
srt Documentation

Release 3.5.3

Chris Down

Mar 28, 2023

CONTENTS

1	Documentation	3
1.1	Quickstart	3
1.2	API documentation	4
2	Indices and tables	9
	Python Module Index	11
	Index	13

srt is a tiny Python library for parsing, modifying, and composing SRT files.

1.1 Quickstart

1.1.1 Parse an SRT to Python objects

```
>>> import srt
>>> subtitle_generator = srt.parse('''\
... 1
... 00:31:37,894 --> 00:31:39,928
... OK, look, I think I have a plan here.
...
... 2
... 00:31:39,931 --> 00:31:41,931
... Using mainly spoons,
...
... 3
... 00:31:41,933 --> 00:31:43,435
... we dig a tunnel under the city and release it into the wild.
...
... ''')
>>> subtitles = list(subtitle_generator)
>>>
>>> subtitles[0].start
datetime.timedelta(0, 1897, 894000)
>>> subtitles[1].content
'Using mainly spoons,'
```

1.1.2 Compose an SRT from Python objects

```
>>> print(srt.compose(subtitles))
1
00:31:37,894 --> 00:31:39,928
OK, look, I think I have a plan here.

2
00:31:39,931 --> 00:31:41,931
Using mainly spoons,

3
00:31:41,933 --> 00:31:43,435
```

(continues on next page)

(continued from previous page)

```
we dig a tunnel under the city and release it into the wild.
```

1.2 API documentation

A tiny library for parsing, modifying, and composing SRT files.

exception `srt.SRTParseError` (*expected_start, actual_start, unmatched_content*)

Raised when part of an SRT block could not be parsed.

Parameters

- **expected_start** (*int*) – The expected contiguous start index
- **actual_start** (*int*) – The actual non-contiguous start index
- **unmatched_content** (*str*) – The content between the expected start index and the actual start index

class `srt.Subtitle` (*index, start, end, content, proprietary=""*)

The metadata relating to a single subtitle. Subtitles are sorted by start time by default. If no index was provided, index 0 will be used on writing an SRT block.

Parameters

- **index** (*int or None*) – The SRT index for this subtitle
- **start** (*datetime.timedelta*) – The time that the subtitle should start being shown
- **end** (*datetime.timedelta*) – The time that the subtitle should stop being shown
- **proprietary** (*str*) – Proprietary metadata for this subtitle
- **content** (*str*) – The subtitle content. Should not contain OS-specific line separators, only `\n`. This is taken care of already if you use `srt.parse()` to generate Subtitle objects.

to_srt (*strict=True, eol='\n'*)

Convert the current `Subtitle` to an SRT block.

Parameters

- **strict** (*bool*) – If disabled, will allow blank lines in the content of the SRT block, which is a violation of the SRT standard and may cause your media player to explode
- **eol** (*str*) – The end of line string to use (default “`\n`”)

Returns The metadata of the current `Subtitle` object as an SRT formatted subtitle block

Return type `str`

exception `srt.TimestampParseError`

Raised when an SRT timestamp could not be parsed.

`srt.compose` (*subtitles, reindex=True, start_index=1, strict=True, eol=None, in_place=False*)

Convert an iterator of `Subtitle` objects to a string of joined SRT blocks.

```
>>> from datetime import timedelta
>>> start = timedelta(seconds=1)
>>> end = timedelta(seconds=2)
>>> subs = [
```

(continues on next page)

(continued from previous page)

```

...     Subtitle(index=1, start=start, end=end, content='x'),
...     Subtitle(index=2, start=start, end=end, content='y'),
... ]
>>> compose(subs)
'1\n00:00:01,000 --> 00:00:02,000\nx\n\n2\n00:00:01,000 --> ...'
```

Parameters

- **subtitles** (iterator of *Subtitle* objects) – The subtitles to convert to SRT blocks
- **reindex** (*bool*) – Whether to reindex subtitles based on start time
- **start_index** (*int*) – If reindexing, the index to start reindexing from
- **strict** (*bool*) – Whether to enable strict mode, see *Subtitle.to_srt()* for more information
- **eol** (*str*) – The end of line string to use (default “\n”)
- **in_place** (*bool*) – Whether to reindex subs in-place for performance (version <=1.0.0 behaviour)

Returns A single SRT formatted string, with each input *Subtitle* represented as an SRT block

Return type *str*

`srt.make_legal_content` (*content*)

Remove illegal content from a content block. Illegal content includes:

- Blank lines
- Starting or ending with a blank line

```

>>> make_legal_content('\nfoo\n\nbar\n')
'foo\nbar'
```

Parameters **content** (*str*) – The content to make legal

Returns The legalised content

Return type *srt*

`srt.parse` (*srt*, *ignore_errors=False*)

Convert an SRT formatted string (in Python 2, a *unicode* object) to a *generator* of *Subtitle* objects.

This function works around bugs present in many SRT files, most notably that it is designed to not bork when presented with a blank line as part of a subtitle’s content.

```

>>> subs = parse("""\
... 422
... 00:31:39,931 --> 00:31:41,931
... Using mainly spoons,
...
... 423
... 00:31:41,933 --> 00:31:43,435
... we dig a tunnel under the city and release it into the wild.
...
... """)
>>> list(subs)
[Subtitle(...index=422...), Subtitle(...index=423...)]
```

Parameters

- **srt** (*str* or a file-like object) – Subtitles in SRT format
- **ignore_errors** – If True, garbled SRT data will be ignored, and we'll continue trying to parse the rest of the file, instead of raising *SRTParseError* and stopping execution.

Returns The subtitles contained in the SRT file as *Subtitle* objects

Return type generator of *Subtitle* objects

Raises *SRTParseError* – If the matches are not contiguous and `ignore_errors` is False.

`srt.sort_and_reindex` (*subtitles*, *start_index=1*, *in_place=False*, *skip=True*)

Reorder subtitles to be sorted by start time order, and rewrite the indexes to be in that same order. This ensures that the SRT file will play in an expected fashion after, for example, times were changed in some subtitles and they may need to be resorted.

If `skip=True`, subtitles will also be skipped if they are considered not to be useful. Currently, the conditions to be considered “not useful” are as follows:

- Content is empty, or only whitespace
- The start time is negative
- The start time is equal to or later than the end time

```
>>> from datetime import timedelta
>>> one = timedelta(seconds=1)
>>> two = timedelta(seconds=2)
>>> three = timedelta(seconds=3)
>>> subs = [
...     Subtitle(index=999, start=one, end=two, content='1'),
...     Subtitle(index=0, start=two, end=three, content='2'),
... ]
>>> list(sort_and_reindex(subs))
[Subtitle(...index=1...), Subtitle(...index=2...)]
```

Parameters

- **subtitles** – *Subtitle* objects in any order
- **start_index** (*int*) – The index to start from
- **in_place** (*bool*) – Whether to modify subs in-place for performance (version <=1.0.0 behaviour)
- **skip** (*bool*) – Whether to skip subtitles considered not useful (see above for rules)

Returns The sorted subtitles

Return type generator of *Subtitle* objects

`srt.srt_timestamp_to_timedelta` (*timestamp*)

Convert an SRT timestamp to a *timedelta*.

```
>>> srt_timestamp_to_timedelta('01:23:04,000')
datetime.timedelta(seconds=4984)
```

Parameters **timestamp** (*str*) – A timestamp in SRT format

Returns The timestamp as a *timedelta*

Return type `datetime.timedelta`

Raises `TimestampParseError` – If the timestamp is not parseable

`srt.timedelta_to_srt_timestamp` (*timedelta_timestamp*)

Convert a `timedelta` to an SRT timestamp.

```
>>> import datetime
>>> delta = datetime.timedelta(hours=1, minutes=23, seconds=4)
>>> timedelta_to_srt_timestamp(delta)
'01:23:04,000'
```

Parameters `timedelta_timestamp` (*datetime.timedelta*) – A datetime to convert to an SRT timestamp

Returns The timestamp in SRT format

Return type `str`

INDICES AND TABLES

- genindex
- search

PYTHON MODULE INDEX

S

`srt`, 4

INDEX

C

`compose()` (*in module srt*), 4

M

`make_legal_content()` (*in module srt*), 5

module

srt, 4

P

`parse()` (*in module srt*), 5

S

`sort_and_reindex()` (*in module srt*), 6

srt

 module, 4

`srt_timestamp_to_timedelta()` (*in module srt*), 6

`SRTParseError`, 4

`Subtitle` (*class in srt*), 4

T

`timedelta_to_srt_timestamp()` (*in module srt*), 7

`TimestampParseError`, 4

`to_srt()` (*srt.Subtitle method*), 4