sjkabc Documentation

Release 1.4.0

Svante Kvarnström

June 21, 2016

Contents

sjkabc is a Python 3 library that provides ABC music notation parsing functionality.

CHAPTER 1

Thanks to

- Dr Bryan Duggan - for his thesis on 'Machine annotation of traditional Irish dance music' and for creating Tunepal
- Peter van Dijk - for help and advice on Python related issues

Contents:

1.1 Tutorial

sjkabc provides a set of functions that parses ABC music notation. There are currently three major types of functions:

- strip functions: These all begin with *strip_*. The strip functions return the passed string stripped from, for example, ornaments or accidentals. These functions can be used to create simplified notation suitable for searching.
- expand functions: These functions will expand repeats, multiple endings, and 'long notes'. These functions are also valuable for creating searchable ABC.
- parse functions: These are helper functions which use *Parser* to parse files and directories.

The two core classes in *sjkabc* are *Parser* and *Tune*, which are used to parse and describe pieces of ABC music.

1.1.1 Basic usage

Parsing a string of ABC

The *Parser* class is iterable, and will return *Tune* objects representing ABC tunes found in the input string. Here's an example of how to parse a tune and print its title:

```
from sjkabc import Parser

abc_string = """
X: 37
T: Apples In Winter
C: Trad.
R: Jig
M: 6/8
L: 1/8
K: Em
BEE BEE|Bdf edB|BAF FEF|DFA BAF|
BEE BEE|Bdf edB|BAB dAF|1FED EGA:|2FED EAc||
```

```
|:e2f gfe|eae edB|BAF FEF|DFA BAF|
e2f gfe|eae edB|BAB dAF|1FED EAc:|2FED E3|]
"""
for tune in Parser(abc_string):
    print tune.title[0]
```

Not all ABC header keys may be defined several times, but all Parser attributes are lists for the sake of consistency. Several titles may, for example, be defined simply by using several T: statements, but only one index number (X:) is permitted.

Parsing a file

The *parse_file()* helper function is used to parse files containing ABC notation. For example:

```
from sjkabc import parse_file

for tune in parse_file('test.abc'):
    print('Parsed {} with index number {}.'.format(tune.title[0], tune.index[0])
```

Parsing a directory of files

To parse all .abc files in a directory, one may use *parse_dir()*, which works in the same fashion as *parse_file()*.

```
from sjkabc import parse_dir

for tune in parse_dir('/data/music/abc/'):
    print('Parsed {} with index number {}.'.format(tune.title[0], tune.index[0])
```

1.2 sjkabc

sjkabc.sjkabc

This module provides functionality for parsing ABC music notation.

copyright

3. 2016 by Svante Kvarnström

license BSD, see LICENSE for more details.

sjkabc.**HEADER_KEYS**
Supported ABC notation header keys. This *dict* is used to populate the attributes of *Tune*.

sjkabc.**DECORATIONS = ['!trill!', '!trill(!', '!trill)!', '!lowermordent!', '!uppermordent!', '!mordent!', '!pralltriller!', '!roll!',**
List of decoration symbols according to the ABC notation standard v2.1.

class sjkabc.**Parser**(*abc=None*)
This class provides iterable parsing capabilities.

Parser must be initialised with a string containing ABC music notation. This class is iterable and will return a *Tune* object for every tune found in the provided ABC notation.

Example:

```
>>> for tune in Parser(abc):
...     print('Parsed ', tune.title)
```

See also:

Tune

parse(*abc*)
Parse ABC notation.

This function will append found ABC tunes to *self.tunes*.

Parameters **abc** – string containing abc to parse

class sjkabc.**Tune**(***kwargs*)
This class represents a parsed tune.

Its attributes are generated from *HEADER_KEYS*, with the addition of *abc* and *expanded_abc()*.

Example:

```
>>> t = Tune()
>>> t.title = 'Example tune'
>>> t.abc = '|:abc abc:|'
>>> t.expanded_abc
'abcabcabcabc'
```

See also:

HEADER_KEYS, *Parser*

abc = None
Tune body.

expanded_abc
Expanded ABC suitable for searching

Returns expanded abc

Return type str

format_abc()
Format ABC tune

This will return the current *Tune* as a properly formatted string, including header fields and ABC.

Returns ABC string suitable for writing to file

Return type str

sjkabc.**expand_abc**(*abc*)
Create searchable abc string

This runs all the stripping and expanding functions on the input string, and also makes it lowercase.

Parameters **abc** (*str*) – string of abc to expand

Returns string of expanded abc

Return type str

See also:

strip_octave(), *strip_accidentals()*, *strip_triplets()*, *strip_chords()* *strip_ornaments()*, *expand_notes()*, *expand_parts()*, *strip_whitespace()* *strip_bar_dividers()*, *strip_extra_chars()*, *strip_slurs()*

sjkabc.**expand_notes**(*abc*)
Expand notes, so that E2 becomes EE et.c.

Parameters **abc** (*str*) – abc to expand

Returns expanded abc

Return type str

sjkabc.**expand_parts**(*abc*)
Expand repeats with support for (two) alternate endings.

Example:

```
>>> print(expand_parts('aaa|bbb|1ccc:|2ddd|]'))
aaa|bbb|ccc|aaa|bbb|ddd|
```

Parameters **abc** (*str*) – abc to expand

Returns expanded abc

Return type str

sjkabc.**get_field_from_id**(*id*)
Get long field name from id char.

Parameters **id** (*str*) – id char, for example 'T'

Returns long field name, like 'title'

Return type str

Raises **KeyError** – if key does not exist.

sjkabc.**get_id_from_field**(*field*)
Get id char from field name

Parameters **field** (*str*) – 'long' name of field, for example 'title'

Returns id character, for example 'T'

Return type str

Raises **KeyError** – if key does not exist.

sjkabc.**parse_dir**(*dir*)
Run *Parser* on every file with .abc extension in *dir*

Parameters **dir** – Directory of abc files

Returns *Tune* object for every found file

Return type *Tune*

See also:

parse_file(), *Parser*, *Tune*

sjkabc.**parse_file**(*filename*)
Run Parser on file contents

This function is iterable.

Example

```
>>> for tune in parse_file('test.abc'):
...     print(tune.title)
```

Parameters **filename** – Name of file to parse

Returns *Tune* object for every found tune.

Return type *Tune*

See also:

parse_dir(), *Parser*, *Tune*

sjkabc.**strip_accidentals**(*abc*)

Remove accidentals from string.

Example:

```
>>> from sjkabc import strip_accidentals
>>> stripped = strip_whitespace('abc ^c=de|_e^fg _g=fe')
>>> stripped
'abc cde|efg gfe'
```

Parameters **abc** (*str*) – abc to filter

Returns abc with accidentals removed

Return type str

sjkabc.**strip_bar_dividers**(*abc*)

Strip bar dividers from string

This function can safely be run before expand_parts, as it won't remove repeats.

Example:

```
>>> from sjkabc import strip_bar_dividers
>>> stripped = strip_bar_dividers('abcd bcde|bcde abcd|defg abcd|bebe')
>>> stripped
'abcd bcdebcde abcddefg abcdbebe'
```

Parameters **abc** (*str*) – abc to filter

Returns abc without bar dividers

Return type str

sjkabc.**strip_chords**(*abc*)

Strip chords and 'guitar chords' from string.

Example:

```
>>> from sjkabc import strip_chords
>>> stripped = strip_chords('"G" abc|"Em" bcd|[GBd] cde')
>>> stripped
' abc| bcd | cde'
```

Parameters **abc** (*str*) – abc to filter

Returns abc with chords stripped

Return type str

sjkabc.**strip_decorations**(*abc*)
Remove decorations

Removes decorations defined in the v2.1 ABC notation standard.

Parameters **abc** (*str*) – ABC notation to process

Returns stripped ABC

Return type str

See also:

DECORATIONS

New in version 1.2.0.

sjkabc.**strip_extra_chars**(*abc*)
Strip misc extra chars (/<>)

Parameters **abc** (*str*) – abc to filter

Returns filtered abc

Return type str

sjkabc.**strip_gracenotes**(*abc*)
Remove gracenotes

Example:

```
>>> stripped = strip_gracenotes('abc bcd|c3 def|{/def}efg abc|')
>>> stripped
'abc bcd|c3 def|efg abc|'
```

Parameters **abc** (*str*) – abc to strip

Returns abc stripped from gracenotes

Return type str

sjkabc.**strip_octave**(*abc*)
Remove octave specifiers from string.

Example:

```
>>> from sjkabc import strip_octave
>>> stripped = strip_octave("A,B,C,d'e'f'")
>>> stripped
'ABCdef'
```

Parameters **abc** (*str*) – abc to filter

Returns abc with octave specifiers removed

Return type str

sjkabc.**strip_ornaments**(*abc*)
Remove gracenotes, tildes, trills, turns and fermatas from string.

Example:

```
>>> from sjkabc import strip_ornaments
>>> stripped = strip_ornaments('abc bcd|~c3 def|{/def}efg !trill(!ab|')
>>> stripped
'abc bcd|c3 def|efg ab|'
```

Parameters **abc** (*str*) – abc to filter

Returns filtered abc

Return type str

Deprecated since version 1.2.0: Use *strip_gracenotes()* and *strip_decorations()* instead.

sjkabc.**strip_slurs**(*abc*)
Remove slurs from string.

Example:

```
>>> strip_slurs('|:ab(cd) (a(bc)d):|')
|:abcd abcd:|
```

Warning: Don't use this before *strip_decorations()* as it may change certain decorations so that they wont be recognized. One example would be *!trill(!*.

Parameters **abc** (*str*) – abc to manipulate

Returns abc stripped from slurs

Return type str

sjkabc.**strip_triplets**(*abc*)
Remove duplets, triplets, quadruplets, etc from string.

Please note that this simply removes the (n and leaves the following notes.

Example:

```
>>> from sjkabc import strip_triplets
>>> stripped = strip_triplets('AB(3cBA Bcde|fd(3ddd (4efed (4BdBF')
>>> stripped
'ABcBA Bcde|fdddd efed BdBF'
```

Parameters **abc** (*str*) – abc to filter

Returns abc without triplets

Return type str

sjkabc.**strip_whitespace**(*abc*)
Remove whitespace and newlines from string.

Parameters **abc** (*str*) – abc to filter

Returns abc with whitespace removed

Return type str

sjkabc.**wrap_line**(*string*, *id*, *max_length=78*, *prefix='+'*)
Wrap header line.

Parameters

- **string** (*str*) – string to wrap

- **id** (*str*) – character id of header line
- **max_length** (*int*) – maximum line length
- **prefix** (*str*) – Line prefix for wrapped lines (first line exempted)

Returns wrapped line

Return type str

See also:

get_id_from_field()

1.3 Limitations

Please note that *sjkabc* does not currently fully conform to the ABC music standard as it does not implement all features and header keys. This is the future goal, however, and contributions are very welcome.

1.4 Contributing to the sjkabc project

1.4.1 Getting a copy of the source code

The first step to contributing to sjkabc is to get a copy of the source code. The easiest way of doing this is by using git. Browse the directory you want the source code repository in and run the following command:

```
$ git clone https://github.com/sjktje/sjkabc.git
```

1.4.2 Setting up a virtualenv

Now that you've got a copy of the source code you can install sjkabc. The most convenient way of doing this is by using a *virtual environment*. I prefer using virtualenvwrapper for managing my virtual environments, and you probably should too.

I refer you to the virtualenvwrapper documentation for setting up a virtual environment

---

Note: Please note that sjkabc will only run under python 3, so when creating your virtualenvironment take care to make sure it'll use python 3. One way of doing this would be using the –python option:

```
$ mkvirtualenv --python=python3 sjkabc
```

1.4.3 Installing the development version

Assuming you called your virtual environment *sjkabc*, the following commands will install the development version of sjkabc in your virtual environment.

```
$ cd path/to/sjkabc
$ workon sjkabc
$ git checkout develop
$ python setup.py develop
```

1.4.4 Contributing changes

The discussion and work on sjkabc is done through git, GitHub and more specifically GitHub's issue tracker. The GitHub help website provides information on how to get up and running easily and quickly with git.

Simply clone the sjkabc repository, create a branch off of the *develop* branch and hack away. When done, publish your branch and create a pull request.

Important: Proposed changes must include suitable tests and modifications to the documentation, if appropriate. If you need help with this, publish your branch and ask! We'll work on it together.

1.5 Change Log

1.5.1 1.4.0 (2016-06-21)

- format_abc() wraps header lines prefixing them with '+:'.

1.5.2 1.3.1 (2016-04-23)

- format_abc() will not include empty header lines.

1.5.3 1.3.0 (2016-03-27)

- Added support for continued info field lines (+:)
- Added factories for testing
- Unittest replaced with pytest
- Tune.format_abc() will no longer include empty info fields
- Add strip_slurs() which removes slurs (parentheses) from string. This function is called by expand_abc.

1.5.4 1.2.2 (2016-03-11)

- Fixed bug introduced by the Tune.abc and Tune.expanded_abc changes in last release This bug broke Tune.format_abc()

1.5.5 1.2.1 (2016-03-10)

- Fixed bug which caused Tune.expanded_abc never to be set

1.5.6 1.2.0 (2016-03-10)

- Added support for P: (parts)
- Added tune method format_abc which returns a properly formatted tune
- Added support for F: (file)

- Added support :: shorthand syntax (which equals :||:)
- Added strip_decorations()
- Added strip_gracenotes()
- Removed support for A: as it's deprecated according to the ABC standard
- Fixed Travis-CI setup
- The strip_ornaments is now deprecated. It has been replaced by strip_decorations and strip_gracenotes and will be removed in future versions.

1.5.7 1.1.0 (2016-03-08)

- The parser is now an iterable object (Parser) which yields Tune objects
- Added lots of documentation
- Documentation is now in ReST, generated with Sphinx.
- Some minor refactoring of code

S

A

D

E

F

G

H

P

S

T

W