jicgeometry Documentation

Release 0.6.0

Tjelvar Olsson and Matthew Hartley

September 23, 2015

Contents

CHAPTER 1

JIC Geometry

Python package for basic geometry operations.

- Documentation: http://jicgeometry.readthedocs.org/en/latest/
- GitHub: https://github.com/JIC-CSB/jicgeometry
- PyPI: https://pypi.python.org/pypi/jicgeometry
- Free software: MIT License

1.1 Features

- Lightweight: no dependencies outside Python's standard library
- Cross-platform: Linux, Mac and Windows are all supported
- Works with with Python 2.7, 3.2, 3.3, and 3.4

1.2 Quick Guide

To install `jicgeometry`:

```
sudo pip install jicgeometry
```

Create some points:

```
>>> from jicgeometry import Point2D
>>> p1 = Point2D(6, 1)
>>> p2 = Point2D(3, 5)
```

Find the distances between two points:

```
>>> p1.distance(p2)
5.0
```

Add/subtract points from each other:

```
>>> p1 + p2
<Point2D(x=9, y=6, dtype=int)>
```

Scale points using multiplication/division:

```
>>> p1 / 2.0
<Point2D(x=3.00, y=0.50, dtype=float)>
```

Treat points as vectors:

```
>>> p1.unit_vector
<Point2D(x=0.99, y=0.16, dtype=float)>
>>> round(p1.magnitude, 4)
6.0828
```

1.3 History

1.3.1 0.6.0

- Added Point3D class

1.3.2 0.5.0

- Initial upload to PyPi

CHAPTER 2

API documentation

2.1 jicgeometry

Module for geometric operations.

The module contains two classes to perform geometric operations in 2D and 3D space:

- *jicgeometry.Point2D*
- *jicgeometry.Point3D*

A 2D point can be generated using a pair of x, y coordinates.

```
>>> p1 = Point2D(3, 0)
```

Alternatively, a 2D point can be created from a sequence.

```
>>> l = [0, 4]
>>> p2 = Point2D(l)
```

The x and y coordinates can be accessed as properties or by their index.

```
>>> p1.x
3
>>> p1[0]
3
```

Addition and subtraction result in vector arithmetic.

```
>>> p1 + p2
<Point2D(x=3, y=4, dtype=int)>
>>> p1 - p2
<Point2D(x=3, y=-4, dtype=int)>
```

Scalar multiplication is supported.

```
>>> (p1 + p2) * 2
<Point2D(x=6, y=8, dtype=int)>
```

Scalar division uses true division and as a result always returns a 2D point of `dtype` `float`.

```
>>> p1 / 2
<Point2D(x=1.50, y=0.00, dtype=float)>
```

It is possible to calculate the distance between two points.

```
>>> p1.distance(p2)
5.0
```

Points can also be treated as vectors.

```
>>> p3 = p1 + p2
>>> p3.unit_vector
<Point2D(x=0.60, y=0.80, dtype=float)>
>>> p3.magnitude
5.0
```

class jicgeometry.**Point2D**(*a1*, *a2=None*)
Class representing a point in 2D space.

astuple()
Return the x, y coordinates as a tuple.

astype(*dtype*)
Return a point of the specified dtype.

distance(*other*)
Return distance to the other point.

dtype
Return the type of the x, y coordinates as a string.

magnitude
Return the magnitude when treating the point as a vector.

unit_vector
Return the unit vector.

class jicgeometry.**Point3D**(*a1*, *a2=None*, *a3=None*)
Class representing a point in 3D space.

astuple()
Return the x, y coordinates as a tuple.

astype(*dtype*)
Return a point of the specified dtype.

distance(*other*)
Return distance to the other point.

dtype
Return the type of the x, y coordinates as a string.

magnitude
Return the magnitude when treating the point as a vector.

unit_vector
Return the unit vector.

j

A

D

J

M

P

U