santa-helpers Documentation

Release 0.0.1

Magdalena Rother

Mar 08, 2022

CONTENTS:

CHAPTER

ONE

SANTA-HELPERS

Helpers for Advent of Codee

- Free software: MIT license
- Documentation: https://santa-helpers.readthedocs.io.

1.1 Features

- Calculate manhattan distance

```
>>> distances.manhattan((-3, 1), (0, 0))

4
```

- Generate neighbors

```
>>> list(neighbors.neighbors((1, 1)))

[(1, 0), (0, 1), (2, 1), (1, 2)]

>>> list(neighbors.neighbors((1, 1), 8))

[
    (0, 0),
    (1, 0),
    (2, 0),
    (0, 1),
    (2, 1),
    (0, 2),
    (1, 2),
    (2, 2)
]

>>> list(neighbors.neighbors((1, 1), p_min=(1, 1)))
```

(continues on next page)

(continued from previous page)

```
[(2, 1), (1, 2)]
```

- Generate points in path

```
>>> list(paths.path_points((0, 0), 'R2'))

[(1, 0), (2, 0)]
```

1.2 Credits

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

CHAPTER

TWO

INSTALLATION

2.1 Stable release

To install santa-helpers, run this command in your terminal:

```
$ pip install santa_helpers
```

This is the preferred method to install santa-helpers, 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 santa-helpers can be downloaded from the Github repo.

You can either clone the public repository:

```
$ git clone git://github.com/lenarother/santa_helpers
```

Or download the tarball:

```
$ curl -OJL https://github.com/lenarother/santa_helpers/tarball/master
```

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

```
$ python setup.py install
```

CHAPTER

THREE

USAGE

To use santa-helpers in a project:

```
import santa_helpers
```

CHAPTER

FOUR

SANTA_HELPERS

4.1 santa_helpers package

4.1.1 Submodules

4.1.2 santa_helpers.distances module

santa_helpers.distances.**manhattan**(*p1: Tuple[int, ...]*, *p2: Tuple[int, ...]*) → int
Calculate Manhattan distance between two points.

4.1.3 santa_helpers.neighbors module

santa_helpers.neighbors.**is_point_in_range**(*p*, *p_min=None*, *p_max=None*) → bool
Check if point lays between two other points.

Args: p: tuple (x, y) p_min (optional): min border point p_max (optional): max border point

Returns: True if p_max >= p >= p_min

santa_helpers.neighbors.**neighbors**(*p*, *n=4*, *p_min=None*, *p_max=None*)
Point neighbor generator.

Args: p: tuple (x, y) n: int 4 (no diagonal) or 8 (with diagonal) p_min (optional): min grid point, if not given infinite p_max (optional): max grid point, if not given infinite

Yields: point (x, y)

4.1.4 santa_helpers.parse module

santa_helpers.parse.**parse_grid_to_dict**(*data: str*) → dict
Parse grid given as a string to dictionary. k: coordinates (x, y) v: value

Example: X.O => { (0, 0): 'X', (1, 0): '.', (2, 0): 'O', … (0, 1): '.', (1, 1): '.', (2, 1): '.', ..O (0, 2): '.', (1, 2): '.', (2, 2): 'O', }

4.1.5 santa_helpers.paths module

santa_helpers.paths.**get_direction**(*ch: str*) → Tuple[int, int]
Coordinates point for direction

Args: ch: str - direction as a single letter UDLR or NEWS

Returns: tuple (x, y) - direction coordinates. E.g.:

N -> (0, 1) # north S -> (0, -1) # south L -> (-1, 0) # left

Raises KeyError: if direction char not in allowed directions.

santa_helpers.paths.**get_target_point**(*start: Tuple[int, int]*, *steps: str*) → Tuple[int, int]
Coordinates of target point based on start point and steps.

Args: start: tuple (x, y) - coordinates of the starting point. steps: string e.g U15 - contains direction (U D L R or N E S W)

and number of steps.

Directions - two direction systems are possible:

Relative orientations: U - Up -> (0, 1) D - Down -> (0, -1) L - Left -> (-1, 0) R - Right -> (1, 0)

Geographical directions: N - North -> (0, 1) E - East -> (1, 0) S - South -> (0, -1) W - West -> (-1, 0)

Returns: tuple (x, y) - coordinates of the target point.

santa_helpers.paths.**path_points**(*start*, *steps*)
Generate coordinates of each path point based on start point and steps.

Args: start: tuple (x, y) - coordinates of the starting point. steps: string e.g U15 - contains direction (UDLR or NESW)

and number of steps.

Directions - two direction systems are possible:

Relative orientations: U - Up -> (0, 1) D - Down -> (0, -1) L - Left -> (-1, 0) R - Right -> (1, 0)

Geographical directions: N - North -> (0, 1) E - East -> (1, 0) S - South -> (0, -1) W - West -> (-1, 0)

Yields: tuple (x, y) - point.

4.1.6 Module contents

Top-level package for santa-helpers.

CHAPTER

FIVE

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:

5.1 Types of Contributions

5.1.1 Report Bugs

Report bugs at https://github.com/lenarother/santa_helpers/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.

5.1.2 Fix Bugs

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

5.1.3 Implement Features

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

5.1.4 Write Documentation

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

5.1.5 Submit Feedback

The best way to send feedback is to file an issue at https://github.com/lenarother/santa_helpers/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 :)

5.2 Get Started!

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

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

```
$ git clone git@github.com:your_name_here/santa_helpers.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 santa_helpers
$ cd santa_helpers/
$ python setup.py develop
```

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 santa_helpers tests
$ python setup.py test or pytest
$ 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. Submit a pull request through the GitHub website.

5.3 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 3.5, 3.6, 3.7 and 3.8, and for PyPy. Check https://travis-ci.com/lenarother/santa_helpers/pull_requests and make sure that the tests pass for all supported Python versions.

5.4 Tips

To run a subset of tests:

```
$ pytest tests.test_santa_helpers
```

5.5 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
```

Travis will then deploy to PyPI if tests pass.

CHAPTER

SIX

HISTORY

6.1 0.0.1 (2022-02-02)

- First release on PyPI.
- neighbors
- parsing to dict

6.2 0.0.2 (2022-03-08)

- manhattan distance
- points in path

CHAPTER

SEVEN

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

S

INDEX