
jsonseq Documentation

Release 1.0a1

Sean Gillies

Jul 31, 2019

Contents:

1	Introduction	3
2	API Documentation	5
2.1	jsonseq package	5
2.2	jsonseq.decode module	5
2.3	jsonseq.encode module	6
3	Project Home	7
4	See Also	9
5	Indices and tables	11
	Python Module Index	13
	Index	15

RFC 7464 JSON Text Sequences encoding and decoding for Python.

CHAPTER 1

Introduction

The `JSONSeqEncoder` class takes streams of JSON-serializable Python objects and yields for each object its JSON representation sandwiched between an optional ASCII record separator (RS, `\x1e`) and a newline (`\n`).

```
>>> from jsonseq.encode import JSONSeqEncoder
>>> for chunk in JSONSeqEncoder().encode(({ "a": i, "b": i } for i in range(3))):
...     print(repr(chunk))
...
'{"a": 0, "b": 0}\n'
'{"a": 1, "b": 1}\n'
'{"a": 2, "b": 2}\n'
```

The RS allows pretty-printed JSON to be streamed out in sequences that can be decoded again.

```
>>> for chunk in JSONSeqEncoder(with_rs=True, indent=2).encode(({ "a": i, "b": i } for
↳ i in range(3))):
...     print(repr(chunk))
...
'\x1e{\n  "a": 0,\n  "b": 0\n}\n'
'\x1e{\n  "a": 1,\n  "b": 1\n}\n'
'\x1e{\n  "a": 2,\n  "b": 2\n}\n'
```

You can also get small chunks of the JSON sequences as they are encoded with `JSONSeqEncoder`'s `iterencode` method.

```
>>> for chunk in JSONSeqEncoder(with_rs=True).iterencode(({ "a": i } for i in
↳ range(3))):
...     print(repr(chunk))
...
'\x1e'
'{'
'"a"'
':'
'0'
'}'
'\n'
```

(continues on next page)

(continued from previous page)

```
'\x1e'
'{ '
'"a"'
': '
'1'
'}'
'\n'
'\x1e'
'{ '
'"a"'
': '
'2'
'}'
'\n'
```

You can use either `encode` or `iterencode` to copy JSON text sequences to a file.

```
with open("/tmp/example.jsons", "w") as f:
    for chunk in JSONSeqEncoder(with_rs=True, indent=2).iterencode(({ "a": i, "b": i }
↪ for i in range(3))):
        f.write(chunk)
```

There is no need to add a newline when calling the file's `write` method. `JSONSeqEncoder` ensures that it's already there where it needs to be.

The `JSONSeqDecoder` class takes streams of JSON texts sandwiched between the optional RS and a newline and yields decoded Python objects.

```
>>> stream = ['\x1e', '{', '"a"', ': ', '0', '}', '\n', '\x1e', '{', '"a"', ': ', '1',
↪ '}', '\n', '\x1e', '{', '"a"', ': ', '2', '}', '\n']
>>> for obj in JSONSeqDecoder().decode(stream):
...     print(repr(obj))
...
{'a': 0}
{'a': 1}
{'a': 2}
```

Objects can be read from a file in the same way.

```
>>> with open("/tmp/example.jsons") as f:
...     for obj in JSONSeqDecoder().decode(f):
...         print(repr(obj))
...
{'a': 0, 'b': 0}
{'a': 1, 'b': 1}
{'a': 2, 'b': 2}
```


2.1 jsonseq package

The jsonseq package.

2.2 jsonseq.decode module

RFC 7464 GeoJSON Text Sequence decoding.

class jsonseq.decode.JSONSeqDecoder (**kws)
Bases: object
Decode Python objects from a stream of JSON texts.

Methods

<i>decode</i> (self, seq)	Iterate over decoded objects in the JSON text sequence.
---------------------------	---

decode (*self*, *seq*: *Iterable*) → Iterator[object]
Iterate over decoded objects in the JSON text sequence.

Parameters

seq [Iterable] JSON strings or pieces of strings.

Yields

object

2.3 jsonseq.encode module

RFC 7464 GeoJSON Text Sequence encoding.

class jsonseq.encode.JSONSeqEncoder (*with_rs: bool = True, **kws*)

Bases: object

Encodes sequences of Python objects.

Methods

<i>encode</i> (self, iterable)	Yield JSON representations of objects.
<i>iterencode</i> (self, iterable)	Yield parts of JSON representations as available.

encode (*self, iterable: Iterable*) → Iterator[str]

Yield JSON representations of objects.

Parameters

iterable [Iterable] An iterable object, the source of Python objects to encode.

Yields

str

iterencode (*self, iterable: Iterable*) → Iterator[str]

Yield parts of JSON representations as available.

Parameters

iterable [Iterable] An iterable object, the source of Python objects to encode.

Yields

str

CHAPTER 3

Project Home

<https://github.com/sgillies/jsonseq>

CHAPTER 4

See Also

- [RFC 7464 JSON Text Sequences](#)

CHAPTER 5

Indices and tables

- `genindex`
- `modindex`
- `search`

j

- [jsonseq, 5](#)
- [jsonseq.decode, 5](#)
- [jsonseq.encode, 6](#)

D

`decode()` (*jsonseq.decode.JSONSeqDecoder method*), [5](#)

E

`encode()` (*jsonseq.encode.JSONSeqEncoder method*), [6](#)

I

`iterencode()` (*jsonseq.encode.JSONSeqEncoder method*), [6](#)

J

`jsonseq` (*module*), [5](#)

`jsonseq.decode` (*module*), [5](#)

`jsonseq.encode` (*module*), [6](#)

`JSONSeqDecoder` (*class in jsonseq.decode*), [5](#)

`JSONSeqEncoder` (*class in jsonseq.encode*), [6](#)