trianglelib

Release 1.1

Brandon Rhodes, Lei Mao

Aug 01, 2020

CONTENTS:

CHAPTER

ONE

EXAMPLE: GUIDE.RST — THE TRIANGLELIB GUIDE

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

1.1 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 *trianglelib.utils* module, and also by a pair of methods of the main *Triangle* class itself.

1.2 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
```

1.3 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 *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

TWO

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

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

CHAPTER

THREE

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.

3.1 The "shape" module

Use the triangle class to represent triangles.

class `trianglelib.shape.`**`Triangle`**(*a*, *b*, *c*)

A *`Triangle`* object is a three-sided polygon.

You instantiate a *`Triangle`* by providing exactly three lengths `a`, `b`, and `c`.

They can either be intergers or floating-point numbers, and should be listed clockwise around the triangle.

If the three lengths *cannot* make a valid triangle, then `ValueError` will be raised instead.

```
>>> from trianglelib.shape import Triangle
>>> t = Triangle(3, 4, 5)
>>> print(t.is_equilateral())
False
>>> print(t.area())
6.0
```

Triangles support the following attributes, operators, and methods.

a
b
c

The three side lengths provided during instantiation.

triangle1 == triangle2

Returns true if the two triangles have sides of the same lengths, in the same order. Note that it is okay if the two triangles happen to start their list of sides at a different corner; `3,4,5` is the same triangle as `4,5,3` but neither of these are the same triangle as their mirror image `5,4,3`.

__init__(*a*, *b*, *c*)
Create a *Triangle* object with sides of lengths *a*, *b*, and *c*.

Raises *ValueError* if the three length values provided cannot actually form a triangle.

Parameters

- **a** (float) – side length one
- **b** (float) – side length two
- **c** (float) – side length three

Raises

- **ValueError** – side lengths must all be positive
- **ValueError** – one side is too long to make a triangle

is_equivalent(*triangle*)
Return whether this triangle equals another triangle.

Parameters **triangle** (*Triangle*) – another *Triangle* object

Returns whether the two *Triangle* objects are equivalent

Return type bool

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

Parameters **triangle** (*Triangle*) – another *Triangle* object

Returns whether the two *Triangle* objects are similar

Return type bool

is_equilateral()
Return whether this *Triangle* object is equilateral.

Returns whether the *Triangle* object is equilateral

Return type bool

is_isosceles()
Return whether this *Triangle* object is isosceles.

Returns whether the *Triangle* object is isosceles

Return type bool

perimeter()
Return the perimeter of this *Triangle* object.

Returns the perimeter of the *Triangle* object.

Return type float

area()
Return the area of this *Triangle* object.

Returns the area of the *Triangle* object.

Return type float

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

Parameters **factor** (float) – scaling factor

Returns a scaled new *Triangle* object

Return type *Triangle*

3.2 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.

Parameters

- **a** (float) – side length one
- **b** (float) – side length two
- **c** (float) – side length three

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

Return type float

trianglelib.utils.**compute_perimeter**(*a*, *b*, *c*)
Return the perimeter 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.

Parameters

- **a** (float) – side length one
- **b** (float) – side length two
- **c** (float) – side length three

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

Return type float

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

Parameters

- **a** (float) – side length one
- **b** (float) – side length two
- **c** (float) – side length three

Returns whether lengths *a*, *b*, and *c* are an equilateral triangle

Return type bool

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

Parameters

- **a** (float) – side length one

- **b** (float) – side length two
- **c** (float) – side length three

Returns whether lengths *a*, *b*, and *c* are an isosceles triangle

Return type bool

trianglelib.utils.**is_triangle**(*a*, *b*, *c*)

Return whether lengths *a*, *b*, *c* can be the sides of a triangle.

Parameters

- **a** (float) – side length one
- **b** (float) – side length two
- **c** (float) – side length three

Returns whether lengths *a*, *b*, *c* can be the sides of a triangle

Return type bool

CHAPTER

FOUR

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

t

INDEX

Symbols

A

B

C

E

I

M

P

S

T