Timecode

Release 0.1

Michael J. Jordan

May 14, 2023

USERS GUIDE

1 Timecode **1**

2 TimecodeRange **3**

2.1 Timecode 3
2.2 TimecodeRange 5
2.3 Counting Modes 8
2.4 Timecode 9
2.5 TimecodeRange 10
2.6 Counting Modes 10
2.7 Footage (Feet+Frames) 11

Python Module Index **13**

Index **15**

CHAPTER

ONE

TIMECODE

Let's get ya started real quick here.

```
>>> # Import it
>>> from timecode import Timecode
```

You can create a *Timecode* from a frame number, or a timecode `str`.

```
>>> # Timecode from a frame number
>>> Timecode(86400)
<Timecode 01:00:00:00 @ 24 NDF>
```

```
>>> # Timecode from a string
>>> Timecode("01:00:00:00")
<Timecode 01:00:00:00 @ 24 NDF>
..
>>> # Protip: You don't need leading zeroes
>>> Timecode("30:00")
<Timecode 00:00:30:00 @ 24 NDF>
```

Specify a rate (see: *Specifying The Rate*)

```
>>> # Specify a rate
>>> Timecode("59:40", rate=30)
<Timecode 00:00:59:40 @ 30 NDF>
```

Specify a counting mode (see: *Specifying The Counting Mode*)

```
>>> # Specify a different counting mode
>>> from timecode.modes import DropFrame
>>> Timecode("32:19;28", mode=DropFrame())
<Timecode 00;32;19;28 @ 30 DF>
```

For more information, check out the *Timecode* section of the Users Guide.

CHAPTER

TWO

TIMECODERANGE

Define a range of frames by start, end, and/or duration!

```
>>> from timecode import Timecode, TimecodeRange
>>> tc_start = Timecode("00:59:59:00")
>>> tc_dur = Timecode("00:00:00:10")
>>> tc_range = TimecodeRange(start=tc_start, duration=tc_dur)
>>> repr(tc_range)
<TimecodeRange 00:59:59:00 - 00:59:59:10 (10) @ 24 NDF>
..
>>> Loop over it amd admore the frames for what they are
>>> for tc in tc_range:
...     print(tc)
...
00:59:59:00
00:59:59:01
00:59:59:02
00:59:59:03
00:59:59:04
00:59:59:05
00:59:59:06
00:59:59:07
00:59:59:08
00:59:59:09
```

For more information, check out the *TimecodeRange* section of the Users Guide.

Note: This project is under active development.

2.1 Timecode

`Timecode` is good.

class timecode.**Timecode**(*timecode: str | int | Timecode, mode: CountingMode | None = None, rate: int | None = None*)

Timecode representing a given frame number and rate

2.1.1 Specifying The Counting Mode

A *CountingMode* can be set during the creation of a *Timecode* object by setting the `mode` parameter.

NonDropFrame and *DropFrame* counting modes are available in the *timecode.modes* subpackage.

Timecode will default to *NonDropFrame* mode, unless otherwise specified.

```
>>> from timecode import Timecode
>>> from timecode.modes import DropFrame
..
>>> Timecode(86400, mode=DropFrame())
<Timecode 00;48;02;28 @ 30 DF>
```

The default counting mode can be set program-wide by assigning a *CountingMode* class to *timecode.Timecode.DEFAULT_MODE*

Note: Also included in the *timecode.modes* subpackage is an abstract class *CountingMode*, which can be subclassed to make your own weird little counting modes. Give it a shot!

See also: *Counting Modes*

2.1.2 Specifying The Rate

Timecode will default to the *DEFAULT_RATE* set by the *CountingMode*. Or, it may be explicitly set with the `rate` parameter.

```
>>> from timecode import Timecode
..
>>> Timecode("01:00:00:00", rate=30)
<Timecode 01:00:00:00 @ 30 NDF>
```

Warning: The *CountingMode* will validate the specified rate and may throw an exception if the rate is inappropriate. For example, *DropFrame* only accepts frame rates which are multiples of 30.

2.1.3 Math

So you got all these timecodes goin', but what do you do with them? Well I guess you can add them together:

```
>>> # Two Timecodes
>>> Timecode("01:00:01:00") + Timecode("02:03")
<Timecode 01:00:03:03 @ 24 NDF>
..
>>> # A Timecode and some frames
>>> Timecode("59:59:00") + 24
<Timecode 01:00:00:00 @ 24 NDF>
```

Oh! You can *resample()* from one kind to another:

```
>>> from timecode import Timecode
>>> from timecode.modes import DropFrame, NonDropFrame
..
>>> Timecode("00:48:20:12", mode=NonDropFrame()).resample(rate=30)
<Timecode 00:48:20:15 @ 30 NDF>
..
>>> Timecode("00:48:20:15", rate=30, mode=NonDropFrame()).resample(mode=DropFrame())
<Timecode 00;48;23;13 (88) @ 30 DF>
```

2.1.4 More Info

See *timecode.Timecode* in the API Documentation.

2.2 TimecodeRange

class timecode.**TimecodeRange**(*, *start:* Timecode | *str* | *int* | *None* = *None*, *duration:* Timecode | *str* | *int* | *None* = *None*, *end:* Timecode | *str* | *int* | *None* = *None*)

An unbroken sequence of timecodes between a specified range

2.2.1 Creating A Range

TimecodeRange requires at least two of the following parameters: `start`, `duration`, `end`. The third parameter will be calculated from the other two if it is not given.

```
>>> from timecode import Timecode, TimecodeRange
>>> from timecode.modes import DropFrame, NonDropFrame
..
>>> # Create a `TimecodeRange` from a start and duration
>>> tc_start    = Timecode("00:59:59:00")
>>> tc_duration = Timecode("01:02:00")
>>> TimecodeRange(start=tc_start, duration=tc_duration)
<TimecodeRange 00:59:59:00 - 01:01:01:00 (1488) @ 24 NDF>
```

```
>>> # Create a `TimecodeRange` from a start and an end
>>> tc_start  = Timecode("00:59:59:00")
>>> tc_end    = Timecode("01:01:01:00")
>>> TimecodeRange(start=tc_start, end=tc_end)
<TimecodeRange 00:59:59:00 - 01:01:01:00 (1488) @ 24 NDF>
```

`start`, `duration`, and `end` can be provided as *Timecode* objects, or as any of the standard types supported by the *Timecode* constructor (`str` or `int`). The types can be mixed and matched for each input parameter.

```
>>> # Create a `TimecodeRange` that is 48 frames long
>>> tc_start     = Timecode("01:00:00:00")
>>> frm_duration = 48
>>> TimecodeRange(start=tc_start, duration=frm_duration)
<TimecodeRange 01:00:00:00 - 01:00:02:00 (48) @ 24 NDF>
```

2.2.2 CountingModes and Rates

The *CountingMode* and rate of the *TimecodeRange* object is determined by the `mode` and `rate` of the input *Timecode* objects; or by the *Timecode* defaults if none are provided. Thus, to set the `mode` and/or `rate` for a *TimecodeRange*, at least one of the inputs should be a *Timecode* object with the desired settings.

```
>>> # Create a `TimecodeRange` that is 60 FPS drop frame (oh my)
>>> tc_start = Timecode("01:00:00;00", rate=60, mode=DropFrame())
>>> duration = 120
>>> TimecodeRange(start=tc_start, duration=duration)
<TimecodeRange 01;00;00;00 - 01;00;02;00 (120) @ 60 DF>
```

> **Warning:** If more than one of the input parameters are *Timecode* objects, they must all have matching *CountingMode*s and `rate`s. Otherwise, *TimecodeRange* will raise an exception.

2.2.3 Using TimecodeRange

```
>>> from timecode import Timecode, TimecodeRange
...
>>> tc_start = Timecode("01:00:00:00")
>>> tc_duration = Timecode("00:10")
>>> tc_range = TimecodeRange(start=tc_start, duration=tc_duration)
>>> tc_range
<TimecodeRange 01:00:00:00 - 01:00:00:10 (10) @ 24 NDF>
```

Accessing Properties

`start`, `duration`, and `end`, properties return *Timecode* objects.

```
>>> tc_range.start
<Timecode 01:00:00:00 @ 24 NDF>
..
>>> tc_range.start.frame_number
86400
..
>>> tc_range.duration
<Timecode 00:00:00:10 @ 24 NDF>
..
>>> tc_range.end
<Timecode 01:00:00:10 @ 24 NDF>
..
>>> tc_range.end.frame_number
86410
```

Iterating over a `TimecodeRange`

```
>>> len(tc_range)
10
..
>>> for tc in tc_range:
...     repr(tc)
...
'<Timecode 01:00:00:00 @ 24 NDF>'
'<Timecode 01:00:00:01 @ 24 NDF>'
'<Timecode 01:00:00:02 @ 24 NDF>'
'<Timecode 01:00:00:03 @ 24 NDF>'
'<Timecode 01:00:00:04 @ 24 NDF>'
'<Timecode 01:00:00:05 @ 24 NDF>'
'<Timecode 01:00:00:06 @ 24 NDF>'
'<Timecode 01:00:00:07 @ 24 NDF>'
'<Timecode 01:00:00:08 @ 24 NDF>'
'<Timecode 01:00:00:09 @ 24 NDF>'
```

Checking For Membership

```
>>> # Based on Timecode string
>>> "01:00:00:05" in tc_range
True
..
>>> # Based on frame number
>>> 86405 in tc_range
True
..
>>> # Based on a `Timecode` object
>>> Timecode("01:00:00:05") in tc_range
True
```

Subsets

```
>>> # Check if all frames in a `TimecodeRange` exist in another `TimecodeRange`
>>> tc_range_inner = TimecodeRange(start="01:00:00:05", duration=2)
>>> tc_range_inner in tc_range
True
```

2.2.4 More Info

See *timecode.TimecodeRange* in the the API Documentation.

2.3 Counting Modes

A *CountingMode* defines the frame counting behavior of a *Timecode* or *TimecodeRange* object. It also handles string formatting, as different modes may be represented differently (e.g. *DropFrame* timecode traditionally uses a `;` separator).

class timecode.modes.**CountingMode**

An abstract class to facilitate timecode frame counting modes

Two of the most common modes are provided: *NonDropFrame* and *DropFrame*. Additional modes may be created by subclassing the *CountingMode* class.

Note: The *CountingMode* classes are provided in the *timecode.modes* subpackage.

2.3.1 Using a Counting Mode

A *CountingMode* is typically provided by setting the `mode` parameter to an instance of a *CountingMode* during the creation of a *Timecode* object:

```
>>> from timecode import Timecode
>>> from timecode.modes import DropFrame, NonDropFrame
>>> Timecode("01:00:00:00", mode=DropFrame())
<Timecode 01;00;00;00 @ 30 DF>
```

See also: *Specifying The Counting Mode*

2.3.2 Defaults

A *CountingMode* defines a default `rate` to use if the `rate` is not explicitly set during the creation of a *Timecode* object. It may also define additional rules to validate the `rate`. Below are the defaults and rules for the out-of-the-box *CountingMode* classes:

Mode	Default Rate	Additional Rules
NonDropFrame	24	Rate must be a positive integer
DropFrame	30	Rate must be a positive integer, and a multiple of 30

Here we illustrate the default rates and rate validation:

```
>>> from timecode import Timecode
>>> from timecode.modes import DropFrame, NonDropFrame
..
>>> # NonDropFrame defaults to 24 fps
>>> Timecode("01:00:00:00", mode=NonDropFrame())
<Timecode 01:00:00:00 @ 24 NDF>
```

(continues on next page)

(continued from previous page)

```
..
>>> # DropFrame defaults to 30 fps
>>> Timecode("01:00:00:00", mode=DropFrame())
<Timecode 01;00;00;00 @ 30 DF>
..
>>> # DropFrame throws a `ValueError` for rates
>>> # that are not multiples of 30
>>> Timecode("01:00:00:00", mode=DropFrame(), rate=24)
ValueError: Drop Frame mode requires the rate to be a multiple of 30.
```

See also: *Specifying The Rate*

2.3.3 More Info

See `timecode.modes.CountingMode` in the API Documentation.

2.4 Timecode

class timecode.**Timecode**(*timecode: str* | *int* | Timecode, *mode:* CountingMode | *None* = *None*, *rate: int* | *None* = *None*)

Timecode representing a given frame number and rate

DEFAULT_MODE

alias of `NonDropFrame`

resample(*, *mode:* CountingMode | *None* = *None*, *rate: int* | *None* = *None*)

Create a new timecode object resampled to a new rate or frame counting mode

property frame_number: `int`

The timecode as a frame number

property rate: `int`

The rate (per second) of this timecode

property mode: `CountingMode`

The counting mode used with this timecode

property hours: `int`

The hours elapsed

property minutes: `int`

The minutes per hour elapsed

property seconds: `int`

The seconds per minute elapsed

property frames: `int`

The frames per second elapsed

property is_negative: `bool`

Is the timecode negative

property is_positive: bool

Is the timecode positive

2.5 TimecodeRange

class timecode.**TimecodeRange**(*, *start:* Timecode | *str* | *int* | *None* = *None*, *duration:* Timecode | *str* | *int* | *None* = *None*, *end:* Timecode | *str* | *int* | *None* = *None*)

An unbroken sequence of timecodes between a specified range

ALLOW_NEGATIVE_RANGES = False

Allow a start timecode which is later than the end timecode

property start: *Timecode*

The start timecode in this range

property end: *Timecode*

The end timecode in this range (exclusive)

property duration: *Timecode*

The duration of this timecode range

2.6 Counting Modes

class timecode.modes.**CountingMode**

Bases: ABC

An abstract class to facilitate timecode frame counting modes

SEPARATOR = ':'

The character expected and used for separation of elements

ALLOW_NEGATIVE_TIMECODE = True

Allow negative timecodes with this mode

DEFAULT_RATE = 24

The default frame rate to use if not provided

classmethod validate_rate(*rate:* *int* | *None* = *None*) → int

Validate and clean the user-provided rate

classmethod hours(*framenumber:* *int*, *rate:* *int*) → int

Hours element

classmethod minutes(*framenumber:* *int*, *rate:* *int*) → int

Minutes element

classmethod seconds(*framenumber:* *int*, *rate:* *int*) → int

Seconds element

classmethod frames(*framenumber:* *int*, *rate:* *int*) → int

Frames element

class timecode.modes.**NonDropFrame**

Bases: *CountingMode*

Non-drop-frame timecode mode

DEFAULT_RATE = 24

The default frame rate to use if not provided

SEPARATOR = ':'

The character expected and used for separation of elements

class timecode.modes.**DropFrame**

Bases: *CountingMode*

Drop-frame timecode mode

DEFAULT_RATE = 30

DF timecodes are intended for multiples of 30

SEPARATOR = ';'

The character expected and used for separation of elements

classmethod validate_rate(*rate: int | None = None*) → int

Validate and clean the user-provided rate

classmethod get_dropped_frames(*framenumber: int, rate: int*) → int

Calculate the number of frames to drop

classmethod hours(*framenumber: int, rate: int*) → int

Hours element

classmethod minutes(*framenumber: int, rate: int*) → int

Minutes element

classmethod seconds(*framenumber: int, rate: int*) → int

Seconds element

classmethod frames(*framenumber: int, rate: int*) → int

Frames element

2.7 Footage (Feet+Frames)

class timecode.**Footage**(*frames: str | int | None, frames_per_foot: int = 16*)

A rate-agnostic frame counter based on the number of frames per foot in various film formats

property feet: int

Number of full feet

PYTHON MODULE INDEX

t

timecode.modes, 10

INDEX

A

ALLOW_NEGATIVE_RANGES (*timecode.TimecodeRange attribute*), 10
ALLOW_NEGATIVE_TIMECODE (*timecode.modes.CountingMode attribute*), 10

C

CountingMode (*class in timecode.modes*), 10

D

DEFAULT_MODE (*timecode.Timecode attribute*), 9
DEFAULT_RATE (*timecode.modes.CountingMode attribute*), 10
DEFAULT_RATE (*timecode.modes.DropFrame attribute*), 11
DEFAULT_RATE (*timecode.modes.NonDropFrame attribute*), 11
DropFrame (*class in timecode.modes*), 11
duration (*timecode.TimecodeRange property*), 10

E

end (*timecode.TimecodeRange property*), 10

F

feet (*timecode.Footage property*), 11
Footage (*class in timecode*), 11
frame_number (*timecode.Timecode property*), 9
frames (*timecode.Timecode property*), 9
frames() (*timecode.modes.CountingMode class method*), 10
frames() (*timecode.modes.DropFrame class method*), 11

G

get_dropped_frames() (*timecode.modes.DropFrame class method*), 11

H

hours (*timecode.Timecode property*), 9
hours() (*timecode.modes.CountingMode class method*), 10
hours() (*timecode.modes.DropFrame class method*), 11

I

is_negative (*timecode.Timecode property*), 9
is_positive (*timecode.Timecode property*), 9

M

minutes (*timecode.Timecode property*), 9
minutes() (*timecode.modes.CountingMode class method*), 10
minutes() (*timecode.modes.DropFrame class method*), 11
mode (*timecode.Timecode property*), 9
module
    timecode.modes, 10

N

NonDropFrame (*class in timecode.modes*), 10

R

rate (*timecode.Timecode property*), 9
resample() (*timecode.Timecode method*), 9

S

seconds (*timecode.Timecode property*), 9
seconds() (*timecode.modes.CountingMode class method*), 10
seconds() (*timecode.modes.DropFrame class method*), 11
SEPARATOR (*timecode.modes.CountingMode attribute*), 10
SEPARATOR (*timecode.modes.DropFrame attribute*), 11
SEPARATOR (*timecode.modes.NonDropFrame attribute*), 11
start (*timecode.TimecodeRange property*), 10

T

Timecode (*class in timecode*), 9
timecode.modes
    module, 10
TimecodeRange (*class in timecode*), 10

V

`validate_rate()` (*timecode.modes.CountingMode class method*), 10
`validate_rate()` (*timecode.modes.DropFrame class method*), 11