tasks3 Documentation

Release 0.8.0

Harsh Parekh

Jul 30, 2023

Contents:

CHAPTER 1

tasks3

A commandline tool to create and manage tasks and todos.

Most task management tools are their own applications, so to manage tasks you have to perform context switching by leaving what you're working on to go to the task manager application.

`tasks3` aims to solve that by bringing your tasks to you instead.

Each task is automatically assigned to the directory it was created in and you can easily retrieve tasks under a directory.

- Free software: GNU General Public License v3
- Documentation: https://tasks3.readthedocs.io.

1.1 Features

1.1.1 Help

It is easy to explore all capabilities of `tasks3` by running `tasks3 --help`. Each command also has its own help page which can be accessed by running:

```
$ tasks3 <command> --help
```

1.1.2 Create Tasks

Easily create tasks from the commandline and delegate them to folders.

Create a task in a specific folder with default settings.

```
$ tasks3 add --title "Think of a cool name" \
    --folder "~/Documents/story" \
    --yes
Added Task:
[e1c100] Think of a cool name (    ) (  )
  [path: ~/Documents/story]
```

Create a task in a current folder with custom settings and description.

```
$ tasks3 add --title "Try new model" \
    --urgency 4 --importance 3 \
    --description "Try:\n - model with 3 layers.\n - model with 4 layers." \
    --yes
Added Task:
[a0a5f4] Try new model () ( )
    Try:
     - model with 3 layers.
     - model with 4 layers.
```

1.1.3 Edit Existing Tasks

You can edit existing tasks with the `tasks3 edit` command.

For example: You can use `edit` to update the urgency of a task.

```
$ tasks3 edit --urgency 4 e1c100
Updated Task:
[e1c100] Think of a cool name () (  )
  [path: ~/Documents/story]
```

1.1.4 Search Tasks

You can search for tasks using various filters.

You can search for tasks with a specific importance value.

```
$ tasks3 search --importance 2
[4a14d0] What is right here and now
[f79155] Think of a cool name [path: /home/<user>/Documents/project]
[2ce91b] See home [path: /home]
```

You can restrict search to a folder and its sub-directories.

```
$ tasks3 search --folder ~/Documents/project --output-format yaml
title: Think of a Cool name
urgency: 2
importance: 2
tags: null
folder: /home/<user>/Documents/project
```

You can also search for sub-strings in task title or description. It is also possible to restrict the search to tasks that have a specific set of tags. Run `tasks3 search --help` to get see a full list off options.

1.1.5 Show Tasks

You can show all tasks under current directory.

```
$ tasks3 show
[a0a5f4] Try new model () ( )
    Try:
     model with 3 layers.
     model with 4 layers.
[4a14d0] What is right here and now (    ) (  )
```

You can also show a particular task by specifying its id.

```
$ tasks3 show 1d8a9a
[1d8a9a] Give a Title to this Task. (    ) ()
  (Hello tasks3)
    Task with
    multi-line
    desc
```

If you prefer to see the task in a different format, you can use the `--output-format` option.

```
$ tasks3 show --output-format json 1d8a9a
{
  "id": "1d8a9a",
  "title": "Give a Title to this Task.",
  "urgency": 2,
  "importance": 4,
  "tags": [
    "Hello tasks3"
  ],
  "folder": "/home/<user>/Documents/tasks3",
  "description": "Task with \nmulti-line \ndesc"
}
```

1.1.6 Complete Tasks

You can use the `tasks3 mark <task_id>` or `tasks3 edit --done <task_id>` command to mark a task as completed.

```
$ tasks3 mark 2e0b84
[2e0b84] Adding support for task completion (    ) (  )
```

1.1.7 Delete Tasks

You can use the `tasks3 delete <task_id>` command to delete a task.

Note: Deleting is a destructive action, prefer to mark the task as complete to hide it.

```
$ tasks3 remove --yes 2e0b84
Removed Task: [2e0b84] Adding support for task deletion (    ) (  )
```

1.1.8 Shell Integration

tasks3 supports shell integration for bash, zsh, and fish; tasks3 will automatically run `tasks3 show -o oneline` when you `cd` into a directory to show the tasks in that directory.

You can setup shell integration by adding the following command to your `.rc` file.

```
eval "$(tasks3 shell $(basename $SHELL))"
```

Note: Pull requests to support additional shells are greatly appreciated. Please see Contributing page for information on how to contribute.

1.2 Credits

This package was created with Cookiecutter and the audreyr/cookiecutter-pypackage project template.

CHAPTER 2

Installation

2.1 Stable release

To install tasks3, run this command in your terminal:

```
$ pip install tasks3
```

This is the preferred method to install tasks3, as it will always install the most recent stable release.

If you don't have pip installed, this Python installation guide can guide you through the process.

2.2 From sources

The sources for tasks3 can be downloaded from the Github repo.

You can either clone the public repository:

```
$ git clone git://github.com/hXtreme/tasks3
```

Or download the tarball:

```
$ curl -OJL https://github.com/hXtreme/tasks3/tarball/master
```

Once you have a copy of the source, you can install it with:

```
$ pip install -e .
$ pip install -r requirements_dev.txt
```

CHAPTER 3

Usage

To use tasks3 in a project:

```
import tasks3
```

CHAPTER 4

Contributing

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

You can contribute in many ways:

4.1 Types of Contributions

4.1.1 Report Bugs

Report bugs at https://git.parekh.page/tasks3/issues.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

4.1.2 Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with "bug" and "help wanted" is open to whoever wants to implement it.

4.1.3 Implement Features

Look through the GitHub issues for features. Anything tagged with "enhancement" and "help wanted" is open to whoever wants to implement it.

4.1.4 Write Documentation

tasks3 could always use more documentation, whether as part of the official tasks3 docs, in docstrings, or even on the web in blog posts, articles, and such.

4.1.5 Submit Feedback

The best way to send feedback is to file an issue at https://git.parekh.page/tasks3/issues.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

4.2 Get Started!

Ready to contribute? Here's how to set up *tasks3* for local development.

1. Fork the *tasks3* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/tasks3.git
```

3. Install your local copy into a virtualenv. Assuming you have virtualenvwrapper installed, this is how you set up your fork for local development:

```
$ mkvirtualenv tasks3
$ cd tasks3/
$ pip install -e .
$ pip install -r requirements_dev.txt
```

4. Create a branch for local development:

```
$ git checkout -b name-of-your-bugfix-or-feature
```

   Now you can make your changes locally.

5. When you're done making changes, check that your changes pass flake8 and the tests, including testing other Python versions with tox:

```
$ flake8 tasks3 tests
$ tox
```

   To get flake8 and tox, just pip install them into your virtualenv.

6. Commit your changes and push your branch to GitHub:

```
$ git add .
$ git commit -m "Your detailed description of your changes."
$ git push origin name-of-your-bugfix-or-feature
```

7. [Optional] Add your details (alias, email, website) to `AUTHORS.rst` file.
8. Submit a pull request through the GitHub website.

4.3 Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.
2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in README.rst.
3. The pull request should work for Python 3.9, 3.10, and for PyPy. Check https://git.parekh.page/tasks3/actions/workflows/tox-test.yml and make sure that the tests pass for all supported Python versions.

4.4 Tips

To run a subset of tests manually you can install pytest and run:

```
$ pytest tests.test_tasks3
```

4.5 Deploying

A reminder for the maintainers on how to deploy. Make sure all your changes are committed (including an entry in HISTORY.rst). Then run:

```
$ bump2version patch # possible: major / minor / patch
$ git push
$ git push --tags
```

A github workflow will trigger if tests pass and it will deploy the package to PyPI.

CHAPTER 5

Credits

5.1 Development Lead

- Harsh Parekh <harsh_parekh@outlook.com>

5.2 Contributors

None yet. Why not be the first?

CHAPTER 6

History

6.1 0.8.0: (2023-02-22)

- Add support to mark tasks as done.
- Update dependencies

6.2 0.7.0: (2022-07-30)

- Added support to delete a task from the cli.
- Updated dependencies.

6.3 0.6.0 (2022-05-16)

- Added support to edit existing tasks.
- Update dev-requirements.

6.4 0.5.1 (2022-05-06)

- Added shell integration for fish.

6.5 0.5.0 (2022-05-06)

- Added shell integration for zsh and bash.
- Improve the index page.

- Add more info to Contributing page.

6.6 0.4.4 (2022-05-03)

- Improve docs

6.7 0.4.3 (2022-05-03)

- Fix python version in setup.py

6.8 0.4.2 (2022-05-03)

- Upgrade development status to Alpha.

6.9 0.4.1 (2022-05-03)

- Resolve a SNAFU with tags.

6.10 0.4.0 (2022-05-03)

- Add the ability to search for tasks.
- Add json output format for tasks.
- Implement the `tasks3 task show` cli endpoint.
- Update docs.
- Add Output format preference to config.
- Make the cli interface easier to use (flatten the task command tree)

6.11 0.3.3 (2022-05-02)

- Switch docs theme to `sphinx_rtd_theme`.

6.12 0.3.2 (2022-05-02)

- Add workflow to check for package compatability with PyPI. This should make sure that the issue with v0.3.0 does not occur again.

6.13 0.3.1 (2022-05-02)

- Fix README to render on PyPI.

6.14 0.3.0 (2022-05-02)

- Remove `tasks3 db init` cli command.
- Implement `tasks3 task add` cli command.
- Implement `task.yaml`, `task.short`, `task.one_line` methods to display task.

6.15 0.2.8 (2022-05-01)

- Use dataclass to store configuration settings.
- Flatten tasks3.config module into config.py file.

6.16 0.2.7 (2022-04-30)

- Remove usage of deprecated SQLAlchemy api `db_engine.table_names`.
- Remove deprecated pytest configuration option `collect_ignore`.

6.17 0.2.6 (2022-04-30)

- Flatten tasks3.db.model module into models.py
- Linting changes
- Minor refactoring

6.18 0.2.4 (2022-04-30)

- Remove pytest from dependency and let tox handle testing.

6.19 0.2.3 (2022-04-30)

- Migrate testing to github-workflow
- Update SQLAlchemy package version.
- Switch deployment workflow to python 3.9

6.20 0.2.0 (2022-04-30)

- Drop support for python<=3.8

6.21 0.1.0 (2020-08-17)

- Implement tasks3.add
- Implement tasks3.edit
- Implement tasks3.remove

6.22 0.0.11 (2020-08-04)

- Add support for a yaml configuration file.
- Add database to store Tasks, db models and api to interact with db.
- Switch to using requirements.txt for managing dependency and add back the support for py35.
- Add a bunch of type annotations.
- **Update dependency:**
  - pip to 20.2
  - pytest to 6.0.1
  - tox to 3.18.1
  - coverage to 5.2.1

6.23 0.0.9 - 0.0.10 (2020-07-26)

- Fix version numbers and git tags.

6.24 0.0.8 (2020-07-26)

- Implement a CLI for tasks3.
- Add black (formatter).
- Add some basic test-cases.

6.25 0.0.2 - 0.0.7 (2020-07-20)

- Move deployment away from Travis to Github workflow.

6.26 0.0.1 (2020-07-20)

- First release on PyPI.

CHAPTER 7

tasks3

A commandline tool to create and manage tasks and todos.

Most task management tools are their own applications, so to manage tasks you have to perform context switching by leaving what you're working on to go to the task manager application.

`tasks3` aims to solve that by bringing your tasks to you instead.

Each task is automatically assigned to the directory it was created in and you can easily retrieve tasks under a directory.

- Free software: GNU General Public License v3
- Documentation: https://tasks3.readthedocs.io.

7.1 Features

CHAPTER 8

Help

It is easy to explore all capabilities of `tasks3` by running `tasks3 --help`. Each command also has its own help page which can be accessed by running:

```
$ tasks3 <command> --help
```

CHAPTER 9

Create Tasks

Easily create tasks from the commandline and delegate them to folders.

Create a task in a specific folder with default settings.

```
$ tasks3 add --title "Think of a cool name" \
    --folder "~/Documents/story" \
    --yes
Added Task:
[e1c100] Think of a cool name (    ) (  )
  [path: ~/Documents/story]
```

Create a task in a current folder with custom settings and description.

```
$ tasks3 add --title "Try new model" \
    --urgency 4 --importance 3 \
    --description "Try:\n - model with 3 layers.\n - model with 4 layers." \
    --yes
Added Task:
[a0a5f4] Try new model () ( )
    Try:
     - model with 3 layers.
     - model with 4 layers.
```

CHAPTER 10

Edit Existing Tasks

You can edit existing tasks with the `tasks3 edit` command.

For example: You can use `edit` to update the urgency of a task.

```
$ tasks3 edit --urgency 4 e1c100
Updated Task:
[e1c100] Think of a cool name () (  )
  [path: ~/Documents/story]
```

CHAPTER 11

Search Tasks

You can search for tasks using various filters.

You can search for tasks with a specific importance value.

```
$ tasks3 search --importance 2
[4a14d0] What is right here and now
[f79155] Think of a cool name [path: /home/<user>/Documents/project]
[2ce91b] See home [path: /home]
```

You can restrict search to a folder and its sub-directories.

```
$ tasks3 search --folder ~/Documents/project --output-format yaml
title: Think of a Cool name
urgency: 2
importance: 2
tags: null
folder: /home/<user>/Documents/project
```

You can also search for sub-strings in task title or description. It is also possible to restrict the search to tasks that have a specific set of tags. Run `tasks3 search --help` to get see a full list off options.

CHAPTER 12

Show Tasks

You can show all tasks under current directory.

```
$ tasks3 show
[a0a5f4] Try new model () ( )
    Try:
     model with 3 layers.
     model with 4 layers.
[4a14d0] What is right here and now (    ) (  )
```

You can also show a particular task by specifying its id.

```
$ tasks3 show 1d8a9a
[1d8a9a] Give a Title to this Task. (    ) ()
  (Hello tasks3)
    Task with
    multi-line
    desc
```

If you prefer to see the task in a different format, you can use the `--output-format` option.

```
$ tasks3 show --output-format json 1d8a9a
{
  "id": "1d8a9a",
  "title": "Give a Title to this Task.",
  "urgency": 2,
  "importance": 4,
  "tags": [
    "Hello tasks3"
  ],
  "folder": "/home/<user>/Documents/tasks3",
  "description": "Task with \nmulti-line \ndesc"
}
```

CHAPTER 13

Complete Tasks

You can use the `tasks3 mark <task_id>` or `tasks3 edit --done <task_id>` command to mark a task as completed.

```
$ tasks3 mark 2e0b84
[2e0b84] Adding support for task completion (    ) (  )
```

CHAPTER 14

Delete Tasks

You can use the `tasks3 delete <task_id>` command to delete a task.

Note: Deleting is a destructive action, prefer to mark the task as complete to hide it.

```
$ tasks3 remove --yes 2e0b84
Removed Task: [2e0b84] Adding support for task deletion (    ) (  )
```

CHAPTER 15

Shell Integration

tasks3 supports shell integration for bash, zsh, and fish; tasks3 will automatically run `tasks3 show -o oneline` when you `cd` into a directory to show the tasks in that directory.

You can setup shell integration by adding the following command to your `.rc` file.

```
eval "$(tasks3 shell $(basename $SHELL))"
```

Note: Pull requests to support additional shells are greatly appreciated. Please see Contributing page for information on how to contribute.

This package was created with Cookiecutter and the audreyr/cookiecutter-pypackage project template.

CHAPTER 16

Indices and tables

- genindex
- modindex
- search