Quadrilaterals

Release 1.0

Sadie

Nov 11, 2020

CONTENTS:

1 Configuring Sphinx from scratch **1**

1.1 Context 1

1.2 Quick links 3

2 API **7**

2.1 quadrilaterals.quadrilateral.Quadrilateral 7

2.2 quadrilaterals.kites.kite.Kite 8

2.3 quadrilaterals.kites.rhombi.rhombus.Rhombus 9

2.4 quadrilaterals.parallelograms.parallelogram.Parallelogram 9

2.5 quadrilaterals.parallelograms.rectangles.rectangle.Rectangle 10

2.6 quadrilaterals.parallelograms.rectangles.square.Square 10

2.7 quadrilaterals.trapezia.trapezium.Trapezium 11

2.8 quadrilaterals.trapezia.isosceles_trapezia.isosceles_trapezium.IsoscelesTrapezium 12

3 Indices and tables **13**

Index **15**

CHAPTER

ONE

CONFIGURING SPHINX FROM SCRATCH

Version 1.0

A software demonstration by Sadie Bartholomew of the Sphinx documentation generator.

Note: I am *not associated with the Sphinx project at all*. I have chosen to give a demo of Sphinx because it is a tool I have found extremely useful in my work in RSE roles.

1.1 Context

1.1.1 Demo format

This is a demonstration so will be largely conducted in the terminal. However, notes are provided, as incoporated into the example documentation itself.

Tip: The source and built documentation, which includes these notes, is (and will permanently remain) hosted on GitHub at: https://github.com/sadielbartholomew/sphinx-from-scratch

1.1.2 The Dummy Project

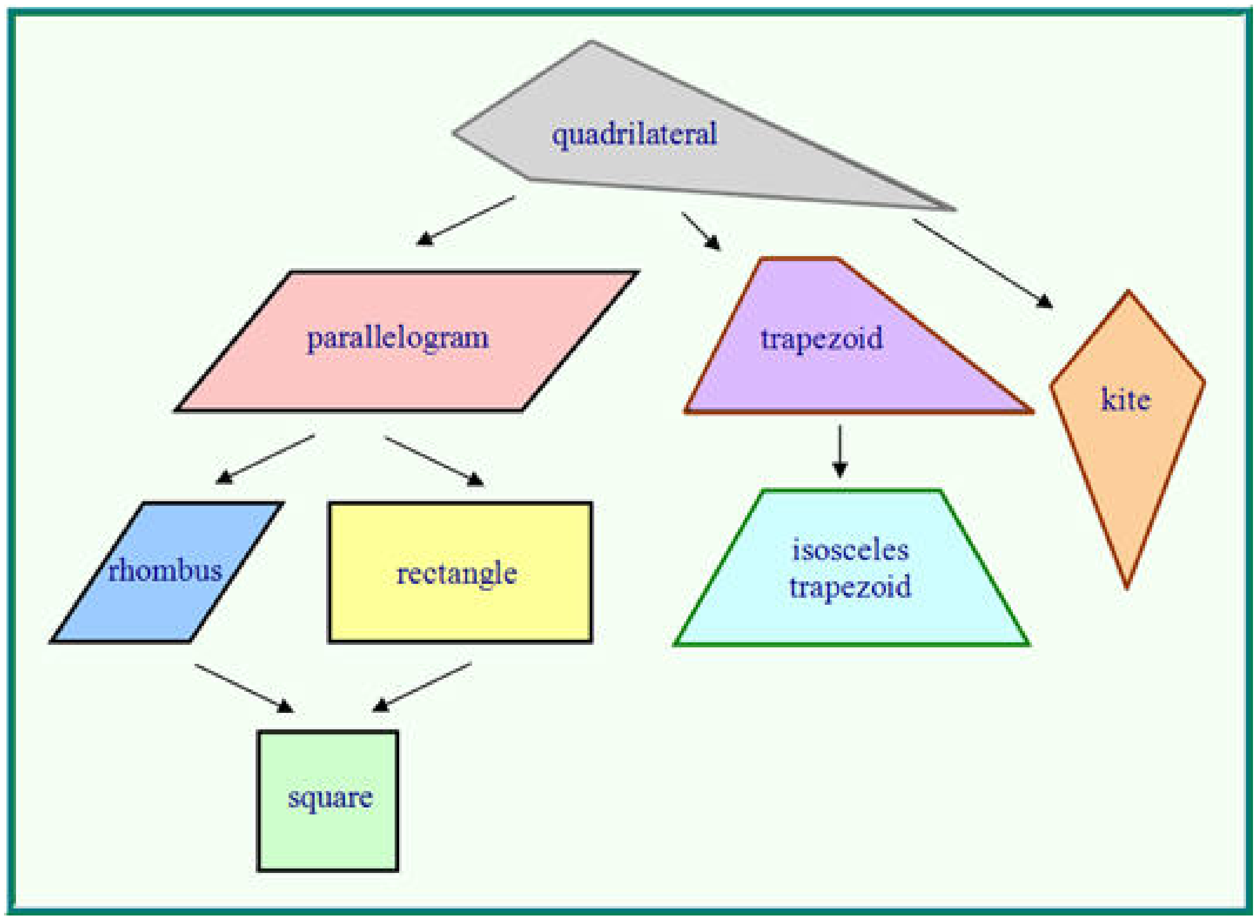
`quadrilaterals`: a very simple and trivial object-oriented Python codebase used as a placeholder for a real-life and very likely more complex project.

The codebase models categories of two-dimensional four-sided shape, which are collectively called quadrilaterals. For each category, such as a square or a rhombus, there are methods to calculate the area, perimeter and number of axes of symmetry.

A diagram[1] showing the main classes in the dummy project, and their inheritance hierarchy[2] :

[1] Image sourced from this webpage.

[2] Strictly this diagram does not capture one other relationship between the quadrilaterals which also exists, and which the dummy project incorporates into the class hierarchy, namely that a *rhombus* is a special form of *kite*.



An example of the code in use (Python console notation):

```
>>> import quadrilaterals
>>> square = quadrilaterals.Square(5)
>>> square.area()
25
>>> square.perimeter()
20
>>> help(square)
Help on Square in module quadrilaterals.parallelograms.rectangles.square object:

class Square(quadrilaterals.parallelograms.rectangles.rectangle.Rectangle,␣
↪quadrilaterals.kites.rhombi.rhombus.Rhombus)
 |  Square(side_length)
 |
 |  Base class common to all squares.
 |
 |  >>> import quadrilaterals
 |  >>> q = quadrilaterals.Square(0.2)
 |
 |  Method resolution order:
 |      Square
 |      quadrilaterals.parallelograms.rectangles.rectangle.Rectangle
 |      quadrilaterals.kites.rhombi.rhombus.Rhombus
 |      quadrilaterals.kites.kite.Kite
 |      quadrilaterals.parallelograms.parallelogram.Parallelogram
```

(continues on next page)

(continued from previous page)

```
 |      quadrilaterals.quadrilateral.Quadrilateral
 |      builtins.object
 |
...
...
...
>>> parallelogram = quadrilaterals.Parallelogram(5, 2, 1.5)
>>> help(parallelogram)
Help on Parallelogram in module quadrilaterals.parallelograms.parallelogram object:

class Parallelogram(quadrilaterals.quadrilateral.Quadrilateral)
 |  Parallelogram(parallel_side_A_length, parallel_side_B_length, any_angle=None)
 |
 |  Base class common to all parallelograms.
 |
 |  >>> import math
 |  >>> q1 = quadrilaterals.Parallelogram(2, 5, 1.1)
 |  >>> q2 = quadrilaterals.Parallelogram(2, 5, math.pi - 1.1)
 |
 |  Method resolution order:
 |      Parallelogram
 |      quadrilaterals.quadrilateral.Quadrilateral
 |      builtins.object
 |
...
...
...
>>> parallelogram.area()
9.974949866040545
>>> parallelogram.perimeter()
14
```

1.2 Quick links

1.2.1 Examples of Sphinx-generated documentation

It is hard to tell what has been made with Sphinx without looking at the codebase source, but often if you scroll to the bottom of some documentation you can often see a "Created using Sphinx <sphinx version>" note in the footer. A small number of examples are referenced below.

- A large but absolutely not comprehensive listing collected by the Sphinx team: https://www.sphinx-doc.org/en/master/examples.html
- Sphinx's own documentation (made with Sphinx, of course!): https://www.sphinx-doc.org/en/master/
- Python 3 documentation: https://docs.python.org/3/
- Python 2 documentation: https://docs.python.org/3/
- Tornado documentation: https://www.tornadoweb.org/en/stable/
- Official RST documentation (note it is also made in Sphinx): https://docutils.readthedocs.io/en/sphinx-docs/user/rst/quickstart.html
- Dask documentation: https://docs.dask.org/en/latest/
- JupyterHub documentation: https://jupyterhub.readthedocs.io/en/stable/

- NumPy documentation: https://numpy.org/doc/stable/reference/
- A small (slightly old) listing of non-Python projects with Sphinx documentation: https://ericholscher.com/blog/2014/feb/11/sphinx-isnt-just-for-python/
- An example of a book made with Sphinx: https://www.theoretical-physics.net/

1.2.2 Sphinx

Basic

- Sphinx documentation homepage: https://www.sphinx-doc.org/en/master/
- Quick start including *sphinx-quickstart* command: https://www.sphinx-doc.org/en/master/usage/quickstart.html
- HTML themes: https://www.sphinx-doc.org/en/master/usage/theming.html

Advanced

- Guidance on extensions: https://www.sphinx-doc.org/en/master/usage/extensions/index.html
- An "awesome" listing of extra Sphinx resources: https://github.com/yoloseem/awesome-sphinxdoc
- Big listing of even more HTML themes: https://sphinx-themes.org/
- Our customisation of the "alabaster" default theme, in practice: https://ncas-cms.github.io/cf-python/
- Example Sphinx extension project repositories:
  - `sphinx-copybutton` (button to copy code in code examples): https://github.com/executablebooks/sphinx-copybutton
  - `sphinx-toggleprompt` (button to hide prompts and outputs in console-like code examples): https://github.com/jurasofish/sphinx-toggleprompt
  - `sphinx-pyreverse` (generates UML diagramms of the code or parts of it into the documentation): https://github.com/alendit/sphinx-pyreverse

reStructuredText (*.rst* extension) format

- Official reStructuredText documentation (note it is also made in Sphinx): https://docutils.readthedocs.io/en/sphinx-docs/user/rst/quickstart.html
- Helpful cheatseets:
  - https://docutils.sourceforge.io/docs/user/rst/quickref.html
  - https://github.com/ralsina/rst-cheatsheet/blob/master/rst-cheatsheet.rst

Documenting projects with Sphinx

- A nice blog post: https://medium.com/@richdayandnight/a-simple-tutorial-on-how-to-document-your-python-project-using-sphi
- A great post showing how to use Sphinx with other tools to document a C++ project: https://devblogs.microsoft.com/cppblog/clear-functional-c-documentation-with-sphinx-breathe-doxygen-cmake/

CHAPTER TWO

API

`quadrilateral.Quadrilateral`(side_A_length, . . .)	Base class common to all quadrilaterals, four-sided polygons.
`kites.kite.Kite`(side_A_length, side_B_length)	Base class common to all kites.
`kites.rhombi.rhombus.Rhombus`(side_length[, . . .])	Base class common to all rhombi.
`parallelograms.parallelogram.Parallelogram`(. . .)	Base class common to all parallelograms.
`parallelograms.rectangles.rectangle.Rectangle`(. . .)	Base class common to all rectangles.
`parallelograms.rectangles.square.Square`(. . .)	Base class common to all squares.
`trapezia.trapezium.Trapezium`(. . . [, height])	Base class common to all trapezia.
`trapezia.isosceles_trapezia.isosceles_trapezium.IsoscelesTrapezium`(. . .)	Base class common to all isosceles trapezia.

2.1 quadrilaterals.quadrilateral.Quadrilateral

class quadrilaterals.quadrilateral.**Quadrilateral**(*side_A_length*, *side_B_length*, *side_C_length*, *side_D_length*)

Base class common to all quadrilaterals, four-sided polygons.

```
>>> q = quadrilaterals.Quadrilateral(1, 2, 3, 4)
```

__init__(*side_A_length*, *side_B_length*, *side_C_length*, *side_D_length*)

Initialize self. See help(type(self)) for accurate signature.

Methods

`__init__`(side_A_length, side_B_length, . . .)	Initialize self.
`area()`	Return the area of the polygon.
`axes_of_symmetry()`	Return the number of axes of symmetry of the quadrilateral.
`perimeter()`	Return the perimeter of the quadrilateral.

Attributes

`dimensions`
`sides`

2.2 quadrilaterals.kites.kite.Kite

class `quadrilaterals.kites.kite.`**`Kite`**(*side_A_length*, *side_B_length*, *angle_AB=None*)

Base class common to all kites.

```
>>> q = quadrilaterals.Kite(3, 5, 2.4)
```

`__init__`(*side_A_length*, *side_B_length*, *angle_AB=None*)

Initialize self. See help(type(self)) for accurate signature.

Methods

`__init__`(side_A_length, side_B_length[, . . .])	Initialize self.
`area()`	Return the area of the kite.
`axes_of_symmetry()`	Return the number of axes of symmetry of the quadrilateral.
`perimeter()`	Return the perimeter of the quadrilateral.

Attributes

`dimensions`
`sides`

2.3 quadrilaterals.kites.rhombi.rhombus.Rhombus

class quadrilaterals.kites.rhombi.rhombus.**Rhombus**(*side_length*, *any_angle=None*)

Base class common to all rhombi.

```
>>> q = quadrilaterals.Rhombus(4, 1.5)
```

__init__(*side_length*, *any_angle=None*)

Initialize self. See help(type(self)) for accurate signature.

Methods

__init__(side_length[, any_angle])	Initialize self.
area()	Return the area of the kite.
axes_of_symmetry()	Return the number of axes of symmetry of the quadrilateral.
perimeter()	Return the perimeter of the quadrilateral.

Attributes

dimensions
sides

2.4 quadrilaterals.parallelograms.parallelogram.Parallelogram

class quadrilaterals.parallelograms.parallelogram.**Parallelogram**(*parallel_side_A_length*, *parallel_side_B_length*, *any_angle=None*)

Base class common to all parallelograms.

```
>>> import math
>>> q1 = quadrilaterals.Parallelogram(2, 5, 1.1)
>>> q2 = quadrilaterals.Parallelogram(2, 5, math.pi - 1.1)
```

__init__(*parallel_side_A_length*, *parallel_side_B_length*, *any_angle=None*)

Initialize self. See help(type(self)) for accurate signature.

Methods

__init__(parallel_side_A_length, . . . [, . . .])	Initialize self.
area()	Return the area of the parallelogram.
axes_of_symmetry()	Return the number of axes of symmetry of the quadrilateral.
perimeter()	Return the perimeter of the quadrilateral.

Attributes

`dimensions`	
`sides`	

2.5 quadrilaterals.parallelograms.rectangles.rectangle.Rectangle

class `quadrilaterals.parallelograms.rectangles.rectangle.`**`Rectangle`**(*height*, *width*)

Base class common to all rectangles.

```
>>> import quadrilaterals
>>> q = quadrilaterals.Rectangle(80, 200)
```

`__init__`(*height*, *width*)

Initialize self. See help(type(self)) for accurate signature.

Methods

`__init__`(height, width)	Initialize self.
`area`()	Return the area of the rectangle.
`axes_of_symmetry`()	Return the number of axes of symmetry of the quadrilateral.
`perimeter`()	Return the perimeter of the quadrilateral.

Attributes

`dimensions`	
`sides`	

2.6 quadrilaterals.parallelograms.rectangles.square.Square

class `quadrilaterals.parallelograms.rectangles.square.`**`Square`**(*side_length*)

Base class common to all squares.

```
>>> import quadrilaterals
>>> q = quadrilaterals.Square(0.2)
```

`__init__`(*side_length*)

Initialize self. See help(type(self)) for accurate signature.

Methods

`__init__`(side_length)	Initialize self.
`area`()	Return the area of the square.
`axes_of_symmetry`()	Return the number of axes of symmetry of the quadrilateral.
`perimeter`()	Return the perimeter of the quadrilateral.

Attributes

`dimensions`
`sides`

2.7 quadrilaterals.trapezia.trapezium.Trapezium

class quadrilaterals.trapezia.trapezium.**Trapezium**(*parallel_side_A_length*, *parallel_side_B_length*, *non_parallel_side_A_length*, *non_parallel_side_B_length*, *height=None*)

Base class common to all trapezia.

```
>>> q = quadrilaterals.Trapezium(5, 7, 5, 4, 4)
```

__init__(*parallel_side_A_length*, *parallel_side_B_length*, *non_parallel_side_A_length*, *non_parallel_side_B_length*, *height=None*)

Initialize self. See help(type(self)) for accurate signature.

Methods

`__init__`(parallel_side_A_length, . . . [, height])	Initialize self.
`area`()	Return the area of the trapezium.
`axes_of_symmetry`()	Return the number of axes of symmetry of the quadrilateral.
`perimeter`()	Return the perimeter of the quadrilateral.

Attributes

`dimensions`
`sides`

2.8 quadrilaterals.trapezia.isosceles_trapezia.isosceles_trapezium.IsoscelesTra

class quadrilaterals.trapezia.isosceles_trapezia.isosceles_trapezium.**IsoscelesTrapezium**(*par par al- lel_ non*

Base class common to all isosceles trapezia.

```
>>> q = quadrilaterals.IsoscelesTrapezium(4, 5, 6)
```

__init__(*parallel_side_A_length*, *parallel_side_B_length*, *non_parallel_sides_length*)
Initialize self. See help(type(self)) for accurate signature.

Methods

`__init__`(parallel_side_A_length, . . .)	Initialize self.
`area`()	Return the area of the isosceles trapezium.
`axes_of_symmetry`()	Return the number of axes of symmetry of the quadrilateral.
`brahmagupta_formula`(a, b, c, d)	Calculate area from side lengths for cyclic quadrilaterals only.
`perimeter`()	Return the perimeter of the quadrilateral.

Attributes

`dimensions`
`sides`

CHAPTER

THREE

INDICES AND TABLES

- genindex
- modindex
- search

INDEX

Symbols

`__init__()` (*quadrilaterals.kites.kite.Kite method*), 8
`__init__()` (*quadrilaterals.kites.rhombi.rhombus.Rhombus method*), 9
`__init__()` (*quadrilaterals.parallelograms.parallelogram.Parallelogram method*), 9
`__init__()` (*quadrilaterals.parallelograms.rectangles.rectangle.Rectangle method*), 10
`__init__()` (*quadrilaterals.parallelograms.rectangles.square.Square method*), 10
`__init__()` (*quadrilaterals.quadrilateral.Quadrilateral method*), 7
`__init__()` (*quadrilaterals.trapezia.isosceles_trapezia.isosceles_trapezium.IsoscelesTrapezium method*), 12
`__init__()` (*quadrilaterals.trapezia.trapezium.Trapezium method*), 11

I

`IsoscelesTrapezium` (*class in quadrilaterals.trapezia.isosceles_trapezia.isosceles_trapezium*), 12

K

`Kite` (*class in quadrilaterals.kites.kite*), 8

P

`Parallelogram` (*class in quadrilaterals.parallelograms.parallelogram*), 9

Q

`Quadrilateral` (*class in quadrilaterals.quadrilateral*), 7

R

`Rectangle` (*class in quadrilaterals.parallelograms.rectangles.rectangle*), 10
`Rhombus` (*class in quadrilaterals.kites.rhombi.rhombus*), 9

S

`Square` (*class in quadrilaterals.parallelograms.rectangles.square*), 10

T

`Trapezium` (*class in quadrilaterals.trapezia.trapezium*), 11