remind-me-some

Release 0.0.1

Audrow Nash

Sep 22, 2020

CONTENTS

Schedules some number of items that are due today.

Tasks that you don't get to are weighted to be more heavily in the future.

CHAPTER

ONE

FEATURES

- Repeatedly schedules tasks at a specified frequency
- Schedule a set number or less tasks each day
- Tasks that don't get done or scheduled will increase in priority
- Tested on Python 3.6, 3.7, and 3.8

CHAPTER

TWO

USAGE

```
$ git clone https://github.com/audrow/remind-me-some
$ pip install remind-me-some
```

```
from datetime import date, timedelta
from remind_me_some.goal import Goal
from remind_me_some.schedule_manager import ScheduleManager

goals = (
    ("Call Mom", timedelta(weeks=1)),
    ("Call Dad", timedelta(weeks=1)),
    ("Call Grandma", timedelta(weeks=2)),
    ("Call Grandpa", timedelta(weeks=2)),
    ("Call Cousin", timedelta(weeks=4)),
    ("Call Uncle", timedelta(weeks=4)),
)
goals_ = []
for goal in goals:
    goals_.append(Goal(name=goal[0], frequency=goal[1]))

sm = ScheduleManager()
sm.add_goals(*goals_)
sm.update_schedule()
print(sm)
sm.run()  # run the callback for the scheduled action
sm.run()  # clear the action if it's completed
print(sm)
```

CHAPTER

THREE

API DOCUMENTATION

If you are looking for information on a specific function, class, or method, this part of the documentation is for you.

3.1 Developer Interface

This part of the documentation covers all the interfaces of Remind-Me-Some.

3.1.1 Schedule Manager

class remind_me_some.**ScheduleManager**(*max_actions_per_day: int = 1, is_exclude_date_fn: Callable[[datetime.date], bool] = <function is_exclude_date>*)

The schedule manager class.

__init__(*max_actions_per_day: int = 1, is_exclude_date_fn: Callable[[datetime.date], bool] = <function is_exclude_date>*) → None

Initialize the schedule manager.

Parameters

- **max_actions_per_day** – The max number of actions that should occur on any day.
- **is_exclude_date_fn** – A function that return True if a date should be excluded and false otherwise. This can be used to avoid scheduling actions on weekends, holidays, etc.

property actions

Get the active actions.

Returns A list of active actions.

add_goal(*goal: remind_me_some.goal.Goal*) → None

Add one new goal.

Parameters **goal** – A goal to add.

add_goals(**goals: remind_me_some.goal.Goal*) → None

Add one or more new goals.

Parameters **goals** – One or more goals.

property goals

Get the current goals.

Returns A list of current goals.

run() → None
Execute or complete ready actions.

update_schedule() → None
Update the schedule to balance actions.

3.1.2 Exclude Date Function

remind_me_some.**is_exclude_date**(*date_: datetime.date*, *is_exclude_holidays: bool = True*, *is_exclude_weekends: bool = True*, *is_exclude_friday: bool = False*) → bool
Return True if a date should be excluded.

Parameters

- **date** – The date to consider.
- **is_exclude_holidays** – True if you would like to exclude holidays.
- **is_exclude_weekends** – True if you would like to exclude weekends.
- **is_exclude_friday** – True if you would like to exclude Fridays.

Returns True if the date should be excluded; false otherwise.

3.1.3 Data Structures

Goal class

class remind_me_some.**Goal**(*name: str*, *frequency: datetime.timedelta*, *priority: float = 1.0*, *interest_rate: float = 0.05*, *last_completed: Optional[datetime.date] = None*, *callback: Optional[Callable] = None*, *is_ready_fn: Optional[Callable] = None*, *is_completed_fn: Optional[Callable] = None*)
Bases: *remind_me_some.event.Event*

The goal class.

__init__(*name: str*, *frequency: datetime.timedelta*, *priority: float = 1.0*, *interest_rate: float = 0.05*, *last_completed: Optional[datetime.date] = None*, *callback: Optional[Callable] = None*, *is_ready_fn: Optional[Callable] = None*, *is_completed_fn: Optional[Callable] = None*) → None
Initialize a goal object.

Goal objects are used to create action objects at some frequency. Most of the information given to the goal object is used to create new action objects.

Parameters

- **name** – The name of the event.
- **frequency** – How often this goal should be completed.
- **priority** – The starting priority an action this goal generates (for determining its relative importance).
- **interest_rate** – The rate that the priority of a generated action grows each day it is pushed back past its original due date.
- **last_completed** – The date that this goal was last completed.
- **callback** – A function to be called when a generated action is run.

- **is_ready_fn** – A function to determine if a generated action is ready. If nothing is supplied this will default to be on or after the action's due date.
- **is_completed_fn** – A function to determine if the generated action has been completed. If nothing is supplied, this will default to be true if the callback has been called at least once.

property last_completed
Get the date when this goal was last completed.

Returns The last date that this goal was completed or None, if it hasn't been completed yet.

make_action() → remind_me_some.action.Action
Generate a new action instance.

Returns An action object.

mark_as_completed() → None
Set the last completed date to today's date.

Action class

class remind_me_some.**Action**(*name: str*, *due: datetime.date*, *priority: float*, *interest_rate: float*, *callback: Optional[Callable[], None]] = None*, *is_ready_fn: Optional[Callable[], bool]] = None*, *is_completed_fn: Optional[Callable[], bool]] = None*)
Bases: *remind_me_some.event.Event*

The action class.

__init__(*name: str*, *due: datetime.date*, *priority: float*, *interest_rate: float*, *callback: Optional[Callable[], None]] = None*, *is_ready_fn: Optional[Callable[], bool]] = None*, *is_completed_fn: Optional[Callable[], bool]] = None*) → None
Initialize an action.

Parameters

- **name** – The name of the action.
- **due** – The planned date for the action to be completed on.
- **priority** – The priority of the action (for determining its relative importance).
- **interest_rate** – The rate that the priority of the action grows each day it is pushed back past its original due date.
- **callback** – A function to be called when the action is run.
- **is_ready_fn** – A function to determine if the action is ready. If nothing is supplied this will default to be on or after the action's due date.
- **is_completed_fn** – A function to determine if the action has been completed. If nothing is supplied, this will default to be true if the callback has been called at least once.

is_due() → bool
Check if the current action is due.

Returns True if the current date is the due date or after; False, otherwise.

push_forward(*days: int = 1*) → None
Bump the due date of the current action and add interest.

Parameters **days** – The number of days to bump the due date by.

Event class

class remind_me_some.event.**Event**(*name: str*, *priority: float*, *interest_rate: float*, *callback: Optional[Callable[], None]] = None*, *is_ready_fn: Optional[Callable[], bool]] = None*, *is_completed_fn: Optional[Callable[], bool]] = None*)

The event class.

__init__(*name: str*, *priority: float*, *interest_rate: float*, *callback: Optional[Callable[], None]] = None*, *is_ready_fn: Optional[Callable[], bool]] = None*, *is_completed_fn: Optional[Callable[], bool]] = None*) → None

Initialize an event.

Parameters

- **name** – The name of the event.
- **priority** – The priority of the event (for determining its relative importance).
- **interest_rate** – The rate that the priority of the event grows each step it is pushed back past its original due date.
- **callback** – A function to be called when the event is run.
- **is_ready_fn** – A function to determine if the event is ready. If nothing is supplied, this will default to be true if the event has not been completed.
- **is_completed_fn** – A function to determine if the event has been completed. If nothing is supplied, this will default to be true if the callback has been called at least once.

callback() → Any

Call the event's callback.

Returns Whatever the callback returns.

is_called() → bool

Check if the event has been called.

Returns True if the callback has been called at least once; false otherwise.

is_completed() → bool

Check if an event has been completed.

Returns True if the is completed function returns true; false otherwise.

is_due() → bool

Check if the event is due.

Returns True if the completion function doesn't return true; false otherwise.

is_ready() → bool

Check if an event is ready.

Returns True if the ready function returns true and the event has not been completed; false otherwise.

push_forward(*steps: int = 1*) → None

Increase the priority of event by applying interest.

Parameters **steps** – The number of times to apply the interest rate to the event's priority.

CHAPTER

FOUR

ISSUES

If you encounter any problems, please file an issue along with a detailed description.

CHAPTER

FIVE

ABOUT REMIND-ME-SOME

Remind-Me-Some was created by Audrow Nash - audrow@hey.com

Distributed under the MIT license. See `LICENSE.txt` for more information.

PYTHON MODULE INDEX

r

INDEX

Symbols

A

C

E

G

I

L

M

P

R

S

U