_id*, *top_k=10*)

Get all quotes in a JSON format from an author.

Parameters

- **author_id** (*string*) – name of the author to get.
- **top_k** (*int*) – number of quotes to retrieve.

Returns list(dict)

static search_author(*author_id*)
Search an author from *Good Reads* server.

Parameters **author_id** (*string*) – name of the author to get.

Returns Author

static search_book(*author_id*, *book_id*)
Search an book from *Good Reads* server.

Parameters

- **author_id** (*string*) – name of the author who made the book.
- **book_id** (*string*) – name of the book.

Returns Book

static search_books(*author_id*, *top_k=10*)
Search books in from an author.

Parameters

- **author_id** (*string*) – name of the author to get.
- **top_k** (*int*) – number of books to retrieve.

Returns list(Book)

static search_quotes(*author_id*, *top_k=50*)
Search quotes from *Good Reads* server.

Parameters

- **author_id** (*string*) – name of the author who made the quote.
- **top_k** (*int*) – number of quotes to retrieve.

Returns Quote

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

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

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

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

static 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 scrapereads.meta

Baseline class for *Good Reads* objects. This class handles connection to *Good Reads* server.

class scrapereads.meta.**AuthorMeta**(*author_id*, *author_name=None*)
Defines an abstract author, from the page info from `https://www.goodreads.com/`.

- `author_name`: name of the author.
- `author_id`: key id of the author.
- `base`: base page of *Good Reads*.
- `href`: href page of the author.
- `url`: url page of the author.

to_json()
Encode the author to a JSON format.

Returns dict

class scrapereads.meta.**BookMeta**(*author_id*, *book_id*, *book_name=None*, *author_name=None*, *edition=None*, *year=None*)
Abstract Book class, used as baseline.

- `author_name`: name of the author.
- `author_id`: key id of the author.
- `book_name`: name of the book.
- `book_id`: key if of the book.
- `year`: year of publication of the book.
- `edition`: edition of the book.
- `base`: base page of *Good Reads*.
- `href`: href page of the book.
- `url`: url page of the book.

get_author()
Get the author pointing to the quote.

Returns Author

register_author(*author*)
Point a quote to an Author.

Parameters **author** (Author) – author to link the quote.

to_json(*encode='ascii'*)
Encode the book to a JSON format.

Returns dict

class scrapereads.meta.**GoodReadsMeta**
Defines the base of all *Good Reads* objects, that scrape and extract online data.

- `base`: base page of the *Good Reads*.
- `href`: href of a page.
- `url`: url page of a *Good Reads* element.

connect(*href=None*)
Connect to a *Good Reads* page.

Parameters **href** (*string, optional*) – if provided, connect to the page reference, else connect to the main page.

Returns bs4.element.Tag

class scrapereads.meta.**QuoteMeta**(*author_id*, *quote_id*, *quote_name=None*, *text=None*, *author_name=None*, *tags=None*, *likes=None*)

Defines a quote from the quote page from `https://www.goodreads.com/author/quotes/`.

- `quote_id`: nif of the quote.
- `book_name`: name of the book / title.
- `book_name`: name of the book / title.
- `book_name`: name of the book / title.
- `quote`: text.

get_author()

Get the author pointing to the quote.

Returns Author

get_book()

Get the book pointing to the quote.

Returns Book

register_author(*author*)

Point a quote to an Author.

Parameters **author** (`Author`) – author to link the quote.

register_book(*book*)

Point a quote to a Book.

Parameters **book** (`Book`) – book to link the quote.

to_json(*encode='ascii'*)

Encode the quote to a JSON format.

Returns dict

3.3 scrapereads.connect

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

scrapereads.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 scrapereads.scrape

Scrape quotes, books and authors from `Good Reads` website.

scrapereads.scrape.**get_author_book_author**(*book_tr*)

Get the author <a> element from a table <tr> element.

Parameters **book_tr** (*bs4.element.Tag*) – <tr> book element.

Returns author name <a> element.

Return type bs4.element.Tag

Examples::

```
>>> for book_tr in scrape_author_books(soup):
...     book_author = get_author_book_author(book_tr)
...     print(book_author.text, book_author.get('href'))
    Sylvia Plath https://www.goodreads.com/author/show/4379.Sylvia_Plath
    Sylvia Plath https://www.goodreads.com/author/show/4379.Sylvia_Plath
    Sylvia Plath https://www.goodreads.com/author/show/4379.Sylvia_Plath
    Sylvia Plath https://www.goodreads.com/author/show/4379.Sylvia_Plath
    Sylvia Plath https://www.goodreads.com/author/show/4379.Sylvia_Plath
```

scrapereads.scrape.**get_author_book_date**(*book_tr*)

Get the published date from a table <tr> element from an author page.

Parameters **book_tr** (*bs4.element.Tag*) – <tr> book element.

Returns date of publication

Return type int

Examples::

```
>>> for book_tr in scrape_author_books(soup):
...     book_date = get_author_book_date(book_tr)
...     print(book_date)
    None
    None
    1958
    2009
    ...
```

scrapereads.scrape.**get_author_book_edition**(*book_tr*)

Get the edition <a> element from a table <tr> element from an author page.

Parameters **book_tr** (*bs4.element.Tag*) – <tr> book element.

Returns book edition <a> element.

Return type bs4.element.Tag

Examples::

```
>>> for book_tr in scrape_author_books(soup):
...     book_edition = get_author_book_edition(book_tr)
...     if book_edition:
...         print(book_edition.text, book_edition.get('href'))
...         print()
    493 editions /work/editions/1385044-the-bell-jar
    80 editions /work/editions/1185316-ariel
    30 editions /work/editions/1003095-the-collected-poems
    45 editions /work/editions/3094683-the-unabridged-journals-of-sylvia-plath
    ...
```

scrapereads.scrape.**get_author_book_ratings**(*book_tr*)

Get the ratings element from a table <tr> element from an author page.

Parameters **book_tr** (*bs4.element.Tag*) – <tr> book element.

Returns ratings element.

Return type bs4.element.Tag

Examples::

```
>>> for book_tr in scrape_author_books(soup):
...     ratings_span = get_author_book_ratings(book_tr)
...     print(ratings_span.contents[-1])
    4.55 avg rating -- 2,414 ratings
    3.77 avg rating -- 1,689 ratings
    4.28 avg rating -- 892 ratings
    4.54 avg rating -- 490 ratings
    ...
```

scrapereads.scrape.**get_author_book_title**(*book_tr*)

Get the book title <a> element from a table <tr> element from an author page.

Parameters **book_tr** (*bs4.element.Tag*) – <tr> book element.

Returns book title <a> element.

Return type bs4.element.Tag

Examples::

```
>>> for book_tr in scrape_author_books(soup):
...     book_title = get_author_book_title(book_tr)
...     print(book_title.text.strip(), book_title.get('href'))
    The Bell Jar /book/show/6514.The_Bell_Jar
    Ariel /book/show/395090.Ariel
    The Collected Poems /book/show/31426.The_Collected_Poems
    The Unabridged Journals of Sylvia Plath /book/show/11623.The_Unabridged_
↪Journals_of_Sylvia_Plath
```

scrapereads.scrape.**get_author_desc**(*soup*)

Get the author description / biography.

Parameters **soup** (*bs4.element.Tag*) – connection to the author page.

Returns long description of the author.

Return type str

Examples::

```
>>> from scrapereads import connect
>>> url = 'https://www.goodreads.com/author/show/1077326'
>>> soup = connect(url)
>>> get_author_desc(soup)
    See also: Robert Galbraith
    Although she writes under the pen name J.K. Rowling, pronounced like␣
↪rolling,
```

(continues on next page)

(continued from previous page)

```
    her name when her first Harry Potter book was published was simply Joanne␣
↪Rowling.
    ...
```

scrapereads.scrape.**get_author_info**(*soup*)
Get all information from an author (genres, influences, website etc.).

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

Returns dict

scrapereads.scrape.**get_author_name**(*soup*)
Get the author's name from its main page.

Parameters **soup** (*bs4.element.Tag*) – connection to the author page.

Returns name of the author.

Return type string

Examples::

```
>>> from scrapereads import connect
>>> url = 'https://www.goodreads.com/author/show/1077326'
>>> soup = connect(url)
>>> get_author_name(soup)
    J.K. Rowling
```

scrapereads.scrape.**get_book_quote_page**(*soup*)
Find the <a> element pointing to the quote page of a book.

Parameters **soup** (*bs4.element.Tag*) –

Returns:

scrapereads.scrape.**get_quote_author_name**(*quote_div*)
Get the author's name from a quote element.

Parameters **quote_div** (*bs4.element.Tag*) – quote element from a quote page.

Returns string

scrapereads.scrape.**get_quote_book**(*quote_div*)
Get the reference (book) from a quote element.

Parameters **quote_div** (*bs4.element.Tag*) – quote element from a quote page.

Returns bs4.element.Tag

scrapereads.scrape.**get_quote_likes**(*quote_div*)
Get the likes <a> tag from a quote element.

Parameters **quote_div** (*bs4.element.Tag*) – quote element from a quote page.

Returns <a> tag for likes.

Return type bs4.element.Tag

scrapereads.scrape.**get_quote_name_id**(*quote_div*)
Get the name and id of a quote element.

Parameters **quote_div** (*bs4.element.Tag*) – quote element from a quote page.

Returns id and name.

Return type tuple

scrapereads.scrape.**get_quote_text**(*quote_div*)
Get the text from a quote element.

Parameters **quote_div** (*bs4.element.Tag*) – quote element to extract the text.

Returns string

scrapereads.scrape.**scrape_author_books**(*soup*)
Retrieve books from an author's page.

Parameters **soup** (*bs4.element.Tag*) – connection to an author books page.

Returns <tr> element.

Return type yield bs4.element.Tag

scrapereads.scrape.**scrape_quote_tags**(*quote_div*)
Scrape tags from a quote element.

Parameters **quote_div** (*bs4.element.Tag*) – quote element from a quote page.

Returns yield <a> tags

scrapereads.scrape.**scrape_quotes**(*soup*)
Retrieve all quote element from a quote page.

Parameters **soup** (*bs4.element.Tag*) – connection to the quote page.

Returns yield bs4.element.Tag

scrapereads.scrape.**scrape_quotes_container**(*soup*)
Get the quote container from a quote page.

Parameters **soup** (*bs4.element.Tag*) – connection to the quote page.

Returns bs4.element.Tag

3.5 scrapereads.utils

Functional functions to process names and data.

scrapereads.utils.**clean_num**(*quote*)
Remove romans numbers from a quote.

Parameters **quote** (*string*) – quote.

Returns string

scrapereads.utils.**name_to_goodreads**(*name*)
Process and convert names in scrapereads format.

Parameters **name** (*string*) – name of an author.

Returns string

scrapereads.utils.**num2roman**(*num*)
Convert a number to roman's format.

Parameters **num** (*int*) – number to convert.

Returns string

scrapereads.utils.**parse_author_href**(*href*)
Split an href and retrieve the author's name and its key.

Parameters **href** (*string*) – Good Reads href pointing to an author page.

Returns author's name and key.

Return type tuple

scrapereads.utils.**process_quote_text**(*quote_text*)
Clean up the text from a quote element.

Parameters **quote_text** (*string*) – quote text to clean.

Returns string

scrapereads.utils.**remove_punctuation**(*string_punct*)
Remove punctuation from a string.

Parameters **string_punct** (*string*) – string with punctuation.

Returns string

scrapereads.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

scrapereads.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

scrapereads.utils.**to_ascii**(*text*)
Convert a text to ASCII format.

Parameters **text** (*string*) – text to process.

Returns string

CHAPTER 4

scrapereads.reads

4.1 scrapereads.reads.author

Defines an Author from `Good Reads`. Connect to https://www.goodreads.com/ to extract quotes and books from famous authors.

class scrapereads.reads.author.**Author**(*author_id*, *author_name=None*)

Defines an author, from the page info from `https://www.goodreads.com/`.

- `name`: name of the author.
- `key`: key id of the author.
- `url`: url page of the author.

add_book(*book*)

Add a book to an Author.

Parameters **book** (`Book`) – book or book's name to add.

add_quote(*quote*)

Add a quote to an Author.

Parameters **quote** (`Quote or string`) – quote or text to add.

books(*cache=True*)

Get all books from an author address. This function extract online data from *Good Reads* if nothing is already saved in the cache.

Parameters **cache** (`bool`) – if `True`, will look for cache items only (and won't scrape online).

Returns yield Quote

classmethod from_url(*url*)

Construct the class from an url.

Parameters **url** (`string`) – url.

Returns Author

get_books(*top_k=None*, *cache=True*)
Get all books from an author address.

Parameters

- **top_k** (*int*) – number of books to return.
- **cache** (*bool*) – if `True`, will look for cache items only (and won't scrape online).

Returns list(Book)

get_info()
Get author information (genres, influences, description etc.)

Returns dict

get_quotes(*lang=None*, *top_k=None*, *cache=True*)
Get all quotes from an author address.

Parameters

- **lang** (*string*) – language to pick up quotes.
- **top_k** (*int*) – number of quotes to retrieve (ordered by popularity).
- **cache** (*bool*) – if `True`, will look for cache items only (and won't scrape online).

Returns list(Quote)

get_similar_authors(*top_k=None*)
Get similar artists from the author.

Parameters **top_k** (*int*) – number of authors to retrieve (ordered by popularity).

Returns list(Author)

quotes(*cache=True*)
Yield all quotes from an author address. This function extract online data from *Good Reads* if nothing is already saved in the cache.

Parameters **cache** (*bool*) – if `True`, will look for cache items only (and won't scrape online).

Returns yield Quote

search_book(*book_id*, *attr='book_id'*, *cache=True*)
Search a book from the books saved in the author's cache.

Parameters

- **book_id** (*string*) – book id (or name) to look for.
- **attr** (*string, optional*) – attribute to search the book from. Options are `'book_id'` and `'book_name'`
- **cache** (*bool*) – if `True`, will look for cache items only (and won't scrape online).

Returns Book

search_quote(*quote_id*, *attr='quote_id'*, *cache=True*)
Search a quote from the books saved in the author's cache.

Parameters

- **quote_id** (*string*) – quote'id to look for.
- **attr** (*string, optional*) – attribute to search the quote from. Options are `'quote_id'` and `'quote_name'`

- **cache** (*bool*) – if `True`, will look for cache items only (and won't scrape online).

Returns Book

to_json(*encode=None*)
Encode the author to a JSON format.

Parameters **encode** (*string*) – encode to ASCII format or not.

Returns dict

4.2 scrapereads.reads.book

Defines a book from an Author.

class scrapereads.reads.book.**Book**(*author_id*, *book_id*, *book_name=None*, *author_name=None*, *edition=None*, *year=None*, *ratings=None*)

add_quote(*quote*)
Add a quote to the Book, that will be saved in the cache.

Parameters **quote** (`Quote`) – quote to add.

get_quotes(*lang=None*, *top_k=None*, *cache=True*)
Get all quotes from a book address.

Parameters

- **lang** (*string*) – language to pick up quotes.
- **top_k** (*int*) – number of quotes to retrieve (ordered by popularity).
- **cache** (*bool*) – if `True`, will look for cache items only (and won't scrape online).

Returns list(Quote)

quotes(*cache=True*)
Yield all quotes from a book address. This function extract online data from *Good Reads* if nothing is already saved in the cache.

Parameters **cache** (*bool*) – if `True`, will look for cache items only (and won't scrape online).

Returns yield Quote

to_json(*encode='ascii'*)
Encode the book to a JSON format.

Returns dict

4.3 scrapereads.reads.quote

Defines a quote from an Author.

class scrapereads.reads.quote.**Quote**(*author_id*, *quote_id*, *text=''*, *quote_name=None*, *author_name=None*, *tags=None*, *likes=None*)
Defines a quote from the quote page from `https://www.goodreads.com/author/quotes/`.

to_json(*encode='ascii'*)
Encode the quote to a JSON format.

Returns dict

CHAPTER 5

Indices and tables

- genindex
- modindex
- search

Python Module Index

S

Index

N

P

Q

R

S

T