sexpdata Documentation

Release 0.0.4.dev1

Takafumi Arakaki

Sep 27, 2017

Contents

sexpdata is a simple S-expression parser/serializer. It has simple *load* and *dump* functions like *pickle*, *json* or *PyYAML* module.

```
>>> from sexpdata import loads, dumps
>>> loads('("a" "b")')
['a', 'b']
>>> print(dumps(['a', 'b']))
("a" "b")
```

You can install *sexpdata* from PyPI:

```
pip install sexpdata
```

Links:

- Documentation (at Read the Docs)
- Repository (at GitHub)
- Issue tracker (at GitHub)
- PyPI
- Travis CI

CHAPTER 1

License

sexpdata is licensed under the terms of the BSD 2-Clause License. See the source code for more information.

`sexpdata.`**`load`**`(`*filelike*, ***kwds*`)`

Load object from S-expression stored in *filelike*.

Parameters **`filelike`** – A text stream object.

See `loads()` for valid keyword arguments.

```
>>> import io
>>> fp = io.StringIO()
>>> sexp = [Symbol('a'), Symbol('b')]   # let's dump and load this object
>>> dump(sexp, fp)
>>> _ = fp.seek(0)
>>> load(fp) == sexp
True
```

`sexpdata.`**`loads`**`(`*string*, ***kwds*`)`

Load object from S-expression *string*.

Parameters

- **`string`** – String containing an S-expression.
- **`nil`** (*`str or None`*) – A symbol interpreted as an empty list. Default is `'nil'`.
- **`true`** (*`str or None`*) – A symbol interpreted as True. Default is `'t'`.
- **`false`** (*`str or None`*) – A symbol interpreted as False. Default is `None`.
- **`line_comment`** (*`str`*) – Beginning of line comment. Default is `';'`.

```
>>> loads("(a b)")
[Symbol('a'), Symbol('b')]
>>> loads("a")
Symbol('a')
>>> loads("(a 'b)")
[Symbol('a'), Quoted(Symbol('b'))]
```

```
>>> loads("(a '(b))")
[Symbol('a'), Quoted([Symbol('b')])]
>>> loads('''
... ;; This is a line comment.
... ("a" "b")  ; this is also a comment.
... ''')
['a', 'b']
>>> loads('''
... # This is a line comment.
... ("a" "b")  # this is also a comment.
... ''', line_comment='#')
['a', 'b']
```

`nil` is converted to an empty list by default. You can use keyword argument *nil* to change what symbol must be interpreted as nil:

```
>>> loads("nil")
[]
>>> loads("null", nil='null')
[]
>>> loads("nil", nil=None)
Symbol('nil')
```

`t` is converted to True by default. You can use keyword argument *true* to change what symbol must be converted to True.:

```
>>> loads("t")
True
>>> loads("#t", true='#t')
True
>>> loads("t", true=None)
Symbol('t')
```

No symbol is converted to False by default. You can use keyword argument *false* to convert a symbol to False.

```
>>> loads("#f")
Symbol('#f')
>>> loads("#f", false='#f')
False
>>> loads("nil", false='nil', nil=None)
False
```

`sexpdata.`**`dump`**(*obj*, *filelike*, ***kwds*)

Write *obj* as an S-expression into given stream *filelike*.

Parameters

- **`obj`** – A Python object.
- **`filelike`** – A text stream object.

See *`dumps()`* for valid keyword arguments.

```
>>> import io
>>> fp = io.StringIO()
>>> dump([Symbol('a'), Symbol('b')], fp)
>>> print(fp.getvalue())
(a b)
```

sexpdata.**dumps**(*obj*, ***kwds*)

Convert python object into an S-expression.

Parameters

- **obj** – A Python object.
- **str_as** ('symbol' or 'string') – How string should be interpreted. Default is 'string'.
- **tuple_as** ('list' or 'array') – How tuple should be interpreted. Default is 'list'.
- **true_as** (*str*) – How True should be interpreted. Default is 't'
- **false_as** (*str*) – How False should be interpreted. Default is '()'
- **none_as** (*str*) – How None should be interpreted. Default is '()'

Basic usage:

```
>>> print(dumps(['a', 'b']))
("a" "b")
>>> print(dumps(['a', 'b'], str_as='symbol'))
(a b)
>>> print(dumps(dict(a=1)))
(:a 1)
>>> print(dumps([None, True, False, ()]))
(() t () ())
>>> print(dumps([None, True, False, ()],
...             none_as='null', true_as='#t', false_as='#f'))
(null #t #f ())
>>> print(dumps(('a', 'b')))
("a" "b")
>>> print(dumps(('a', 'b'), tuple_as='array'))
["a" "b"]
```

More verbose usage:

```
>>> print(dumps([Symbol('a'), Symbol('b')]))
(a b)
>>> print(dumps(Symbol('a')))
a
>>> print(dumps([Symbol('a'), Quoted(Symbol('b'))]))
(a 'b)
>>> print(dumps([Symbol('a'), Quoted([Symbol('b')])]))
(a '(b))
```

sexpdata.**car**(*obj*)

Alias of obj[0].

```
>>> car(loads('(a . b)'))
Symbol('a')
>>> car(loads('(a b)'))
Symbol('a')
```

sexpdata.**cdr**(*obj*)

cdr-like function.

```
>>> cdr(loads('(a . b)'))
Symbol('b')
```

```
>>> cdr(loads('(a b)'))
[Symbol('b')]
>>> cdr(loads('(a . (b))'))
[Symbol('b')]
>>> cdr(loads('(a)'))
[]
>>> cdr(loads('(a . nil)'))
[]
```

CHAPTER 2

Changelog

v0.0.3

- Encoded raw string can be dumped to S-expression, assuming that the encoding is UTF-8. See also tkf/emacs-jedi/#43.

v0.0.2

- Performance improvement. Especially for long string literal.

v0.0.1

- Initial release.

CHAPTER 3

Indices and tables

- genindex
- modindex
- search

Python Module Index

Index