OVERVIEW

someday is a frontend for the command-line calendar application **when** for Linux. It allows easy management of calendars through a menu-driven application.

PROGRAM INVOCATION

```
[python3] someday.py [options]
```

DESCRIPTION

someday is a frontend for **when**, a freely-available command-line calendar application written by Benjamin Crowell (https://www.lightandmatter.com/when/when.html). You can use it on top of **when** to manage your calendars easily through menus. Crucially, it has a feature to allow easy rescheduling. **someday** is intended for the professional procrastinator who wants to schedule a task using **when** and once the date arrives, reschedule it easily if it cannot be done at once. Hopefully, all due tasks get done “someday”.

Other features included: easy creation, deletion, duplication and edition of items; one-key browsing from calendar items containing URLs; search (plain-text and regular expressions); view modes; and a handy three-month calendar.

TARGET PLATFORMS

This app is intended for Linux/Unix (if there’s an X, it will probably work). Maybe you can make it run in other platforms; dependencies are minimal. If you succeed at using this app in other platforms, and want to share your experience (and be credited), please drop me a line at:

https://github.com/estebanflamini/someday/issues

KNOWN BUGS

The program has been tested in Linux, and no *obvious* bugs seem to exist. If you find any bug, please open an issue at:

https://github.com/estebanflamini/someday/issues

DEPENDENCIES

Python >= 3.7 (https://www.python.org) with **curses**.

when (https://www.lightandmatter.com/when/when.html)

CONTRIBUTING

If you would like to contribute to this project, please drop me a line at:

https://github.com/estebanflamini/someday/issues

All contributions will be credited.

INSTALLATION

For the time being, no installation scripts are provided. You will have to install the required dependencies and copy the program file yourself.

In the following instructions, all URLs are given as of the time of this writing. If some link is broken, please use an Internet search engine and your best judgment.

1. Copy **someday.py** to a directory included in the search path for executable files.
2. Check that you have Python >= 3.7 installed:

   ```
   python3 --version
   ```

   In the extremely unlikely event that you don't have Python3 installed, you can get it from a standard repository (if you have an older Python installed, you might want to install **python3-venv** and put Python 3.7+ inside a virtual environment, so it does not interfere with the older installation. See: https://docs.python.org/3/library/venv.html).
3. Check that you have **curses** installed. Run Python and try to import the **curses** module:

   ```
   python3 -c 'import curses'
   ```

   If you receive an error message, you should add **curses** to your Python installation. Usually you can do that using **pip**:

   ```
   pip install curses
   ```
4. Install **when** (either from the standard repositories using your default package manager or download it from https://www.lightandmatter.com/when/when.html and follow the instructions there).

   For example, to install **when** from the standard repositories in Debian:

   ```
   sudo apt-get install when
   ```

If you find any error in these instructions, or if you would like to share detailed installation instructions for your platform (and get credited for doing so), please contact me by opening an issue at:

https://github.com/estebanflamini/someday/issues

USING SOMEDAY

Before using **someday**, you should read **when**'s manual and run it at least once in order to create the application's directory, configuration files and a calendar. **someday** does not create a calendar of its own, it shares the calendars with **when**.

Spend a few days learning to use **when** and its very powerful language for defining calendars.

You will use **someday** as a complement, not a replacement, for **when**.

Below I give you a brief overview of some key concepts you need to understand before using **someday**.

A brief overview of when

when is a free command-line calendar application written by Benjamin Crowell (https://www.lightandmatter.com/when/when.html) which lets you define one or more calendars as plain-text files consisting of items with the following syntax:

```
<date> , <task>
```

`<date>` can be an exact date, usually in the format `YYYY MM DD` (e.g., `2023 07 15`), but you can also spell the month out (e.g., `2023 jul 15`) or replace some parts with wildcards (e.g., `2023 * 15`). A full explanation of the date format is given in **when**'s manual.

Crucially, `<date>` can also be a formula based on 'fancy tests', that is, a Boolean expression that will be "true" or "false" depending on the specific date. That will result in the task being scheduled on those dates where the formula is true. For example, the following line in a calendar

```
d=5 & j>60101 , Pay the rent
```

will show the task 'Pay the rent' every fifth of the month, but only for dates which come after the modified Julian day 60101 (which corresponds to `2023 06 05`). The meaning of this line could be: 'I have to pay the rent every fifth of the month, and the last time I paid the rent was June 5, 2023'.

When you call **when** without any options, it shows all tasks which are scheduled within its default interval (starting yesterday and ending 14 days after today). The date range can be modified using command-line options. If you want to delete or edit tasks, you need to open the underlying calendar file in a text editor. **when** provides you a handy shortcut for doing that; just enter the command

```
when e
```

and **when** will invoke your default text editor to edit the default calendar.

The rest of this manual assumes that you have a good understanding of **when** and feel comfortable scheduling items either for a specific date or based on 'fancy tests'. If that is not your case yet, please stop reading now, spend some time learning to use **when**, and then come back for the rest of this manual.

Using someday as a complement of when

Enter **someday**. When you invoke it without arguments (by writing `someday.py` in the command-line) it will show you exactly the same tasks that **when** would have shown you, but this time in a browseable list and with a menu of actions which can be applied to the selected task. This way you can see and edit your calendar in the same application. (You will still use **when** separately for those things that you can't do or don't want to do within **someday**.)

Use the **up** and **down** arrow keys to browse the list. On any selected item, press **Enter** to show the item in a popup box; that is useful when an item is too long and extends beyond the right edge of the screen.

As you browse the list, a menu below will show you the actions you can apply to the selected item. To invoke an action, either use the **left** and **right** arrow keys to select it and then press the **spacebar**; or press the accelerator key (the uppercase letter shown in the menu).

Editions made with **someday** are kept in an internal buffer and written to the underlying calendar only when you quit **someday**. If you want to discard all editions done in the current session, press **Ctrl-C** in the browseable list: that quits **someday** without writing to the calendar.

Rescheduling and advancing items

someday was written for the professional procrastrinator who wants to schedule a task and then move it around easily if it cannot be completed on the given date. Two types of 'moving around' operations exist in **someday**: rescheduling and advancing. The difference lies on whether it is an item which happens only once or a recurring item.

Above we've seen an example of a task scheduled for a specific day of the month and after a specific date:

```
d=5 & j>60101 , Pay the rent
```

This item will show up every fifth of the month after June 5, 2023. Such items are 'recurring items', that is, they show up at several points of time.

There are also items that are scheduled to happen only once at a specific date, e.g.:

```
2023 06 06 , File for bankruptcy
```

Items which happen only once at a specific date can be rescheduled, deleted or commented out from the calendar using **someday**.

Recurring items cannot be rescheduled, deleted nor commented out using **someday**. That is a protection against inadvertently cancelling all occurrences of a task in the future when you only meant to cancel the one selected in the browseable list.

The only timing operation you can apply to a recurring item is to 'advance' it. To advance a recurring item means to rewrite the underlying formula in such a way that every occurrence of the item before and up to today will not show up again, but every occurrence after today will be unaffected. Suppose today is June 29, 2023. You select the 'pay the rent' item shown above and then invoke the 'advance' action on it. **someday** will edit the formula to be

```
d=5 & j>60125 , Pay the rent
```

because 60125 is the modified Julian date corresponding to June 29, 2023. Now the item will show up every fifth of the month, but the first occurrence will be on July 5, 2023. This way you can keep track of a recurring task and the last time you've done it: you 'advance' (the starting date of) the task.

In addition to recurring items based on a Julian date, **someday** is also able to detect recurring items based on year, e.g.:

```
d=31 & m=1 & y>2022 , Congratulate Guido for his birthday!
```

This item will show up every January 31, starting on January 31, 2023. If you advance such an item, **someday** will rewrite the formula replacing today's year instead of the given year. E.g.:

```
d=31 & m=1 & y>2023 , Congratulate Guido for his birthday!
```

How does someday detect recurring items

For an item to be considered recurring by **someday**

- its date must be given as a formula based on 'fancy tests'
- with one and only one occurrence of a test of the form `j>N` or `y>N`
- in such a way that the formula cannot be possibly true unless the above test is true.

If you know some Boolean logic, the last condition means that the formula could be rewritten as:

```
   j>N & ( <rest of the formula> )
```

or

```
   y>N & ( <rest of the formula> )
```

The ‘rest of the formula’ part specifies the item’s recurrence pattern, and the year/Julian-date test sets a starting point for the item’s recurrence. **someday** cannot advance an item without such starting point.

Items which are neither recurring nor date-specific

Using **when** you can schedule items that will happen more than once but will fail the recurrence test given above. An example of such an item could be

```
   * 01 31 , Guido’s birthday
```

That is a perfectly valid item for **when**, that will show up every January 31. But it is not a ‘recurring’ (=advanceable) item for **someday**, nor it is an item you can reschedule, delete or comment out from the calendar using **someday**. In order to delete, comment or modify the timing of such an item, you need to edit the underlying calendar in a text editor. But you can also use the trick given in the following section.

The edit action

The ‘edit’ action is enabled for every item shown in the list. That is because the edition screen will show the entire underlying line from the calendar, and you can see if it is an item scheduled for a specific date or based on a formula, so there is no risk of *inadvertently* cancelling all occurrences of a recurring item when you only meant to cancel one of them.

If none of the ‘delete’, ‘reschedule’, ‘comment’, and ‘advance’ actions is available for an item, you can edit the item and adjust the formula/date expression until it is either a recurring item or an item that will happen only once. That will enable some of the missing actions.

You can also use this feature to convert an item which is scheduled for a specific date (the only kind you can create using the ‘new’ action within **someday**) to an item based on a formula.

And you can delete any item (even recurring ones) by editing it to be an empty line.

Now it is time to talk about the available actions in **someday**.

Actions

Name:	Edit
Accelerator key:	E
Availability:	All items
Meaning:	Edit the underlying line from the calendar corresponding to the current item. You will be able to edit the whole line, as if you opened the calendar in an external text editor. **someday** will not let you quit the edition screen if the line you enter is not valid for **when**. Press **Ctrl-C** if you want to cancel all changes. Note: you can delete any item (even recurring ones) by editing it to be an empty line.

Name: Done/delete

Accelerator key: D

Availability: Items which happen only once.

Meaning: Delete the underlying line from the calendar. That means you've done the task already, didn't you? Or maybe you're being honest to yourself, after realizing you will never do the task. If you are exactly the same type of procrastinator as I am, you may notice that many tasks you schedule, and then reschedule, and then reschedule, and then…, are rendered obsolete (or most likely, shown to never have been truly necessary) by the mere passing of time, the greatest solver of all our problems.

Name: Reschedule

Accelerator key: R

Availability: Items which happen only once.

Meaning: An edition screen appears where you can enter a new date for the item. You can enter the date as an exact date in the form `YYYY MM DD` or as an integer; a positive integer means how many days after today (1=tomorrow); a negative integer means how many days before today (-1=yesterday). Why on Earth would anyone want to reschedule an item to show up in the past, I do not know; but **someday** is agnostic enough to let you do it if that is what you want. Press **Enter** without entering nothing to return without changes. You can also press **Ctrl-C** to cancel any changes.

Name: Comment

Accelerator key: C

Availability: Items which happen only once.

Meaning: The underlying line in the calendar will be commented out. This is useful if you do not want a specific item to show up again, but want to keep the line in the calendar just in case.

Name: Advance

Accelerator key: A

Availability: Recurring items as understood by **someday** (see above).

Meaning: The underlying line in the calendar will be modified so that all occurrences of the item before and up to today (if the formula contains a test of the form `j>N`) or before and up to this year (if the formula contains a test of the form `y>N`) will not show up again, and all occurrences of the item after that will remain unaffected.

Name: Browse url

Accelerator key: B

Availability: Items which contain one or more URLs of the form `http://`... or `https://...`

Meaning: **someday** will try to open the URLs in your default browser.

Name: dUplicate

Accelerator key: U

Availability: All items

Meaning: **someday** will duplicate the underlying line from the calendar. This is useful to use one item as a template for another: duplicate the original item and then edit the copy.

Name: New

Accelerator key: N

Availability: Always (even when the list is empty)

Meaning: **someday** will ask you to enter an item and then the date. You can enter the date as an exact date in the form `YYYY MM DD` or as an integer; a positive integer means how many days after today (1=tomorrow); a negative integer means how many days before today (-1=yesterday). Why on Earth would anyone want to schedule an item to show up in the past, I do not know; but **someday** is agnostic enough to let you do it if that is what you want. Press **Enter** without entering nothing to return without changes. You can also press **Ctrl-C** to cancel any changes. Note: if you want to schedule an item based on a formula instead of a date, the simplest way to do that within **someday** is to duplicate an existing item and then edit the copy. The edition screen will give you full access to the underlying line in the calendar.

Name: View

Accelerator key: V

Availability: Always (even when the list is empty)

Meaning: **someday** will show you a menu of ‘views’ to change which items show up in the browseable list. (This includes changing the date range being shown and searching for an item over some date range.) A detailed explanation of views is given below.

Name: Monthly cal.

Accelerator key: M

Availability: Always (even when the list is empty)

Meaning: **someday** will show a three-month calendar, with the current date highlighted. Handy, isn’t it?

command-line arguments

someday can be invoked with the following command-line arguments:

`-h, --help`

Show a help message and exit.

`-v, --version`

Show the program's version and exit.

`--calendar <file>`

Use the given file as source calendar, instead of **when**'s default calendar.

`--past <number>`

Starting date of the date range to show. Negative numbers are in the past, positive numbers are in the future (0=today). This argument will be passed directly to **when**.

`--future <number>`

Ending date of the date range to show. Negative numbers are in the past, positive numbers are in the future (0=today). This argument will be passed directly to **when**.

`--past-for-search <number>`

`--future-for-search <number>`

When you use **someday**'s search feature, it will ask you not only a string or regular expression, but also the date range to show. Using these arguments you can specify a default range so that **someday** will not ask you for a date range every time you do a search. The meaning of the numbers is the same as for `--past` and `--future`. Note: if you specify only one of the arguments, **someday** will ask you for the other one every time you do a search.

`--search SEARCH`

`--regex REGEX`

Show items based on either a string or a regular expression. The date range will be the default one, unless you also give the `--past` and/or `--future` arguments. The regular-expression language is the one used by Python, which is similar to Perl's. Do not enclose the regular expression within delimiters.

`--diff`

Show the difference between the new edited calendar and the old one when exiting **someday** with changes.

Views

someday includes several predefined views you can use to change the list of items being shown. You activate a view by invoking the 'view' action and then typing a number. Predefined views are as follow:

Use given arguments: the default view, based on whatever `--past`, `--future`, and `--search`/`--regex` arguments you passed to **someday** in the command-line.

Use when's defaults: override any `--past`, `--future`, and `--search`/`--regex` arguments you passed to **someday** in the command-line, and use **when**'s defaults instead.

Enter a date range: show items for a given date range.

Search a string: show items containing a given (case insensitive) string. **someday** will ask you the desired date range, or you can specify it using the `--past-for-search` and `--future-for-search` command-line arguments.

Search a regex: show items matching a given (case insensitive) regular expression. **someday** will ask you the desired date range, or you can specify it using the `--past-for-search` and `--future-for-search` command-line arguments. The regular-expression language is the one used by Python, which is similar to Perl's. Do not enclose the regular expression within delimiters.

User defined views

In addition to its predefined views, **someday** allows you to create user views. Just create a plain-text file named `someday.viewmodes` and place it in the same directory as `someday.py`. Each view in the file must follow the format:

```
    <name> = <arguments>
```

where `<name>` can be any string (spaces are allowed, but an equal sign is not allowed) and `<arguments>` is a combination of the `--past`, `--future`, and `--search` arguments to define the view. For example:

```
    Rent payments next year = --search="rent" --past=365 --future=730
```

User-defined views are activated from the menu of views by typing **u** and then a number.