
Town Clock

Release 1.1.1

Zack Hankin <zthankin@gmail.com>

Mar 26, 2023

CONTENTS:

1	Readme	1
2	Town Clock	3
2.1	Documentation	3
2.2	Installation	3
2.2.1	Install Dev	3
2.2.2	Install Docs	3
2.2.3	Install for use	3
3	Licence	5
4	town_clock	7
4.1	Subpackages	7
4.1.1	town_clock.clock package	7
4.1.1.1	Submodules	7
4.1.1.2	town_clock.clock.clock module	7
4.1.1.3	town_clock.clock.clock_tower module	8
4.1.1.4	town_clock.clock.pulses module	8
4.1.1.5	town_clock.clock.relay module	8
4.1.1.6	Module contents	8
4.1.2	town_clock.ui package	8
4.1.2.1	Submodules	8
4.1.2.2	town_clock.ui.lcd_screen module	8
4.1.2.3	Module contents	8
4.1.3	town_clock.util package	8
4.1.3.1	Submodules	8
4.1.3.2	town_clock.util.clock_exceptions module	8
4.1.3.3	town_clock.util.clock_logging module	8
4.1.3.4	town_clock.util.location_sunrise_sunset module	8
4.1.3.5	town_clock.util.utils module	9
4.1.3.6	Module contents	10
4.2	Modules	11
4.3	town_clock.controller module	11
4.4	Module contents	11
5	Indices and tables	13
	Python Module Index	15
	Index	17

CHAPTER

ONE

README

2.1 Documentation

Current main: [GitHub Pages](#)

All versions: [Read The Docs](#)

2.2 Installation

2.2.1 Install Dev

```
pip install poetry
poetry install --with=dev
```

2.2.2 Install Docs

```
pip install poetry
poetry install --with=docs
```

2.2.3 Install for use

```
pip install . -e
```

CHAPTER
THREE

LICENCE

TOWN_CLOCK

4.1 Subpackages

4.1.1 town_clock.clock package

4.1.1.1 Submodules

4.1.1.2 town_clock.clock.clock module

clock.py

```
class Clock(name: CLOCK, relay: ClockRelay, time_on_clock: int, slow: int = 0, cutoff: int = 30, sleep_time: float = 0.5)
```

Bases: object

Class Clock

Parameters

- **name** (CLOCK) – The name of the clock in enum form.
- **relay** (ClockRelay) – The relay that this Clock controls.
- **time_on_clock** (int) – minutes past 12 AM/PM (0-719)
- **slow** (int) – Minutes slow, fast is negative.
- **cutoff** (int) – Value is used to work out how long the clock will sleep for. Default is 30.

compare(clock_time: int) → Clock

Compares the time on the clock with the given time and works out how slow or fast it is.

The 'self.cutoff' value is used to work out how long the clock will sleep for.

Parameters

clock_time – int: Time.clock_time, minutes after 12 AM/PM.

Returns

self

cutoff: int = 30

name: CLOCK

pulse(num_pulses: int = 1) → Clock

Pulse the clock

relay: *ClockRelay*

sleep_time: float = 0.5

slow: int = 0

time_on_clock: int

class ClockRelay(*args, **kwargs)

Bases: Protocol

Relay Protocol

pulse()

pulse method

4.1.1.3 town_clock.clock.clock_tower module

4.1.1.4 town_clock.clock.pulses module

4.1.1.5 town_clock.clock.relay module

4.1.1.6 Module contents

4.1.2 town_clock.ui package

4.1.2.1 Submodules

4.1.2.2 town_clock.ui.lcd_screen module

4.1.2.3 Module contents

4.1.3 town_clock.util package

4.1.3.1 Submodules

4.1.3.2 town_clock.util.clock_exceptions module

4.1.3.3 town_clock.util.clock_logging module

4.1.3.4 town_clock.util.location_sunrise_sunset module

Calculates sun position.

find_sunrise_sunset_times(latitude: float, longitude: float, altitude: float) → dict[int, float]

timezone_finder(latitude: float, longitude: float)

4.1.3.5 town_clock.util.utils module

class **CLOCK**(*value*)

Bases: Enum

CLOCK Names A enum used for controlling and altering the clock times independently. todo: Use enum with objects..

ALL = 0

ONE = 1

TWO = 2

classmethod **values**() → tuple[Literal[0], Literal[1], Literal[2]]

class **Log_Level**(*value*)

Bases: Enum

Log Levels: CRITICAL: 50 EXCEPTION = 40 ERROR: 40 WARNING: 30 INFO: 20 DEBUG: 10 NOTSET: 0

CRITICAL = 50

DEBUG = 10

ERROR = 40

EXCEPTION = 40

INFO = 20

NOTSET = 0

PULSE = 45

SUCCESS = 25

WARNING = 30

class **Mode**(*value*)

Bases: Enum

Either Test or Active

DEV = 'dev' TEST = 'test' ACTIVE = 'active'

ACTIVE = 'active'

DEV = 'dev'

TEST = 'test'

convert_position_string_to_number(*position_str: str*) → float

Converts a position string to a number.

Parameters

position_str – str: Position in the form of a string. eg “30.3402W”

4.1.3.6 Module contents

Utility package for town-clock

All modules can use these utility functions and classes.

Author: Zack Hankin Started: 27/01/2023

class **CLOCK**(*value*)

Bases: Enum

CLOCK Names A enum used for controlling and altering the clock times independently. todo: Use enum with objects..

ALL = 0

ONE = 1

TWO = 2

classmethod **values**() → tuple[Literal[0], Literal[1], Literal[2]]

class **Log_Level**(*value*)

Bases: Enum

Log Levels: CRITICAL: 50 EXCEPTION = 40 ERROR: 40 WARNING: 30 INFO: 20 DEBUG: 10 NOTSET: 0

CRITICAL = 50

DEBUG = 10

ERROR = 40

EXCEPTION = 40

INFO = 20

NOTSET = 0

PULSE = 45

SUCCESS = 25

WARNING = 30

class **Mode**(*value*)

Bases: Enum

Either Test or Active

DEV = 'dev' TEST = 'test' ACTIVE = 'active'

ACTIVE = 'active'

DEV = 'dev'

TEST = 'test'

convert_position_string_to_number(*position_str: str*) → float

Converts a position string to a number.

Parameters

position_str – str: Position in the form of a string. eg “30.3402W”

find_sunrise_sunset_times(*latitude: float, longitude: float, altitude: float*) → dict[int, float]

timezone_finder(*latitude: float, longitude: float*)

4.2 Modules

4.3 town_clock.controller module

4.4 Module contents

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

t

`town_clock.clock.clock`, [7](#)
`town_clock.util`, [10](#)
`town_clock.util.location_sunrise_sunset`, [8](#)
`town_clock.util.utils`, [9](#)

A

ACTIVE (*Mode attribute*), 9, 10
ALL (*CLOCK attribute*), 9, 10

C

Clock (*class in town_clock.clock.clock*), 7
CLOCK (*class in town_clock.util*), 10
CLOCK (*class in town_clock.util.utils*), 9
ClockRelay (*class in town_clock.clock.clock*), 8
compare() (*Clock method*), 7
convert_position_string_to_number() (*in module town_clock.util*), 10
convert_position_string_to_number() (*in module town_clock.util.utils*), 9
CRITICAL (*Log_Level attribute*), 9, 10
cutoff (*Clock attribute*), 7

D

DEBUG (*Log_Level attribute*), 9, 10
DEV (*Mode attribute*), 9, 10

E

ERROR (*Log_Level attribute*), 9, 10
EXCEPTION (*Log_Level attribute*), 9, 10

F

find_sunrise_sunset_times() (*in module town_clock.util*), 11
find_sunrise_sunset_times() (*in module town_clock.util.location_sunrise_sunset*), 8

I

INFO (*Log_Level attribute*), 9, 10

L

Log_Level (*class in town_clock.util*), 10
Log_Level (*class in town_clock.util.utils*), 9

M

Mode (*class in town_clock.util*), 10

Mode (*class in town_clock.util.utils*), 9

module

town_clock.clock.clock, 7
town_clock.util, 10
town_clock.util.location_sunrise_sunset, 8
town_clock.util.utils, 9

N

name (*Clock attribute*), 7
NOTSET (*Log_Level attribute*), 9, 10

O

ONE (*CLOCK attribute*), 9, 10

P

PULSE (*Log_Level attribute*), 9, 10
pulse() (*Clock method*), 7
pulse() (*ClockRelay method*), 8

R

relay (*Clock attribute*), 7

S

sleep_time (*Clock attribute*), 8
slow (*Clock attribute*), 8
SUCCESS (*Log_Level attribute*), 9, 10

T

TEST (*Mode attribute*), 9, 10
time_on_clock (*Clock attribute*), 8
timezone_finder() (*in module town_clock.util*), 11
timezone_finder() (*in module town_clock.util.location_sunrise_sunset*), 8
town_clock.clock.clock
module, 7
town_clock.util
module, 10
town_clock.util.location_sunrise_sunset
module, 8

`town_clock.util.utils`
 `module`, [9](#)
`TWO` (*CLOCK attribute*), [9](#), [10](#)

V

`values()` (*CLOCK class method*), [9](#), [10](#)

W

`WARNING` (*Log_Level attribute*), [9](#), [10](#)