PyJSON5 Documentation

Release 0.3.6

René Kijewski

Jun 02, 2018

Contents

A JSON5 serializer and parser library for Python 3 written in Cython.

The serializer returns ASCII data that can safely be used in an HTML template. Apostrophes, ampersands, greater-than, and less-then signs are encoded as unicode escaped sequences. E.g. this snippet is safe for any and all input:

```
"<a onclick='alert(" + encode(data) + ")'>show message</a>"
```

Unless the input contains infinite or NaN values, the result will be valid JSON data.

All valid JSON5 1.0.0 and JSON data can be read, unless the nesting level is absurdly high.

→ Glossary / Index

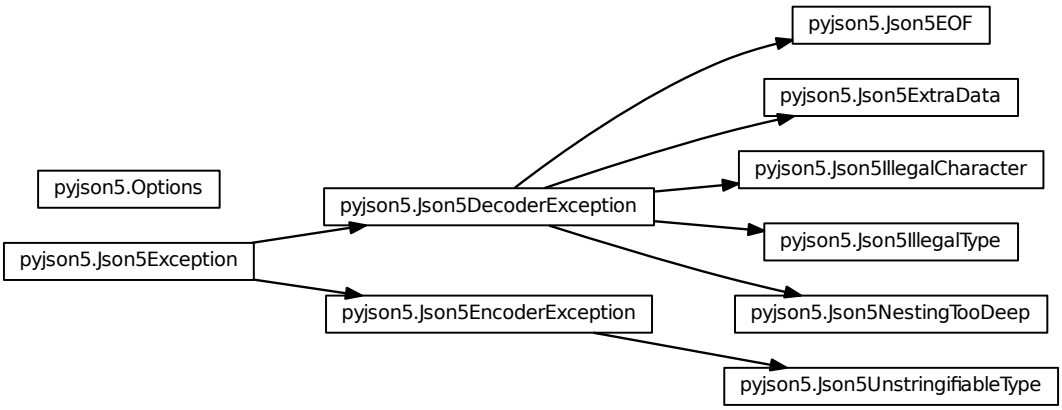
CHAPTER 1

Function summary

decode(data[, maxdepth, some])	Decodes JSON5 serialized data from an `str` object.
decode_buffer(obj[, maxdepth, some, wordlength])	Decodes JSON5 serialized data from an object that supports the buffer protocol, e.g.
decode_callback(cb[, maxdepth, some, args])	Decodes JSON5 serialized data by invoking a callback.
decode_io(fp[, maxdepth, some])	Decodes JSON5 serialized data from a file-like object.
decode_latin1(data[, maxdepth, some])	Decodes JSON5 serialized data from a `bytes` object.
dump(obj, fp, **kw)	Serializes a Python object to a JSON5 compatible unicode string.
dumps(obj, **kw)	Serializes a Python object to a JSON5 compatible unicode string.
encode(data, *[, options])	Serializes a Python object to a JSON5 compatible unicode string.
encode_bytes(data, *[, options])	Serializes a Python object to a JSON5 compatible bytes string.
encode_callback(data, cb[, supply_bytes, ...])	Serializes a Python object into a callback function.
encode_io(data, fp[, supply_bytes, options])	Serializes a Python object into a file-object.
encode_noop(data, *[, options])	Test if the input is serializable.
load(fp, **kw)	Decodes JSON5 serialized data from a file-like object.
loads(s, *[, encoding])	Decodes JSON5 serialized data from a string.

CHAPTER 2

Class / exception summary



CHAPTER 3

Full description

pyjson5.**decode**(*data*, *maxdepth=None*, *some=False*)

Decodes JSON5 serialized data from an `str` object.

```
decode('["Hello", "world!"]') == ['Hello', 'world!']
```

Parameters

- **data** (*unicode*) – JSON5 serialized data
- **maxdepth** (*Optional[int]*) – Maximum nesting level before are the parsing is aborted.
  - If `None` is supplied, then the value of the global variable `DEFAULT_MAX_NESTING_LEVEL` is used instead.
  - If the value is `0`, then only literals are accepted, e.g. `false`, `47.11`, or `"string"`.
  - If the value is negative, then the any nesting level is allowed until Python's recursion limit is hit.
- **some** (*bool*) – Allow trailing junk.

Raises

- *Json5DecoderException* – An exception occured while decoding.
- `TypeError` – An argument had a wrong type.

Returns Deserialized data.

Return type object

pyjson5.**decode_latin1**(*data*, *maxdepth=None*, *some=False*)

Decodes JSON5 serialized data from a `bytes` object.

```
decode_buffer(b'["Hello", "world!"]') == ['Hello', 'world!']
```

Parameters

- **data** (*bytes*) – JSON5 serialized data, encoded as Latin-1 or ASCII.
- **maxdepth** (*Optional[int]*) – see *decode(...)*
- **some** (*bool*) – see *decode(...)*

Raises

- *Json5DecoderException* – An exception occured while decoding.
- TypeError – An argument had a wrong type.

Returns see *decode(...)*

Return type object

pyjson5.**decode_buffer**(*obj*, *maxdepth=None*, *some=False*, *wordlength=None*)

Decodes JSON5 serialized data from an object that supports the buffer protocol, e.g. bytearray.

```
obj = memoryview(b'["Hello", "world!"]')

decode_buffer(obj) == ['Hello', 'world!']
```

Parameters

- **data** (*object*) – JSON5 serialized data. The argument must support Python's buffer protocol, i.e. `memoryview(...)` must work. The buffer must be contigious.
- **maxdepth** (*Optional[int]*) – see *decode(...)*
- **some** (*bool*) – see *decode(...)*
- **wordlength** (*Optional[int]*) – Must be 1, 2, 4 to denote UCS1, USC2 or USC4 data. Surrogates are not supported. Decode the data to an `str` if need be. If `None` is supplied, then the buffer's `itemsize` is used.

Raises

- *Json5DecoderException* – An exception occured while decoding.
- TypeError – An argument had a wrong type.
- ValueError – The value of `wordlength` was invalid.

Returns see *decode(...)*

Return type object

pyjson5.**decode_callback**(*cb*, *maxdepth=None*, *some=False*, *args=None*)

Decodes JSON5 serialized data by invoking a callback.

```
cb = iter('["Hello","world!"]').__next__

decode_callback(cb) == ['Hello', 'world!']
```

Parameters

- **cb** (*Callable[Any, Union[str|bytes|bytearray|int|None]]*) – A function to get values from. The functions is called like `cb(*args)`, and it returns:
  - **str, bytes, bytearray:** `len(...) == 0` denotes exhausted input. `len(...) == 1` is the next character.

  - **int:** < 0 denotes exhausted input. >= 0 is the ordinal value of the next character.
  - **None:** input exhausted
- **maxdepth** (*Optional[int]*) – see *decode(...)*
- **some** (*bool*) – see *decode(...)*
- **args** (*Optional[Iterable[Any]]*) – Arguments to call cb with.

Raises

- *Json5DecoderException* – An exception occured while decoding.
- TypeError – An argument had a wrong type.

Returns see decode(...)

Return type object

pyjson5.**decode_io**(*fp*, *maxdepth=None*, *some=True*)

Decodes JSON5 serialized data from a file-like object.

```
fp = io.StringIO("""
    ['Hello', /* TODO look into specs whom to greet */]
    'Wolrd' // FIXME: look for typos
""")

decode_io(fp) == ['Hello']
decode_io(fp) == 'Wolrd'

fp.seek(0)

decode_io(fp, some=False)
# raises Json5ExtraData('Extra data U+0027 near 56', ['Hello'], "'")
```

Parameters

- **fp** (*IOBase*) – A file-like object to parse from.
- **maxdepth** (*Optional[int] = None*) – see *decode(...)*
- **some** (*bool*) – see *decode(...)*

Raises

- *Json5DecoderException* – An exception occured while decoding.
- TypeError – An argument had a wrong type.

Returns see decode(...)

Return type object

pyjson5.**encode**(*data*, *, *options=None*, ***options_kw*)

Serializes a Python object to a JSON5 compatible unicode string.

```
encode(['Hello', 'world!']) == '["Hello","world!"]'
```

Parameters

- **data** (*object*) – Python object to serialize.

- **options** (*Optional[Options]*) – Extra options for the encoder. If `options` **and** `options_kw` are specified, then `options.update(**options_kw)` is used.
- **options_kw** – See Option's arguments.

Raises

- *Json5EncoderException* – An exception occured while encoding.
- `TypeError` – An argument had a wrong type.

Returns

Unless `float('inf')` or `float('nan')` is encountered, the result will be valid JSON data (as of RFC8259).

The result is always ASCII. All characters outside of the ASCII range are encoded.

The result safe to use in an HTML template, e.g. `<a onclick='alert({{ encode(url) }})'>show message</a>`. Apostrophes `"'"` are encoded as `"\u0027"`, less-than, greater-than, and ampersand likewise.

Return type str

pyjson5.**encode_bytes**(*data*, *, *options=None*, ***options_kw*)

Serializes a Python object to a JSON5 compatible bytes string.

```
encode_bytes(['Hello', 'world!']) == b'["Hello","world!"]'
```

Parameters

- **data** (*object*) – see *encode(...)*
- **options** (*Optional[Options]*) – see *encode(...)*
- **options_kw** – see *encode(...)*

Raises

- *Json5EncoderException* – An exception occured while encoding.
- `TypeError` – An argument had a wrong type.

Returns see *encode(...)*

Return type bytes

pyjson5.**encode_callback**(*data*, *cb*, *supply_bytes=False*, *, *options=None*, ***options_kw*)

Serializes a Python object into a callback function.

The callback function `cb` gets called with single characters and strings until the input `data` is fully serialized.

```
encode_callback(['Hello', 'world!'], print)
#prints:
# [
# "
# Hello
# "
# ,
# "
# world!
# "
" ]
```

Parameters

- **data** (*object*) – see *encode(...)*
- **cb** (*Callable[[Union[bytes|str]], None]*) – A callback function. Depending on the truthyness of `supply_bytes` either `bytes` or `str` is supplied.
- **supply_bytes** (*bool*) – Call `cb(...)` with a `bytes` argument if true, otherwise `str`.
- **options** (*Optional[Options]*) – see *encode(...)*
- **options_kw** – see *encode(...)*

Raises

- *Json5EncoderException* – An exception occured while encoding.
- `TypeError` – An argument had a wrong type.

Returns The supplied argument `cb`.

Return type Callable[[Union[bytes|str]], None]

pyjson5.**encode_io**(*data*, *fp*, *supply_bytes=True*, ***, *options=None*, ***options_kw*)

Serializes a Python object into a file-object.

The return value of `fp.write(...)` is not checked. If `fp` is unbuffered, then the result will be garbage!

Parameters

- **data** (*object*) – see *encode(...)*
- **fp** (*IOBase*) – A file-like object to serialize into.
- **supply_bytes** (*bool*) – Call `fp.write(...)` with a `bytes` argument if true, otherwise `str`.
- **options** (*Optional[Options]*) – see *encode(...)*
- **options_kw** – see *encode(...)*

Raises

- *Json5EncoderException* – An exception occured while encoding.
- `TypeError` – An argument had a wrong type.

Returns The supplied argument `fp`.

Return type IOBase

pyjson5.**encode_noop**(*data*, ***, *options=None*, ***options_kw*)

Test if the input is serializable.

Most likely you want to serialize `data` directly, and catch exceptions instead of using this function!

```
encode_noop({47: 11}) == True
encode_noop({47: object()}) == False
```

Parameters

- **data** (*object*) – see *encode(...)*
- **options** (*Optional[Options]*) – see *encode(...)*
- **options_kw** – see *encode(...)*

Returns `True` iff `data` is serializable.

Return type bool

class pyjson5.**Options**

Customizations for the `encoder_*(...)` function family.

Immutable. Use `Options.update(**kw)` to create a **new** Options instance.

Parameters

- **tojson** (*str|False|None*) –
  - **str:** A special method to call on objects to return a custom JSON encoded string. Must return ASCII data!
  - **False:** No such member exists. (Default.)
  - **None:** Use default.
- **posinfinity** (*str|False|None*) –
  - **str:** String to represent positive infinity. Must be ASCII.
  - **False:** Throw an exception if `float('+inf')` is encountered.
  - **None:** Use default: `"Infinity"`.
- **neginfinity** (*str|False|None*) –
  - **str:** String to represent negative infinity. Must be ASCII.
  - **False:** Throw an exception if `float('-inf')` is encountered.
  - **None:** Use default: `"-Infinity"`.
- **nan** (*str|False|None*) –
  - **str:** String to represent not-a-number. Must be ASCII.
  - **False:** Throw an exception if `float('NaN')` is encountered.
  - **None:** Use default: `"NaN"`.
- **intformat** (*str|False|None*) –
  - **str:** Format string to use with `int`.
  - **False:** Throw an exception if an `int` is encountered.
  - **None:** Use default: `"%d"`.
- **floatformat** (*str|False|None*) –
  - **str:** Format string to use with `float`.
  - **False:** Throw an exception if a `float` is encountered.
  - **None:** Use default: `"%.6e"`.
- **decimalformat** (*str|False|None*) –
  - **str:** Format string to use with `Decimal`.
  - **False:** Throw an exception if a `Decimal` is encountered.
  - **None:** Use default: `"%s"`.
- **mappingtypes** (*Iterable[type]|False|None*) –
  - **Iterable[type]:** Classes the should be encoded to objects. Must be iterable over their keys, and implement `__getitem__`.

- **False:** There are no objects. Any object will be encoded as list of key-value tuples.
- **None:** Use default: `[collections.abc.Mapping]`.

decimalformat
The creation argument `decimalformat`. `None` if `False` was specified.

floatformat
The creation argument `floatformat`. `None` if `False` was specified.

intformat
The creation argument `intformat`. `None` if `False` was specified.

mappingtypes
The creation argument `mappingtypes`. `()` if `False` was specified.

nan
The creation argument `nan`. `None` if `False` was specified.

neginfinity
The creation argument `neginfinity`. `None` if `False` was specified.

posinfinity
The creation argument `posinfinity`. `None` if `False` was specified.

tojson
The creation argument `tojson`. `None` if `False` was specified.

update (*self*, ***kw*)
Creates a new Options instance by modifying some members.

pyjson5.**loads** (*s*, ***, *encoding='UTF-8'*, ***kw*)
Decodes JSON5 serialized data from a string.

Use *decode(...)* instead!

```
loads(s) == decode(s)
```

Parameters

- **s** (*object*) – Unless the argument is an `str`, it gets decoded according to the parameter `encoding`.
- **encoding** (*str*) – Codec to use if `s` is not an `str`.
- **kw** – Silently ignored.

Returns see `decode(...)`

Return type object

pyjson5.**load** (*fp*, ***kw*)
Decodes JSON5 serialized data from a file-like object.

Use *decode_io(...)* instead!

```
load(fp) == decode_io(fp, None, False)
```

Parameters

- **fp** (*IOBase*) – A file-like object to parse from.
- **kw** – Silently ignored.

Returns see `decode(...)`

Return type object

`pyjson5.`**`dumps`**(*obj*, ***kw*)

Serializes a Python object to a JSON5 compatible unicode string.

Use *encode(...)* instead!

```
dumps(obj) == encode(obj)
```

Parameters

- **`obj`** (*object*) – Python object to serialize.
- **`kw`** – Silently ignored.

Returns see `encode(data)`

Return type unicode

`pyjson5.`**`dump`**(*obj*, *fp*, ***kw*)

Serializes a Python object to a JSON5 compatible unicode string.

Use *encode_io(...)* instead!

```
dump(obj, fp) == encode_io(obj, fp)
```

Parameters

- **`obj`** (*object*) – Python object to serialize.
- **`fp`** (*IOBase*) – A file-like object to serialize into.
- **`kw`** – Silently ignored.

exception `pyjson5.`**`Json5Exception`**(*message=None*, **args*)

Base class of any exception thrown by PyJSON5.

`message`

Human readable error description

`with_traceback`()

Exception.with_traceback(tb) – set self.__traceback__ to tb and return self.

exception `pyjson5.`**`Json5EncoderException`**

Base class of any exception thrown by the serializer.

`message`

Human readable error description

`with_traceback`()

Exception.with_traceback(tb) – set self.__traceback__ to tb and return self.

exception `pyjson5.`**`Json5UnstringifiableType`**(*message=None*, *unstringifiable=None*)

The encoder was not able to stringify the input, or it was told not to by the supplied `Options`.

`message`

Human readable error description

`unstringifiable`

The value that caused the problem.

with_traceback()
Exception.with_traceback(tb) – set self.__traceback__ to tb and return self.

exception pyjson5.**Json5DecoderException**(*message=None*, *result=None*, **args*)
Base class of any exception thrown by the parser.

message
Human readable error description

result
Deserialized data up until now.

with_traceback()
Exception.with_traceback(tb) – set self.__traceback__ to tb and return self.

exception pyjson5.**Json5NestingTooDeep**
The maximum nesting level on the input data was exceeded.

message
Human readable error description

result
Deserialized data up until now.

with_traceback()
Exception.with_traceback(tb) – set self.__traceback__ to tb and return self.

exception pyjson5.**Json5EOF**
The input ended prematurely.

message
Human readable error description

result
Deserialized data up until now.

with_traceback()
Exception.with_traceback(tb) – set self.__traceback__ to tb and return self.

exception pyjson5.**Json5IllegalCharacter**(*message=None*, *result=None*, *character=None*, **args*)
An unexpected character was encountered.

character
Extranous character.

message
Human readable error description

result
Deserialized data up until now.

with_traceback()
Exception.with_traceback(tb) – set self.__traceback__ to tb and return self.

exception pyjson5.**Json5ExtraData**(*message=None*, *result=None*, *character=None*, **args*)
The input contained extranous data.

character
Extranous character.

message
Human readable error description

result
Deserialized data up until now.

with_traceback()
Exception.with_traceback(tb) – set self.__traceback__ to tb and return self.

exception pyjson5.**Json5IllegalType**(*message=None*, *result=None*, *value=None*, **args*)
The user supplied callback function returned illegal data.

message
Human readable error description

result
Deserialized data up until now.

value
Value that caused the problem.

with_traceback()
Exception.with_traceback(tb) – set self.__traceback__ to tb and return self.

Python Module Index

p

Index

U

V

W