trianglelib Documentation

Release 1.0

AJ

Jul 09, 2017

Contents

CHAPTER 1

Example: tutorial.rst — The trianglelib tutorial

"There is no royal road to geometry." — Euclid

This module makes triangle processing fun! The beginner will enjoy how the `utils` module lets you get started quickly.

```
>>> from trianglelib import utils
>>> utils.is_isosceles(5, 5, 7)
True
```

But fancier programmers can use the `Triangle` class to create an actual triangle *object* upon which they can then perform lots of operations. For example, consider this Python program:

```
from trianglelib.shape import Triangle
t = Triangle(5, 5, 5)
print 'Equilateral?', t.is_equilateral()
print 'Isosceles?', t.is_isosceles()
```

Since methods like `is_equilateral()` return Boolean values, this program will produce the following output:

```
Equilateral? True
Isosceles? True
```

Read the *Example: guide.rst — The trianglelib guide* to learn more!

Warning: This module only handles three-sided polygons; five-sided figures are right out.

CHAPTER 2

Example: guide.rst — The trianglelib guide

Whether you need to test the properties of triangles, or learn their dimensions, *trianglelib* does it all!

Special triangles

There are two special kinds of triangle for which *trianglelib* offers special support.

Equilateral triangle All three sides are of equal length.

Isosceles triangle Has at least two sides that are of equal length.

These are supported both by simple methods that are available in the *utils* module, and also by a pair of methods of the main *Triangle* class itself.

Triangle dimensions

The library can compute triangle perimeter, area, and can also compare two triangles for equality. Note that it does not matter which side you start with, so long as two triangles have the same three sides in the same order!

```
>>> from trianglelib.shape import Triangle
>>> t1 = Triangle(3, 4, 5)
>>> t2 = Triangle(4, 5, 3)
>>> t3 = Triangle(3, 4, 6)
>>> print t1 == t2
True
>>> print t1 == t3
False
>>> print t1.area()
6.0
>>> print t1.scale(2.0).area()
24.0
```

Valid triangles

Many combinations of three numbers cannot be the sides of a triangle. Even if all three numbers are positive instead of negative or zero, one of the numbers can still be so large that the shorter two sides could not actually meet to make a closed figure. If c is the longest side, then a triangle is only possible if:

$$a + b > c$$

While the documentation for each function in the *trianglelib.utils* module simply specifies a return value for cases that are not real triangles, the Triangle class is more strict and raises an exception if your sides lengths are not appropriate:

```
>>> from trianglelib.shape import Triangle
>>> Triangle(1, 1, 3)
Traceback (most recent call last):
  ...
ValueError: one side is too long to make a triangle
```

If you are not sanitizing your user input to verify that the three side lengths they are giving you are safe, then be prepared to trap this exception and report the error to your user.

CHAPTER 3

The trianglelib API reference

Routines for working with triangles.

The two modules inside of this package are packed with useful features for the programmer who needs to support triangles:

shape This module provides a full-fledged *Triangle* object that can be instantiated and then asked to provide all sorts of information about its properties.

utils For the programmer in a hurry, this module offers quick functions that take as arguments the three side lengths of a triangle, and perform a quick computation without the programmer having to make the extra step of creating an object.

advanced Some advanced stuff.

The "shape" module

Use the triangle class to represent triangles.

class `trianglelib.shape.`**`Triangle`**(*a*, *b*, *c*)
A triangle is a three-sided polygon.

`area`()
Return the area of this triangle.

`is_equilateral`()
Return whether this triangle is equilateral.

Returns bool

If the triangle is equilateral.

`is_isosceles`()
Return whether this triangle is isoceles.

`is_similar`(*triangle*)
Return whether this triangle is similar to another triangle.

perimeter()
Return the perimeter of this triangle.

scale(*factor*)
Return a new triangle, *factor* times the size of this one.

The "utils" module

Routines to test triangle properties without explicit instantiation.

trianglelib.utils.**compute_area**(*a*, *b*, *c*)
Return the area of the triangle with side lengths *a*, *b*, and *c*.

If the three lengths provided cannot be the sides of a triangle, then the area 0 is returned.

trianglelib.utils.**compute_perimeter**(*a*, *b*, *c*)
Return the perimeer of the triangle with side lengths *a*, *b*, and *c*.

If the three lengths provided cannot be the sides of a triangle, then the perimeter 0 is returned.

trianglelib.utils.**is_equilateral**(*a*, *b*, *c*)
Return whether lengths *a*, *b*, and *c* are an equilateral triangle.

trianglelib.utils.**is_isosceles**(*a*, *b*, *c*)
Return whether lengths *a*, *b*, and *c* are an isosceles triangle.

trianglelib.utils.**is_triangle**(*a*, *b*, *c*)
Return whether lengths *a*, *b*, *c* can be the sides of a triangle.

Parameters **a** : float

b : float

c : float

Returns bool

If the sides can form a triangle

The "advaned" module

trianglelib.advanced.**foo**(*a*)
A simple optimization problem.

Parameters **a** : float

The bound on the `x` values

Returns numpy.array

The value of x

This is a Link

Python Module Index

t

Index