
daylio-parser

Release 0.1.0

Meesha

Sep 12, 2021

CONTENTS:

1	Installation	1
2	Config	3
3	Parser	5
4	PlotData	7
5	Stats	9
	Index	11

INSTALLATION

Via pip:

```
pip3 install --user daylio-parser
```

Via Pipenv:

```
pipenv install daylio-parser
```

Via Poetry:

```
poetry add daylio-parser
```


CONFIG

daylio-parser comes with a default config that works for the default Daylio setup after installing the app. That is, there's just 5 moods, called `awful`, `bad`, `meh`, `good`, `rad`.

Each mood has its class:

class Mood

name: str

Name of the mood, must correspond with mood name in the exported CSV.

level: int

Assigned numeral level for the mood (higher = better mood).

color: str

Any hex color.

boundaries: Tuple[float, float]

A tuple with lower and upper bound for the mood. Any average mood that falls within these boundaries will be colored using the `Mood.color`.

The whole mood config for your app will be constructed using the `MoodConfig` class.

class MoodConfig(*mood_list=None, color_palette=None*)

Creates a config with `mood_list`. If the mood list isn't provided, `DEFAULT_MOODS` will be used. All moods are automatically colored using `color_palette` and boundaries are also calculated. Each boundary is exactly 1 in size, with the first one and the last one being only 0.5 in size.

Parameters

- **mood_list** (*List[Tuple[int, str]]*) – A list of moods with (level, name)
- **color_palette** (*List[str]*) – A list of colors (hex values or common names)

from_list(*mood_list, color_palette=None*) → None

Updates the config with a new list of moods.

Parameters

- **mood_list** (*List[Tuple[str, str]]*) – A list of moods with (level, name)
- **color_palette** (*List[str]*) – A list of colors (hex values or common names)

get(*mood_name*) → `Mood`

Returns a `Mood` by its name.

Parameters **mood_name** (*str*) – Mood name

PARSER

class Entry

A class that holds data for an entry in the diary. One day can have multiple entries.

datetime: `datetime.datetime`

mood: `Mood`

activities: `List[str]`

notes: `str`

class Parser(*config=None*)

Parser for the CSV file. If config is not provided, a default one will be created.

Parameters **config** (`MoodConfig`) – MoodConfig for the parser

load_csv(*path*) → `List[Entry]`

Load entries from a CSV file.

Parameters **path** (*str*) – Path to the CSV file

load_from_buffer(*f*) → `List[Entry]`

Actually reads the entries from a CSV file.

Parameters **f** – A file-like object

PLOTDATA

class `PlotData`(*entries*, *config=None*)

A class that provides some data for easier plotting.

Parameters

- **entries** (*List*[`Entry`]) – A list of parsed entries
- **config** (`MoodConfig`) – `MoodConfig` for the parser (if none is provided, a default one will be created)

split_into_bands(*moods*)

Splits input moods into bands, given their boundaries. See `Mood.boundaries`.

interpolate(*avg_moods=None*, *interpolate_steps=360*)

Interpolates moods to make a smooth chart.

Parameters

- **avg_moods** – Average moods to iterate over. If not provided, these are generated by `Stats.average_moods()`
- **interpolate_steps** (*int*) – Number of steps for one day (midnight to midnight)

STATS

class MoodPeriod

This class represents a period of moods.

start_date: `datetime.datetime`

end_date: `datetime.datetime`

duration: `int`

Length of the period as a number of days.

avg_mood: `float`

Average mood for the whole period.

class Stats(*entries*, *config=None*)

A class for computing various stats from the entries.

Parameters

- **entries** (*List*[[Entry](#)]) – A list of parsed entries
- **config** ([MoodConfig](#)) – MoodConfig for the parser (if none is provided, a default one will be created)

average_moods() → *List*[*Tuple*[*datetime.date*, *float*]]

Computes average mood for each day.

activity_moods() → *Dict*[*str*, *Tuple*[*float*, *float*]]

Computes average mood and standard deviation for each activity. The returned dict has mood name as a key and (mean, std) as value.

mean() → *Tuple*[*float*, *float*]

Returns (mean, std) for all entries.

rolling_mean(*N=5*)

Computes a rolling mean for the entries.

Parameters *N* (*int*) – Window size

find_high_periods(*threshold=4*, *min_duration=4*) → *List*[[MoodPeriod](#)]

Find all periods of high moods.

Parameters

- **threshold** (*float*) – Find moods higher than this
- **min_duration** (*int*) – Find periods longer than this

find_low_periods(*threshold=3*, *min_duration=5*) → *List*[[MoodPeriod](#)]

Find all periods of low moods.

Parameters

- **threshold** (*float*) – Find moods higher than this
- **min_duration** (*int*) – Find periods longer than this

INDEX

A

activities (*Entry attribute*), 5
activity_moods() (*Stats method*), 9
average_moods() (*Stats method*), 9
avg_mood (*MoodPeriod attribute*), 9

B

boundaries (*Mood attribute*), 3

C

color (*Mood attribute*), 3

D

datetime (*Entry attribute*), 5
duration (*MoodPeriod attribute*), 9

E

end_date (*MoodPeriod attribute*), 9
Entry (*built-in class*), 5

F

find_high_periods() (*Stats method*), 9
find_low_periods() (*Stats method*), 9
from_list() (*MoodConfig method*), 3

G

get() (*MoodConfig method*), 3

I

interpolate() (*PlotData method*), 7

L

level (*Mood attribute*), 3
load_csv() (*Parser method*), 5
load_from_buffer() (*Parser method*), 5

M

mean() (*Stats method*), 9
Mood (*built-in class*), 3
mood (*Entry attribute*), 5
MoodConfig (*built-in class*), 3

MoodPeriod (*built-in class*), 9

N

name (*Mood attribute*), 3
notes (*Entry attribute*), 5

P

Parser (*built-in class*), 5
PlotData (*built-in class*), 7

R

rolling_mean() (*Stats method*), 9

S

split_into_bands() (*PlotData method*), 7
start_date (*MoodPeriod attribute*), 9
Stats (*built-in class*), 9