isbntools Documentation

Release 4.3.14

Alexandre Conde

Mar 06, 2018

Contents

isbntools provides several useful methods and functions to validate, clean, transform, hyphenate and get metadata for ISBN strings.

Contents:

CHAPTER 1

Usage

For the end user several scripts are provided to use **from the command line**:

```
$ to_isbn10 ISBN13
```

transforms an ISBN13 number to ISBN10.

```
$ to_isbn13 ISBN10
```

transforms an ISBN10 number to ISBN13.

```
$ isbn_info ISBN
```

gives you the *group identifier* of the ISBN.

```
$ isbn_mask ISBN
```

masks (hyphenate) an ISBN (split it by identifiers).

```
$ isbn_meta ISBN [wcat|goob|openl|merge] [bibtex|...] [YOUR_APIKEY_TO_SERVICE]
```

gives you the main metadata associated with the ISBN, `wcat` uses **worldcat.org** (**no key is needed**), `goob` uses the **Google Books service** (**no key is needed**), `openl` uses the **OpenLibrary.org** api (**no key is needed**), `merge` uses a merged record of `wcat` and `goob` records (**no key is needed**) and **is the default option** (you only have to enter, e.g. `isbn_meta 9780321534965`). You can enter API keys and set preferences in the file `isbntools.conf` in your `$HOME\.isbntools` directory (UNIX). For Windows, you should look at `%APPDATA%/isbntools/isbntools.conf`. The output can be formatted as `bibtex`, `csl` (CSL-JSON), `msword`, `endnote`, `refworks`, `opf` or `json` (BibJSON) bibliographic formats.

```
$ isbn_editions ISBN
```

gives the collection of ISBNs that represent a given book (uses **Open Library** and **LibraryThing**).

```
$ isbn_validate ISBN
```

validates ISBN10 and ISBN13.

```
$ ... | isbn_stdin_validate
```

to use with *posix pipes* (e.g. `cat FILE_WITH_ISBNs | isbn_stdin_validate`).

> **TIP** Suppose you want to extract the ISBN of a pdf eboook (MYEBOOK.pdf). Install pdfminer and then enter in a command line:
>
> ```
> $ pdf2txt.py -m 5 MYEBOOK.pdf | isbn_stdin_validate
> ```

```
$ isbn_from_words "words from title and author name"
```

a *fuzzy* script that returns the *most probable* ISBN from a set of words! (You can verify the result with `isbn_meta`)!

```
$ isbn_goom "words from title and author name"␣
↪[bibtex|csl|opf|msword|endnote|refworks|json]
```

a script that returns from **Google Books multiple references**.

```
$ isbn_doi ISBN
```

returns the doi's ISBN-A code of a given ISBN.

```
$ isbn_ean13 ISBN
```

returns the EAN13 code of a given ISBN.

```
$ isbn_ren FILENAME
```

renames (using metadata) files in the **current directory** that have ISBNs in their filename (e.g. `isbn_ren 1783559284_book.epub`, `isbn_ren "*.pdf"`).

> Enter `isbn_ren` to see many other options.

```
$ isbntools
```

writes version and copyright notice and **checks if there are updates**.

With

```
$ isbn_repl
```

you will get a REPL with history, autocompletion, fuzzy options, redirection and access to the shell.

Following is a typical session:

```
$ isbn_repl

    Welcome to the isbntools 4.2.2 REPL.
    ** For help type 'help' or '?'
    ** To exit type 'exit' :)
    ** To run a shell command, type '!<shellcmnd>'
    ** Use '#' in place of the last ISBN

$ isbn> ?

Commands available (type ?<command> to get help):
-------------------------------------------------
```

```
BIBFORMATS  conf   doi      editions    goom  mask       to_isbn13
PROVIDERS   cover  doi2tex  exit        help  meta       validate
audit       desc   ean13    from_words  info  to_isbn10

$ isbn> meta 9780156001311 tex
@book{9780156001311,
     title = {The Name Of The Rose},
    author = {Umberto Eco},
      isbn = {9780156001311},
      year = {1994},
 publisher = {Harcourt Brace}
}
$ isbn> meta 9780156001311 tex >>myreferences.bib
$ isbn> !ls
myreferences.bib
$ isbn> desc #
It is the year 1327. Franciscans in an Italian abbey are suspected of
heresy, but Brother William of Baskerville's investigation is suddenly
overshadowed by seven bizarre deaths. Translated by William Weaver. A Helen
and Kurt Wolff Book
$ isbn> cover #
     thumbnail:  http://books.google.com/books/content?id=PVVyuD1UY1wC&
↪printsec=frontcover&img=1&zoom=1
smallThumbnail:  http://books.google.com/books/content?id=PVVyuD1UY1wC&
↪printsec=frontcover&img=1&zoom=5
$ isbn> exit
bye
```

Within REPL many of the operations are faster.

Many more scripts could be written with the `isbntools` library, using the methods for extraction, cleaning, validation and standardization of ISBNs.

Just for fun, suppose I want the *most spoken about* book with certain words in his title. For a *quick-and-dirty solution*, enter the following code in a file and save it as `isbn_tmsa_book.py`.

```
#!/usr/bin/env python
import sys
from isbntools import *

query = sys.argv[1].replace(' ', '+')
isbn = isbn_from_words(query)

print("The ISBN of the most `spoken-about` book with this title is %s" % isbn)
print("")
print("... and the book is:")
print("")
print((meta(isbn)))
```

Then in a command line (in the same directory):

```
$ python isbn_tmsa_book.py 'noise'
```

In my case I get:

```
The ISBN of the most `spoken-about` book with this title is 9780143105985

... and the book is:
```

```
{'Publisher': u'Penguin Books', 'Language': u'eng', 'Title': u'White noise',
'Year': u'2009', 'ISBN-13': u'9780143105985', 'Authors': u'Don DeLillo ;
introduction by Richard Powers.'}
```

Have fun!

CHAPTER 2

Install

From the command line enter (in some cases you have to precede the command with `sudo`):

```
$ pip install isbntools
```

or:

```
$ easy_install isbntools
```

or:

```
$ pip install isbntools-4.3.14.tar.gz
```

(first you have to download the file!)

You should check if the install was successful, by enter:

```
$ isbntools
```

CHAPTER 3

For Devs

3.1 API's Main Namespaces

In the namespace `isbntools.app` you have access to the core methods: `is_isbn10`, `is_isbn13`, `to_isbn10`, `to_isbn13`, `canonical`, `clean`, `notisbn`, `get_isbnlike`, `get_canonical_isbn`, `mask`, `meta`, `info`, `editions`, and `isbn_from_words`. The exceptions raised by these methods can all be catched using `ISBNToolsException`.

You can use advanced features by using the classes and functions exposed in namespace `isbnlib.dev`, namely:

- `WEBService` a class that handles the access to web services (just by passing an url) and supports `gzip`. You can subclass it to extend the functionality... but probably you don't need to use it! It is used in the next class.
- `WEBQuery` a class that uses `WEBService` to retrieve and parse data from a web service. You can build a new provider of metadata by subclassing this class. His main methods allow passing custom functions (*handlers*) that specialize them to specific needs (`data_checker` and `parser`).
- `Metadata` a class that structures, cleans and 'validates' records of metadata. His method `merge` allows to implement a simple merging procedure for records from different sources. The main features can be implemented by a call to `stdmeta` function!
- `vias` exposes several functions to put calls to services, just by passing the name and a pointer to the service's `query` function. `vias.parallel` allows to put threaded calls, however doesn't implement throttling! You can use `vias.serial` to make serial calls and `vias.multi` to use several cores. The default is `vias.serial`, but you can change that in the conf file.
- `bouth23` a small module to make it possible the code to run in **bouth** python 2 and python 3.

The exceptions raised by these methods can all be catched using `ISBNLibDevException`.

In `isbnlib.dev.helpers` you can find several methods, that we found very useful, but you should consider them as beta software. They can change a lot in the future.

Finally, `isbntools.conf` provides methods to edit the configuration file and helpers to work with isbntools's modules.

> **WARNING**: If you inspect the library, you will see that there are a lot of private modules (their name starts with '_'). These modules **should not** be accessed directly since, with high probability, your program will break with a further version of the library!
>
> You should access only methods in the API's `isbntools`, `isbnlib.dev`, `isbnlib.dev.helpers` and `isbntools.conf`

All these classes follow a simple design pattern and, if you follow it, will be very easy to integrate your classes with the rest of the lib.

3.2 Plugins

The support for pluggins **was dropped** from `isbntools`, however continues to support modules! The reason is that `isbnlib` now supports plugins for metadata and new formatters.

3.3 Merge Metadata

The original quality of metadata, at the several services, is not very good! If you need high quality metadata in your app, the only solution is to use *polling & merge* of several providers **and** a **lot** of cleaning and standardization for fields like `Authors` and `Publisher`. A *simple merge* provider is now the default in `isbn_meta` (and `isbntools.meta`). It gives priority to `wcat` but overwrites the `Authors` field with the value from `goob`. Uses the `merge` method of `Metadata` and *serial* calls to services by default (faster for fast Internet connections). You can change that, by setting `VIAS_MERGE=parallel` or `VIAS_MERGE=multi` (see note below).

> **Take Note**: These classes are optimized for one-calls to services and not for batch calls.

3.4 Just an ISBN lib!

If you just want to integrate the lib in your project, you have several options, depending on your needs...

1. If you need only basic manipulation of ISBNs (validation, transforming, extraction, hyphenation, ...) but not custom metadata or file renaming, then you don't need a conf file. Just use the methods in `isbntools`. But probably you are better served with isbnlib.
2. If you rely heavily in metadata (or file renaming) and don't want to implement caching yourself, then you **need** an `isbntools.conf` file in a directory were your program could write. You can use `isbntools.conf` to programatically manipulate the conf file.
3. If you want to vendorize the lib you should take a careful look at `setup.py` and maybe this package (datafolder) could help!

Anyway, you could use the `isbn_...` scripts in the `isbntools/bin` directory as examples on how to use the library and as debugger tools for your implementation.

> **Don't forget** to take a look at isbnlib.

You can browse the code at GitHub.

CHAPTER 4

Conf File

You can enter API keys and set preferences in the file `isbntools.conf` in your `$HOME/.isbntools` directory (UNIX). For Windows, you should look at `%APPDATA%/isbntools/isbntools.conf` (**create these, directory and file, if don't exist** [Now just enter `isbn_conf make`]). The file should look like:

```
[MISC]
REN_FORMAT={firstAuthorLastName}{year}_{title}_{isbn}
DEBUG=False

[SYS]
URLOPEN_TIMEOUT=10
THREADS_TIMEOUT=12

[SERVICES]
DEFAULT_SERVICE=merge
VIAS_MERGE=parallel
ISBNDB_API_KEY=
```

The values are self-explanatory!

> **NOTE** If you are running `isbntools` inside a virtual environment, the `isbntools.conf` file will be inside folder `isbntools` at the root of the environment.

The easier way to manipulate these files is by using the script `isbn_conf`. At a terminal enter:

```
$ isbn_conf show
```

to see the current conf file.

This script has many options that allow a controlled editing of the conf file. Just enter `isbn_conf` for help.

CHAPTER 5

Known Issues

1. The `meta` method and the `isbn_meta` script sometimes give a wrong result (this is due to errors on the chosen service), in alternative you could try one of the others services.
2. The `isbntools` works internally with unicode, however this doesn't solve errors of lost information due to bad encode/decode at the origin!
3. Periodically, agencies, issue new blocks of ISBNs. The range of these blocks is on a database that `mask` uses. So it could happen, if you have a version of `isbntools` that is too old, `mask` doesn't work for valid (recent) issued ISBNs. The solution? **Update isbntools often**!
4. Calls to metadata services are cached by default. If you don't want this feature, just enter `isbn_conf setopt cache no`. If by any reason you need to clear the cache, just enter `isbn_conf delcache`.

Any issue that you would like to report, (if you are a developer) please do it at github or at stackoverflow with tag **isbntools**, (if you are an end user) at twitter.

CHAPTER 6

ISBN

To know about ISBN:

- http://en.wikipedia.org/wiki/International_Standard_Book_Number
- http://www.isbn-international.org/

CHAPTER 7

Code

7.1 Search

Search/Browse the code at github

7.2 Status

7.3 How to Contribute

`isbntools` has a very small code base, so it is a good project to begin your adventure in open-source… and it is an app, a lib and a framework at the same time, so you will find plenty of opportunities to contribute.

7.3.1 Main Steps

1. Make sure you have a GitHub account
2. Submit a ticket for your issue or idea, on GitHub issues (if possible wait for some feedback before any serious commitment… :)
3. Fork the repository on GitHub
4. `pip install -r requirements-dev.txt`
5. Do your code… (**remember the code must run on python 2.6, 2.7, 3.3, 3.4, pypy and be OS independent**) (you will find `travis-ci.org` very handy for this!)
6. Write tests for your code using `nose` and put then in the directory `isbnools/test`

7. Pass **all tests** and with **coverage > 90%**. Check the coverage in Coveralls.
8. **Check if all requirements are fulfilled**!
9. Make a pull request on github...

7.3.2 Important

If you don't have experience in these issues, don't be put off by these requirements, see them as a learning opportunity. Thanks!

> For full instructions read the CONTRIBUTING doc.

CHAPTER 8

Extra Functionality

Extra functionality is provided, but the user needs to install external libraries or signing for API Keys with specific services.

Check in pypi for packages with name **isbntools.contrib. . .**