Town Clock

Release 1.1.1

Zack Hankin <zthankin@gmail.com>

Mar 26, 2023

CONTENTS:

CHAPTER

ONE

README

CHAPTER

TWO

TOWN CLOCK

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

CHAPTER

FOUR

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

CHAPTER

FIVE

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

t

INDEX

V

W