BibtexParser Documentation

Release 0.6.2

F. Boulogne

Sep 27, 2017

Contents

Author François Boulogne and other contributors

Download Stable version

Developer's corner github.com project

Generated Sep 27, 2017

Version 0.6.2

BibtexParser is a python library to parse bibtex files. The code is tested with unittests.

If you use BibtexParser for your project, feel free to send me an email. I would be happy to hear that and to mention your project in the documentation.

Contents:

CHAPTER 1

How to install?

Requirements

- python **2.7**
- python **3.3** or newer

Package manager

- [Archlinux](https://aur.archlinux.org/packages/python-bibtexparser/)

PyPI

See Pypi

To install with pip:

```
pip install bibtexparser
```

Manual installation

Download the archive.

```
python setup.py install --root=/usr/local/bin
```

CHAPTER 2

How to test?

This page briefly describes how to run the test suite. This is useful for contributors but also for packagers.

Virtualenv

You can make a virtualenv. I like pew for that because the API easier.

The first time, you need to make a virtualenv

```
pew mkproject bibtexparser
python setup.py install
nosetest
```

If you already have a virtualenv

```
pew workon bibtexparser
python setup.py install
nosetest
```

Tox

The advantage of Tox is that you can build and test the code against several versions of python. Of course, you need tox to be installed on your system. The configuration file is tox.ini, in the root of the project.

```
tox # and nothing more :)
```

CHAPTER 3

Tutorial

Preparing a BibTeX file

Prepare a BibTeX sample file for illustration purpose:

```
bibtex = """@ARTICLE{Cesar2013,
  author = {Jean César},
  title = {An amazing title},
  year = {2013},
  month = jan,
  volume = {12},
  pages = {12--23},
  journal = {Nice Journal},
  abstract = {This is an abstract. This line should be long enough to test
     multilines...},
  comments = {A comment},
  keywords = {keyword1, keyword2}
}
"""

with open('bibtex.bib', 'w') as bibfile:
    bibfile.write(bibtex)
```

Parsing the file into a bibliographic database object

OK. Everything is in place. Let's parse the BibTeX file.

```
import bibtexparser

with open('bibtex.bib') as bibtex_file:
    bibtex_str = bibtex_file.read()
```

```
bib_database = bibtexparser.loads(bibtex_str)
print(bib_database.entries)
```

It prints a list of dictionaries for reference entries, for example books, articles:

```
[{'journal': 'Nice Journal',
  'comments': 'A comment',
  'pages': '12--23',
  'month': 'jan',
  'abstract': 'This is an abstract. This line should be long enough to␣
↪test\nmultilines...',
  'title': 'An amazing title',
  'year': '2013',
  'volume': '12',
  'ID': 'Cesar2013',
  'author': 'Jean César',
  'keyword': 'keyword1, keyword2',
  'ENTRYTYPE': 'article'}]
```

Note that, by convention, uppercase keys are auto-generated data, while lowercase keys come from the original bibtex file.

Alternatively, you can parse a file-like object directly like this:

```
import bibtexparser

with open('bibtex.bib') as bibtex_file:
    bib_database = bibtexparser.load(bibtex_file)
```

Creating a BibTeX file or string

The bibliographic data can be converted back into a BibTeX file like this:

```
import bibtexparser

bibtex_str = bibtexparser.dumps(bib_database)
```

Customizations

By default, the parser does not alter the content of each field and keeps it as a simple string. There are many cases where this is not desired. For example, instead of a string with a multiple of authors, it could be parsed as a list.

To modify field values during parsing, a callback function can be supplied to the parser which can be used to modify BibTeX entries. The library includes several functions which may be used. Alternatively, you can read them to create your own functions.

```
import bibtexparser
from bibtexparser.bparser import BibTexParser
from bibtexparser.customization import *

# Let's define a function to customize our entries.
# It takes a record and return this record.
def customizations(record):
```

```
    """Use some functions delivered by the library

    :param record: a record
    :returns: -- customized record
    """
    record = type(record)
    record = author(record)
    record = editor(record)
    record = journal(record)
    record = keyword(record)
    record = link(record)
    record = page_double_hyphen(record)
    record = doi(record)
    return record

with open('bibtex.bib') as bibtex_file:
    parser = BibTexParser()
    parser.customization = customizations
    bib_database = bibtexparser.load(bibtex_file, parser=parser)
    print(bib_database.entries)
```

If you think that you have a customization which could be useful to others, please share with us!

Accents and weird characters

Your bibtex may content accents and specific characters. They are sometimes coded like this `\'{e}` but this is not the correct way, `{\'e}` is prefered. Moreover, you may want to manipulate é. There is different situations:

- Case 1: you plan to use this library to work with latex and you assume that the original bibtex is clean. You have nothing to do.
- Case 2: you plan to use this library to work with latex but your bibtex is not really clean.

```
import bibtexparser
from bibtexparser.bparser import BibTexParser
from bibtexparser.customization import homogeneize_latex_encoding

with open('bibtex.bib') as bibtex_file:
    parser = BibTexParser()
    parser.customization = homogeneize_latex_encoding
    bib_database = bibtexparser.load(bibtex_file, parser=parser)
    print(bib_database.entries)
```

- Case 3: you plan to use this library to work with something different and your bibtex is not really clean. Then, you probably want to use unicode.

```
import bibtexparser
from bibtexparser.bparser import BibTexParser
from bibtexparser.customization import convert_to_unicode

with open('bibtex.bib') as bibtex_file:
    parser = BibTexParser()
    parser.customization = convert_to_unicode
    bib_database = bibtexparser.load(bibtex_file, parser=parser)
    print(bib_database.entries)
```

Note: if you want to mix different customization functions, you can write your own function.

Using the writer

In the first section we prepared a BibTeX sample file, we can prepare the same file using pure python and the `BibTexWriter` class.

```
from bibtexparser.bwriter import BibTexWriter
from bibtexparser.bibdatabase import BibDatabase

db = BibDatabase()
db.entries = [
    {'journal': 'Nice Journal',
     'comments': 'A comment',
     'pages': '12--23',
     'month': 'jan',
     'abstract': 'This is an abstract. This line should be long enough to␣
↪test\nmultilines...',
     'title': 'An amazing title',
     'year': '2013',
     'volume': '12',
     'id': 'Cesar2013',
     'author': 'Jean César',
     'keyword': 'keyword1, keyword2',
     'type': 'article'}]

writer = BibTexWriter()
with open('bibtex.bib', 'w') as bibfile:
    bibfile.write(writer.write(db))
```

This code generates the following file:

```
@article{Cesar2013,
 abstract = {This is an abstract. This line should be long enough to test
multilines...},
 author = {Jean César},
 comments = {A comment},
 journal = {Nice Journal},
 keyword = {keyword1, keyword2},
 month = {jan},
 pages = {12--23},
 title = {An amazing title},
 volume = {12},
 year = {2013}
}
```

The writer also has several flags that can be enabled to customize the output file. For example we can use `indent` and `comma_first` to customize the previous entry, first the code:

```
from bibtexparser.bwriter import BibTexWriter
from bibtexparser.bibdatabase import BibDatabase

db = BibDatabase()
db.entries = [
    {'journal': 'Nice Journal',
     'comments': 'A comment',
```

```
     'pages': '12--23',
     'month': 'jan',
     'abstract': 'This is an abstract. This line should be long enough to␣
↪test\nmultilines...',
     'title': 'An amazing title',
     'year': '2013',
     'volume': '12',
     'id': 'Cesar2013',
     'author': 'Jean César',
     'keyword': 'keyword1, keyword2',
     'type': 'article'}]

writer = BibTexWriter()
writer.indent = '    '     # indent entries with 4 spaces instead of one
writer.comma_first = True  # place the comma at the beginning of the line
with open('bibtex.bib', 'w') as bibfile:
    bibfile.write(writer.write(db))
```

This code results in the following, customized, file:

```
@article{Cesar2013
,    abstract = {This is an abstract. This line should be long enough to test
multilines...}
,    author = {Jean César}
,    comments = {A comment}
,    journal = {Nice Journal}
,    keyword = {keyword1, keyword2}
,    month = {jan}
,    pages = {12--23}
,    title = {An amazing title}
,    volume = {12}
,    year = {2013}
}
```

Flags to the writer object can modify not only how an entry is printed but how several BibTeX entries are sorted and separated. See the *bibtexparser: API* for the full list of flags.

CHAPTER 4

bibtexparser: API

`bibtexparser` — Parsing and writing BibTeX files

BibTeX <http://en.wikipedia.org/wiki/BibTeX> is a bibliographic data file format.

The *bibtexparser* module provides parsing and writing of BibTeX files functionality. The API is similar to the `json` module. The parsed data is returned as a simple `BibDatabase` object with the main attribute being `entries` representing bibliographic sources such as books and journal articles.

Parsing is a simple as:

```
>>>> import bibtexparser
>>>> with open('bibtex.bib') as bibtex_file:
>>>>    bibtex_database = bibtexparser.load(bibtex_file)
```

And writing:

```
>>>> import bibtexparser
>>>> with open('bibtex.bib', 'w') as bibtex_file:
>>>>     bibtexparser.dump(bibtex_database, bibtex_file)
```

`bibtexparser.`**`load`**(*bibtex_file*, *parser=None*)

Load `BibDatabase` object from a file

Parameters

- **`bibtex_file`** (*file*) – input file to be parsed
- **`parser`** (*BibTexParser*) – custom parser to use (optional)

Returns bibliographic database object

Return type *BibDatabase*

`bibtexparser.`**`loads`**(*bibtex_str*, *parser=None*)

Load `BibDatabase` object from a string

Parameters

- **bibtex_str** (*str or unicode*) – input BibTeX string to be parsed
- **parser** (*BibTexParser*) – custom parser to use (optional)

Returns bibliographic database object

Return type *BibDatabase*

bibtexparser.**dumps**(*bib_database*, *writer=None*)
Dump BibDatabase object to a BibTeX string

Parameters

- **bib_database** (BibDatabase) – bibliographic database object
- **writer** (*BibTexWriter*) – custom writer to use (optional) (not yet implemented)

Returns BibTeX string

Return type unicode

bibtexparser.**dump**(*bib_database*, *bibtex_file*, *writer=None*)
Save BibDatabase object as a BibTeX text file

Parameters

- **bib_database** (BibDatabase) – bibliographic database object
- **bibtex_file** (*file*) – file to write to
- **writer** (*BibTexWriter*) – custom writer to use (optional) (not yet implemented)

bibtexparser.bibdatabase — The bibliographic database object

class bibdatabase.**BibDatabase**
A bibliographic database object following the data structure of a BibTeX file.

comments = None
List of BibTeX comment (*@comment{...}*) blocks.

entries = None
List of BibTeX entries, for example *@book{...}*, *@article{...}*, etc. Each entry is a simple dict with BibTeX field-value pairs, for example *'author': 'Bird, R.B. and Armstrong, R.C. and Hassager, O.'* Each entry will always have the following dict keys (in addition to other BibTeX fields):

•*ID* (BibTeX key)

•*ENTRYTYPE* (entry type in lowercase, e.g. *book*, *article* etc.)

entries_dict
Return a dictionary of BibTeX entries. The dict key is the BibTeX entry key

preambles = None
List of BibTeX preamble (*@preamble{...}*) blocks.

strings = None
OrderedDict of BibTeX string definitions (*@string{...}*). In order of definition.

bibtexparser.bparser — Modifying the default parser

bibtexparser.customization — Record customization functions

bibtexparser.bwriter — Modifying the default writer

CHAPTER 5

Logging module to understand failures

Syntax of bibtex files is simple but there are many possible variations. This library probably fails for some of them.

Bibtexparser includes a large quantity of debug messages which helps to understand why and where the parser fails. The example below can be used to print these messages in the console.

```
import logging
import logging.config

logger = logging.getLogger(__name__)

logging.config.dictConfig({
    'version': 1,
    'disable_existing_loggers': False,
    'formatters': {
        'standard': {
            'format': '%(asctime)s [%(levelname)s] %(name)s %(funcName)s:%(lineno)d:
↪%(message)s'
        },
    },
    'handlers': {
        'default': {
            'level':'DEBUG',
            'formatter': 'standard',
            'class':'logging.StreamHandler',
        },
    },
    'loggers': {
        '': {
            'handlers': ['default'],
            'level': 'DEBUG',
            'formatter': 'standard',
            'propagate': True
        }
    }
})
```

```
if __name__ == '__main__':
    bibtex = """@ARTICLE{Cesar2013,
      author = {Jean César},
      title = {An amazing title},
      year = {2013},
      month = jan,
      volume = {12},
      pages = {12--23},
      journal = {Nice Journal},
      abstract = {This is an abstract. This line should be long enough to test
             multilines...},
      comments = {A comment},
      keywords = {keyword1, keyword2},
    }
    """

    with open('/tmp/bibtex.bib', 'w') as bibfile:
        bibfile.write(bibtex)

    from bibtexparser.bparser import BibTexParser

    with open('/tmp/bibtex.bib', 'r') as bibfile:
        bp = BibTexParser(bibfile.read())
        print(bp.get_entry_list())
```

I recommend you to use this output if you would like to report a bug.

CHAPTER 6

Bibtex tips and conventions

Format

http://maverick.inria.fr/~Xavier.Decoret/resources/xdkbibtex/bibtex_summary.html

- Comments
- Variable
- @preamble
- Name convention

Display your bibliography in html pages

Have a look to this project: http://www.monperrus.net/martin/bibtexbrowser/

Upper case letters in titles

Put the letter/word in curly braces like {this}.

How to handle accents and special characters?

TODO

IEEE citation reference

- https://origin.www.ieee.org/documents/ieeecitationref.pdf

Common Errors in Bibliographies John Owens

- http://www.ece.ucdavis.edu/~jowens/biberrors.html

Common abbreviations for journals

- Jabref list http://jabref.sourceforge.net/resources.php#downloadlists

CHAPTER 7

Who uses BibtexParser?

If your project uses BibtexParser, you can ask for the addition of a link in this list.

- http://timotheepoisot.fr/2013/11/10/shared-bibtex-file-markdown/
- https://github.com/Phyks/BMC
- http://aurelien.naldi.info/research/publications.html
- http://robot.kut.ac.kr/publications
- https://git.atelo.org/etlapale/bibgen
- https://onmenwhostareongraphs.wordpress.com/2015/06/09/graph-display-software-for-author-relationships-with-bibtex-files/
- https://github.com/vitorfs/parsifal

CHAPTER 8

Other projects

- http://pybtex.sourceforge.net/
- http://pybliographer.org/
- https://github.com/matthew-brett/babybib

CHAPTER 9

Indices and tables

- genindex
- modindex
- search

Python Module Index

b

Index