PyVerm Documentation

Release 0.3.0

Marius Hürzeler

Dec 12, 2019

User Guide

What is PyVerm?

PyVerm is a Python-Package for geodetic and surveying calculations. The main focus is on calculations for surveying in switzerland, but PyVerm should be as versatile as possible. In addition to its use in education and research, it should also be possible to use it as a component for software development.

PyVerm is currently in its first phase of development.

How to install PyVerm?

```
# Python 3.5 and higher or PyPy3.5
pip install pyverm
```

How to use PyVerm?

```
import pyverm

standpoint = pyverm.Point(2600000, 1200000, 0)
orientation = 123.4567

station = pyverm.station(standpoint, orientation)
new_point = station.survey(pyverm.ObservationPolar(
                                reduced_horizontal_angle=375.00,
                                reduced_distance=575.1234
                                )
                           )

azimuth = pyverm.azimuth(standpoint, new_point)
```

CHAPTER 1

User Guide

1.1 Get Started

1.1.1 Installation

You can install pyverm easily over PIP.

```
pip install pyverm
```

1.1.2 Code Examples

Distance and Azimuth

```
import pyverm

point_1 = pyverm.Point(2600123, 1200456, 0)
point_2 = pyverm.Point(2600789, 1200123, 0)

distance = pyverm.distance(point_1, point_2)

azimuth = pyverm.azimuth(point_1, point_2)
```

Polar Stakeout

```
import pyverm

standpoint = pyverm.Point(2600000, 1200000, 0)
orientation = 123.4567
```

(continues on next page)

(continued from previous page)

```
station = pyverm.station(standpoint, orientation)
observation = station.stakeout(pyverm.Point(2600010, 1200020, 0))
```

Free Station

```
import pyverm

observations = [
    pyverm.ObservationPolar(
        reduced_targetpoint=(2600100, 1200100),
        reduced_horizontal_angle=0,
        reduced_distance=141.421356237),
    pyverm.ObservationPolar(
        reduced_targetpoint=(2600000, 1199800),
        reduced_horizontal_angle=150,
        reduced_distance=200),
    pyverm.ObservationPolar(
        reduced_targetpoint=(2599900, 1200100),
        reduced_horizontal_angle=300,
        reduced_distance=141.421356237)
]

station = pyverm.station_helmert(observations)

standpoint = station.standpoint
orientation = station.orientation

new_point = station.survey(pyverm.ObservationPolar(
                                reduced_horizontal_angle=375.00,
                                reduced_distance=575.1234
                                )
                           )
```

1.2 Documentation

PyVerm is a Python-Package for geodetic and surveying calculations. The main focus is on calculations for surveying in switzerland, but PyVerm should be as versatile as possible. In addition to its use in education and research, it should also be possible to use it as a component for software development.

class pyverm.**ObservationPolar**(***kwargs*)

__init__(***kwargs*)

Represent a polar observation with all associated values as simple and usable as possible.

Despite all attributes are optional, depending on the function certain attributes must be present.

Todo:

- Document the reduction of the raw values
- implement the reduction of the distance
- add unittest for this class

Parameters

- **reduced_targetpoint** (*tuple or pyverm.Point*) – (optional) Point which was measured with this observation
- **reduced_horizontal_angle** (*float or decimal*) – (optional) horizontal angle in gon with all corrections
- **reduced_zenith_angle** (*float or decimal*) – (optional) zenith angle in gon with all corrections
- **reduced_distance** (*float or decimal*) – (optional) distance in meters with all corrections
- **raw_horizontal_angle** (*float or decimal*) – (optional) horizontal angle in gon
- **raw_horizontal_angle_2** (*float or decimal*) – (optional) horizontal angle in gon in second direction
- **raw_zenith_angle** (*float or decimal*) – (optional) zenith angle in gon
- **raw_zenith_angle_2** (*float or decimal*) – (optional) zenith angle in gon in second direction
- **raw_distance** (*float or decimal*) – (optional) distance in meters **not yet implemented**
- **raw_distance_2** (*float or decimal*) – (optional) distance in meters in second direction **not yet implemented**

reduced_horizontal_angle
Return reduced_horizontal_angle or if None and raw in two direction present, return calculated reduced angle :return:

reduced_zenith_angle
Return reduced_zenith_angle or if None and raw in two direction present, return calculated reduced angle :return:

class pyverm.**Station**(*standpoint*, *orientation*)

stakeout(*point*)
Return the observation values, which are needed to stakeout the given point.

Parameters **point** (*tuple or pyverm.Point*) – point to stakeout

Returns *ObservationPolar* object

Return type *pyverm.ObservationPolar*

survey(*observation*)
Returns the Point, which was surveyed with the given observation.

Parameters **observation** (pyverm.ObservationPolar) –

Returns *Point* object

Return type *pyverm.Point*

class pyverm.**Point**(*y*, *x*, *z*)

x
Alias for field number 1

y
Alias for field number 0

z
Alias for field number 2

pyverm.**azimuth**(*point_a*, *point_b*)
Return the azimuth form point A to point B.

The azimuth is the clockwise angle from the north (x axis) and the connecting line from point a to point b. It is calculated with the following formula:

$$azimuth = \arctan 2(\Delta y / \Delta x)$$

Parameters

- **point_a** (*tuple or pyverm.Point*) – Point A
- **point_b** (*tuple or pyverm.Point*) – Point B

Returns azimuth in gon

Return type Decimal

pyverm.**distance**(*point_a*, *point_b*)
Return the 2D distance from point A to Point B.

$$distance = \sqrt{\Delta y^2 + \Delta x^2}$$

Parameters

- **point_a** (*tuple or pyverm.Point*) – Point A
- **point_b** (*tuple or pyverm.Point*) – Point B

Returns distance in meters

Return type Decimal

pyverm.**station**(*standpoint*, *orientation*)
Return a station with standpoint and orientation.

Parameters

- **standpoint** (*tuple or pyverm.Point*) – standpoint
- **orientation** (*int or decimal*) – azimuth of null direction

Returns *Station* object

Return type *pyverm.Station*

pyverm.**station_abriss**(*standpoint*, *observations*)
Calculate the orientation from the observations and return the station object.

$$abriss = \frac{\sum_{i=1}^{n}(azimuth_{standpoint_i \rightarrow targetpoint_i} - horizontal\ angle_i)}{n}$$

Parameters

- **standpoint** (*tuple or pyverm.Point*) – standpoint
- **observations** (*list or tuple with pyverm.ObservationPolar*) – a list or a tuple containing the Observations

Returns *Station* object

Return type *pyverm.Station*

pyverm.**station_helmert**(*observations*)
Calculate the standpoint and orientation and return the station object.

The station is calculated locally an then transformed in the coordinate system truth a helmert transformation. The orientation gets determined over an abriss.

Parameters **observations** (*list or tuple with pyverm.ObservationPolar*) – a list or a tuple containing the Observations

Returns *Station* object

Return type *pyverm.Station*

pyverm.**transformation_helmert**(*sourcepoints*, *destinationpoints*)
Calculates the transformation and returns a Transformation object.

Parameters

- **sourcepoints** (*list or tuple with pyverm.Point*) – the source points for the transformation
- **destinationpoints** (*list or tuple with pyverm.Point*) – the destination points for the transformation

Returns Transformation object

Return type *pyverm.Station*

CHAPTER 2

General Information's

2.1 FAQ

ToDo: ask me questions

Links

- **Documentation** pyverm.readthedocs.io
- **Sourcecode** github.com/doppelmeter/pyverm
- **PyPi** pypi.org/project/pyverm
- **License** GNU GPLv3

2.2 Road Map

2.2.1 v0.2.0

- more functions
- better structure

2.2.2 v0.5.0

- big refactoring

2.2.3 v1.0.0

- additional german API
- some users ;)

2.2.4 after v1.0.0

- GUI
- QGIS-Pluging
- or something completely different

2.3 Release Notes

PyVerm uses Semantic Versioning

2.3.1 0.3.0 (12.12.2019)

- add pyverm.geocom

2.3.2 0.1.3 (25.12.2018)

- test deployment with travis CI

2.3.3 0.1.2 (25.12.2018)

- trying to understand git

2.3.4 0.1.1 (24.12.2018)

- adding support for PyPY
- adding more unittests
- adding Travis CI

2.3.5 0.1.0 (22.12.2018)

- big refactoring
- first “stable” API

2.3.6 0.0.3 (15.12.2018)

- first solution
- first upload to PyPi

2.3.7 0.0.0 (06.12.2018)

- Idea

2.4 License

PyVerm is licensed under the GNU General Public License version 3.

CHAPTER 3

Developers Guide

3.1 Development

3.1.1 How to contribute?

Bug Reports

If you find a bug, please open an bug report on GitHub.

Ideas

If you have a brilliant idea for a new feature, please open an feature request GitHub.

Code

In the current development phase, I prefer to write the code myself. But for special things you can contact me via GitHub.

3.2 How to git?

3.2.1 Make a new Feature

- create a feature branch, by checking out development branch to the feature branch

  ```
  git checkout -b myfeature development
  ```

- make your developements

- when finished, merge features branch into development branch, by checking out development branche

  `git checkout development`

  mergin feature branch to development branch **Important:** `--no-ff`

  `git merge --no-ff myfeature`

  delet your features branch

  `git branch -d myfeature`

  push development branch to origin (GitHub)

  `git push origin development`

3.2.2 Make a Release

- check out release branch from development branch -> name = release-X.X.X

  `git checkout -b release-X.X.X development`
- increasing version number and commit this

  `git commit -a -m "version number changed for release"`
- make other stuff for release preparation and commit it
- to finish the release, check out the master branch

  `git checkout master`

  merge release branch to master branch

  `git merge --no-ff release-1.2`

  make a tag

  `git tag -a v1.2.1`

  push released master to origin (GitHub) with all tags

  `git push origin master`

  and merge release with development branch

  `git checkout development`

  merge release branch to master branch

  `git merge --no-ff release-1.2`

  push released master to origin (GitHub) with all tags

  `git push origin development`
- finally delete release branch

  `git branch -d release-1.2`

3.2.3 Hot Fix Branch

https://nvie.com/posts/a-successful-git-branching-model/

3.3 Translations for Surveying-Words

English	German
station	Station
stationing	Stationierung
distance	Distanze
station height	Stationshöhe
target height	Zielhöhe
horizontal angle	Horizontalwinkel
vertical angle	Vertikalwinkel
zenith angle	Zenitwinkel
horizontal angle 2	Horizontalwinkel (2. Lage)
vertical angle 2	Vertikalwinkel (2. Lage)
zenith angle 2	Zenitwinkel (2. Lage)
azimuth	Azimut
slope distance	Schrägdistanz
slope distance 2	Schrägdistanz (2. Lage)
horizontal distance	Horizontaldistanz
vertical distance	Vertikaldistanz
offset in out	Längsverschiebung
offset left right	Querverschiebung
station id	Standpunktnummer
target id	Zielpunktnummer
station point	Standpunkt
target point	Zielpunkt

3.4 To-Do

Todo:

- Document the reduction of the raw values
- implement the reduction of the distance
- add unittest for this class

(The original entry is located in /home/docs/checkouts/readthedocs.org/user_builds/pyverm/checkouts/latest/pyverm/_classes.py:docstrin
of pyverm.ObservationPolar.__init__, line 5.)

Python Module Index

p

Index