taperable$_{h}elix Documentation$

Release 0.8.17

Wink Saville

Oct 12, 2020

CONTENTS

CHAPTER

ONE

PACKAGE DOCS 0.8.17

class taperable_helix.**HelixLocation**(*radius=None*, *horz_offset=0*, *vert_offset=0*)

radius: Optional[float] = None
radius of helix if none h.radius

horz_offset: float = 0
horizontal offset added to radius then x and y calculated

vert_offset: float = 0
vertical added to z of radius

class taperable_helix.**Helix**(*radius*, *pitch*, *height*, *taper_out_rpos=0*, *taper_in_rpos=1*, *inset_offset=0*, *first_t=0*, *last_t=1*)
This class represents a taperable Helix.

The required attributes are radius, pitch and height. Thse attributes create simple single line helix. But the primary purpose for Helix is to create a set of helical "wires" using non-zero values for taper_rpos, horz_offset and vert_offset to define solid helixes that can taper at each end to a point.

This is useful for creating internal and external threads for nuts and bolts. This is accomplished by invoking helix() multiple times with same radius, pitch, taper_rpos, inset_offset, first_t, and last_t. But with different HelixLocation radius, horz_offset and vert_offset.

Each returned function will then generate a helix defining an edge of the thread. The edges can be used to make faces and subsequently a solid of the thread. This can then be combined with the "core" objects which the threads are "attached" using a "union" operator.

radius: float
radius of the basic helix.

pitch: float
pitch of the helix per revolution. I.e the distance between the height of a single "turn" of the helix.

height: float
height of the cyclinder containing the helix.

taper_out_rpos: float = 0
taper_out_rpos is a decimal number with an inclusive range of 0..1 such that (taper_out_rpos * t_range) defines the t value where tapering out ends, it begins at t == first_t. A ValueError exception is raised if taper_out_rpos < 0 or > 1 or taper_out_rpos > taper_in_rpos. Default is 0 which is no out taper.

taper_in_rpos: float = 1
taper_in_rpos: is a decimal number with an inclusive range of 0..1 such that (taper_in_rpos * t_range) defines the t value where tapering in begins, it ends at t == last_t. A ValueError exception is raised if taper_out_rpos < 0 or > 1 or taper_out_rpos > taper_in_rpos. Default is 1 which is no in taper.

inset_offset: float = 0
inset_offset: the helix will start at z = inset_offset and will end at z = height - (2 * inset_offset). Default 0.

first_t: float = 0
first_t is the first t value passed to the returned function. Default 0

last_t: float = 1
last_t is the last t value passed to the returned function. Default 1

helix(*hl=None*)
This function returns a Function that is used to generates points on a helix.

It takes an optional HelixLocation which refines the location of the final helix when its tapered. If HelixLocation is None then the radius is Helix.radius and horz_offset and vert_offset will be 0. If its not None HelixLocation.radius maybe None, in which case Helix.radius will be used. and HelixLocation.horz_offset will be added to the radius and used to calculate x and y. The HelixLocation.vert_offset will be added to z.

This function returns a function, f. The funciton f that takes one parameter, an inclusive value between first_t and last_t. We then define t_range=last_t-first_t and the rel_height=(last_t-t)/t_range. The rel_height is the relative position along the "z-axis" which is used to calculate function functions returned tuple(x, y, z) for a point on the helix.

Credit: Adam Urbanczyk from cadquery [forum post](https://groups.google.com/g/cadquery/c/5kVRpECcxAU/m/7no7_ja6AAAJ)

Parameters **hl** (`Optional[HelixLocation]`) – Defines a refinded location when the helix is tapered

Return type `Callable[[float], Tuple[float, float, float]]`

Returns A function which is passed "t", an inclusive value between first_t and last_t and returns a 3D point (x, y, z) on the helix as a function of t.

CHAPTER

TWO

INSTALLATION

2.1 Stable release

To install taperable-helix, run this command in your terminal:

```
$ pip install taperable-helix
```

This is the preferred method to install taperable-helix, 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 Test release from testpypi

To install taperable-helix from testpypi, run this command in your terminal:

```
$ pip install --index-url https://test.pypi.org/simple/ taperable-helix
```

2.3 From sources

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

You can either clone the public repository:

```
$ git clone git://github.com/winksaville/py-taperable-helix taperable-helix
$ cd taperable-helix
```

Or download the tarball

```
$ curl -OJL https://github.com/winksaville/py-taperable-helix/releases/v0.8.17.tar.gz
```

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

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

Or if you want to install in editable mode for development:

```
$ make install-dev
```

```
$ pip install -e . -r dev-requirements.txt
```

2.4 Uninstall

```
$ pip uninstall taperable-helix
```

CHAPTER

THREE

USAGE

To use taperable_helix in a project:

```
import taperable_helix
```

CHAPTER

FOUR

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 taperable-helix 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

taperable_helix could always use more documentation, whether as part of the official taperable_helix 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 taperable-helix 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 taperable-helix for local development.

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

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

3. Instantiate an (virtual) enviorment which supports python3.7, isort, black, flake8 and bump2version. Using *make install-dev* will install appropriate development dependencies:

```
$ <instantiate your virtual environment if necessary>
$ cd taperable_helix/
$ make install-dev
```

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 are formantted correctly and pass the tests:

```
$ make format
$ make test
```

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.

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 3.7 and 3.8.

4.4 Tips

To run a particular test execute *pytest* with the test file to run followed by a *::xxx* where *xxx* is the test name. See pytest usage for more info:

```
$ pytest tests/test_taperable_helix.py::test_helix_torp_0pt1_tirp_0pt9_ho_0pt2
```

4.5 Deploying

A reminder for the maintainers on how to deploy. Make sure all your changes are committed. Then run and validate that test.pypi.org is good:

```
$ bump2version patch # param maybe: major | minor | patch
$ make push-tags
$ make release-testpypi
```

Finally, assuming test.pypi.org is good, push to pypi.org:

```
$ make release
```

CHAPTER

FIVE

CREDITS

This code originated from a post by Adam Urbanczyk to the CadQuery forum.

5.1 Development Lead

- Wink Saville <wink@saville.com>

5.2 Contributors

None yet. Why not be the first?

CHAPTER

SIX

INDICES AND TABLES

- genindex
- modindex
- search

INDEX