triangleLib Documentation

Release 1.0

Uwe Hernandez Acosta

May 30, 2018

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:

$$a + b > c\partial_x\psi(x)$$

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.

1.4 The subMod

This sub module is for testing the includings of several sub docs

CHAPTER

TWO

THE TRIANGLELIB API REFERENCE

2.1 The "shape" module

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

Methods

<Here is the description of instantiation.>

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

is_equilateral()
Return whether this triangle is equilateral.

is_isosceles()
Return whether this triangle is isoceles.

perimeter()
Return the perimeter of this triangle.

area()
Return the area of this triangle.

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

2.2 The "utils" module

Routines to test triangle properties without explicit instantiation.

triangelib.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.

triangelib.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.

triangelib.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.

CHAPTER

THREE

THE "SUBMOD" MODULE

This sub module is for testing the includings of several sub docs

3.1 The 'subFunc' Module

This is the docstring for the example.py module. Modules names should have short, all-lowercase names. The module name may have underscores if this improves readability.

Every module should have a docstring at the very top of the file. The module's docstring may extend over multiple lines. If your docstring does extend over multiple lines, the closing three quotation marks must be on a line by itself, preferably preceded by a blank line.

`trianglelib.subMod.subFunc.`**`foo`**(*var1*, *var2*, *long_var_name='hi'*)

A one-line summary that does not use variable names or the function name.

Several sentences providing an extended description. Refer to variables using back-ticks, e.g. *var*.

Parameters **var1** : array_like

Array_like means all those objects – lists, nested lists, etc. – that can be converted to an array. We can also refer to variables like *var1*.

var2 : int

The type above can either refer to an actual Python type (e.g. `int`), or describe the type of the variable in more detail, e.g. `(N,) ndarray` or `array_like`.

long_var_name : {'hi', 'ho'}, optional

Choices in brackets, default first when optional.

Returns type

Explanation of anonymous return value of type `type`.

describe : type

Explanation of return value named *describe*.

out : type

Explanation of *out*.

type_without_description

Other Parameters **only_seldom_used_keywords** : type

Explanation

common_parameters_listed_above : type

Explanation

Raises **BadException**

Because you shouldn't have done that.

See also:

`otherfunc` relationship (optional)

`newfunc` Relationship (optional), which could be fairly long, in which case the line wraps here.

`thirdfunc`, `fourthfunc`, `fifthfunc`

Notes

Notes about the implementation algorithm (if needed).

This can have multiple paragraphs.

You may include some math:

$$X(e^{j\omega}) = x(n)e^{-j\omega n}$$

And even use a greek symbol like $omega$ inline.

References

Cite the relevant literature, e.g. *[R11]*. You may also cite these references in the notes section above.

[R11]

Examples

These are written in doctest format, and should illustrate how to use the function.

```
>>> a = [1, 2, 3]
>>> print [x + 3 for x in a]
[4, 5, 6]
>>> print "a\n\nb"
a
b
```

BIBLIOGRAPHY

[R11] O. McNoleg, "The integration of GIS, remote sensing, expert systems and adaptive co-kriging for environmental habitat modelling of the Highland Haggis using object-oriented, fuzzy-logic and neural-network techniques," Computers & Geosciences, vol. 22, pp. 585-588, 1996.

PYTHON MODULE INDEX

t

INDEX