Python for physicists - exercise 9

Submission instructions - please read carefully:
• To be submitted by *** in the moodle (Lemida) system.
• *** files with py suffixes must be submitted - named exactly as detailed below for each exercise. That is to say that:
• Do not submit complete projects, libraries, zip files, etc., and do not submit all exercises in one file, but in separate files with the names listed below.
• Make sure that the files run and do what is needed (on a recent version of Python, 3.5 or higher).
• Use only the commands we learned in the practice.

Exercise 1. Submit it as file name: **ex09.py**

You are asked to write a code that handles geometric shapes.
(the code only performs calculations accordingly. For details below, it is not necessary to show the shapes).

The code will know how to work with the following shapes:
- Shape
- Quadrilateral
- Parallelogram
- Rectangle
- Rhombus
- Square

You are asked to create the code with objects (object-oriented programming), so first think which of the objects inherits properties of any of the other objects. For this purpose, first, remember the definitions of the different forms.

For example: Every rectangle is, in particular, a parallelogram, therefore a rectangle inherits a parallelogram, and every square is both a rectangle and rhombus, therefore a square inherits both a rectangle and a rhombus, etc.

Instructions:

1. In the ***Shape*** class, create a ***Constructor (__init__ method)*** that receives a list of points, where each point is by itself a list (or ***tuple***) of length 2.
   For example, you can create a shape in the following way:

   my_shape = Shape([[0, 0], [0, 1], [1, 1], [1, 0]])

   And the intention is that the polygon represented by this object will be the one with these four vertices, when straight lines connect any two adjacent points.
   Shape has a method named ***perimeter*** that returns the perimeter of the given shape.

Note: You can use the ***dist*** function from the ***math*** library.

2. Each class should perform a check in its ***Constructor*** that the form is indeed valid for the class: in the square, check that 4 points have been received; In parallel, you must check that it is indeed parallel.
   (The simplest check is according to the property that opposite sides are equal), in a rectangle, you can for example check that it is a parallelogram and in addition, it has a right angle (for example using the Pythagorean theorem[1]), In the rhombus, check that all sides are equal; And in a square, check that all the sides and all the angles are equal.[2]

3. ***Rectangle*** (and the shapes that inherit from it) should have a method called ***area*** that returns the area of the shape.

Examples:

a)

```
s = Shape([(0, 0),(1, 0),(0, 1)])
print(s.perimeter())
→ 3.41421356
```

b)

```
q = Quadrilateral([[0, 0], [1, 0], [1, 1]])
→ Error.

q = Quadrilateral([[0, 0], [1, 0], [1, 1], [0, 1]])
print(q.perimeter())
→ 4.0
```

c)

```
p = P arallelogram([(0, 0),(2, 0),(1, 1),(0, 1)])
→ Error.

r = Rectangle([(0, 0),(1, 0),(1, 1),(0, 1)])
print(r.area())
→ 1.0
```

Foot Notes:

1. You can use the function ***dist*** from the ***math*** library.

Please note that calculations with numbers of type ***float*** are not precise and therefore we do not expect to get an absolute right angle from $c^2=a^2+b^2$, but rather an approximation. So please allow a small computational error, for example, 10^{-6}.

2. If you do the class hierarchy properly, you won't really need to check anything in the case of a square.