srt3

Release 1.0.0

unknown

Jul 12, 2021

CONTENTS

srt3 is a simple Python library for parsing, modifying, and composing SRT files.

CHAPTER

ONE

DOCUMENTATION

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,

```

(continues on next page)

(continued from previous page)

```
3
00:31:41,933 --> 00:31:43,435
we dig a tunnel under the city and release it into the wild.
```

1.1.3 Import Guide

```
### Use srt via srt.func()
# import the whole srt package (including tools)
import srt

# only imports the srt.py module
from srt import srt

### Use srt tools
import srt
# srt.tools.tool.func()
srt.tools.find.find_by_timestamp()

from srt import tools
# tools.tool.func()
tools.find.find_by_timestamp()

# import all members from a tool module.
from srt.tools.find import *
find_by_timestamp()
```

1.2 API Documentation

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

exception 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 Subtitle(*index*, *start*, *end*, *content*, *proprietary=''*)

The metadata relating to a single subtitle. Subtitles are sorted by start time by default.

Parameters

- **index** (*int*) – 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 case 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 TimestampParseError
Raised when an SRT timestamp could not be parsed.

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 = [
...     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

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

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

Convert an SRT formatted string 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.

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'),
```

(continues on next page)

(continued from previous page)

```
... ]
>>> 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) https://en.wikipedia.org/wiki/in-place_algorithm
- **skip** (*bool*) – Whether to skip subtitles considered not useful (see above for rules)

Returns The sorted subtitles

Return type generator of *Subtitle* objects

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

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

1.3 Tools Documentation

srt3 tools perform tasks using the srt module.

Modules

srt.tools.add	Add a subtitle to subtitles.
srt.tools.deduplicate	Merge multiple subtitles together into one.
srt.tools.find	Find subtitles by timestamp.
srt.tools.fixed_timeshift	Shifts subtitles by a fixed number of seconds.
srt.tools.linear_timeshift	Perform linear time correction on a subtitle.
srt.tools.match	Filter and/or process subtitles' content that match a particular pattern.
srt.tools.mux	Merge multiple subtitles with similar start/end times into one.
srt.tools.normalize	Take a badly formatted SRT file and output a strictly valid one.
srt.tools.paste	Paste subtitles into other subtitles at a given timestamp.

1.3.1 srt.tools.add

Add a subtitle to subtitles.

add(*subs*, *start*, *end*, *content=''*, *adjust=False*)
Adds a subtitle to subtitles in the correct position.

Parameters

- **subs** – Subtitle objects
- **start** (*datetime.timedelta*) – The timestamp the subtitle starts at.
- **end** (*datetime.timedelta*) – The timestamp the subtitle ends at.
- **adjust** (*boolean*) – Whether to adjust the timestamps of subsequent subtitles.

Return type generator of Subtitle objects

1.3.2 srt.tools.deduplicate

Merge multiple subtitles together into one.

deduplicate(*orig_subs*, *acceptable_diff*)
Removes subtitles with duplicated content.

Parameters

- **orig_subs** – Subtitle objects
- **acceptable_diff** (*datetime.timedelta*) – The amount of milliseconds a subtitle start time must be to shift.

Return type generator of Subtitle objects

1.3.3 srt.tools.find

Find subtitles by timestamp.

find_by_timestamp(*subs*, *timestamp_one=datetime.timedelta(0)*, *timestamp_two=datetime.timedelta(0)*, *adjust=False*)

Finds subtitles from subtitles by timestamp. When timestamp one > timestamp two, subtitles up to timestamp two and subtitles after timestamp one will be found.

Parameters

- **subs** – Subtitle objects
- **timestamp_one** (*datetime.timedelta*) – The timestamp to find from.
- **timestamp_two** (*datetime.timedelta*) – The timestamp to find to.
- **adjust** (*boolean*) – Whether to adjust the timestamps of found subtitles.

Return type generator of Subtitle objects

1.3.4 srt.tools.fixed_timeshift

Shifts subtitles by a fixed number of seconds.

timeshift(*subtitles*, *seconds_to_shift*)

Performs a fixed timeshift on given subtitles.

Parameters

- **subtitles** – Subtitle objects
- **seconds_to_shift** (*float*) – The amount of seconds to shift.

Return type generator of Subtitle objects

1.3.5 srt.tools.linear_timeshift

Perform linear time correction on a subtitle.

timeshift(*subtitles*, *angular*, *linear*)

Performs a linear timeshift on given subtitles.

Parameters

- **subtitles** – Subtitle objects
- **angular** (*float*) –
- **linear** (*float*) –

Return type generator of Subtitle objects

1.3.6 srt.tools.match

Filter and/or process subtitles' content that match a particular pattern.

match(*subtitles*, *imports*, *func_match*, *func_process*, *lines*)
Passes each matching subtitle-content to a function.

Parameters

- **subtitles** – Subtitle objects
- **imports** – Modules to import in the context of the function.
- **func_match** (*str*) – The function used to match lines.
- **func_process** (*str*) – The function used to process subtitle content.
- **invert** (*bool*) – Whether to only match lines that return False.
- **per_line** – Whether to apply functions to each line of content (as opposed to the whole content string).

Return type generator of Subtitle objects

1.3.7 srt.tools.mux

Merge multiple subtitles with similar start/end times into one.

mux(*subs*, *acceptable_diff*, *attr*, *width*)
Merges subs with similar start/end times together (in-place). This prevents subtitles from jumping around the screen.

Parameters

- **subs** – Subtitle objects
- **acceptable_diff** (*datetime.timedelta*) – The amount of milliseconds a subtitle start time must be to shift.
- **attr** (*str*) –
- **width** (*int*) – The amount of subtitles to consider for time matching at once.

Return type generator of Subtitle objects

1.3.8 srt.tools.normalize

Take a badly formatted SRT file and output a strictly valid one.

normalize(*subs*, *strict*)
Normalises subtitles.

Parameters

- **subs** – Subtitle objects
- **strict** (*bool*) – Whether to enable strict mode, see Subtitle.to_srt() for more information

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

Return type str

Raises *SRTParseError* – If parsing fails.

1.3.9 srt.tools.paste

Paste subtitles into other subtitles at a given timestamp.

paste(*subs*, *copy*, *timestamp*, *space=datetime.timedelta(0)*, *block=False*)
Pastes subtitles into other subtitles at a given timestamp.

Parameters

- **subs** – `Subtitle` objects
- **copy** – The `Subtitle` objects to be pasted.
- **timestamp** (*datetime.timedelta*) – The timestamp to paste at.
- **space** (*datetime.timedelta*) – The amount of space to precede the paste.
- **block** (*boolean*) – Whether to paste the copied subtitles as a block and adjust the timestamps of subsequent subtitles.

Return type generator of `Subtitle` objects

CHAPTER

TWO

INDICES AND TABLES

- genindex
- search

PYTHON MODULE INDEX

S

INDEX