CSCA08 Exercise 9

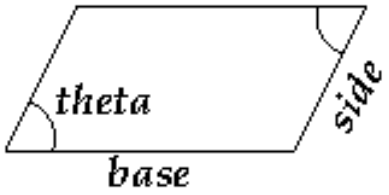
Due: November 26, 2017. 5:00pm
Pre-Run: November 24, 2017. 5:00pm

In this exercise, we will be working with inheritance, and thinking about how we can set up our inheritance hierarchy to make our lives easier. There is an easy way and a difficult way to complete this exercise. If you plan appropriately, you'll find you won't have to write very much code at all.

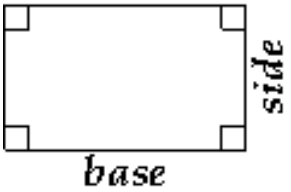
Shapes

Consider the following 4 shapes:

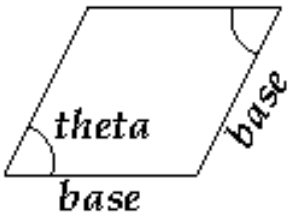
Parallelogram: a four sided figure with parallel pairs of sides. A Parallelogram is defined by the lengths of its two pairs of sides (labelled base and side in the picture below) and the interior angle (in degrees) between adjacent sides (labelled theta in the picture below).



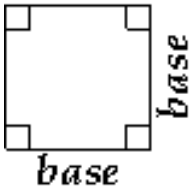
Rectangle: a parallelogram with four right angles. A Rectangle is defined by the lengths of its two pairs of sides (labelled base and side in the picture below).



Rhombus: a parallelogram with four equal sides. A Rhombus is defined by the length of its sides (labelled base in the picture below) and the interior angle (in degrees) between adjacent sides (labelled theta in the picture below).



Square: a parallelogram with four equal sides and four right angles; both a Rectangle and a Rhombus. A Square is defined by the length of its sides (labelled base in the picture below).



Your first task is to figure out the `IS-A` hierarchy of these shapes. Draw it out in a piece of paper. Are there any instances of multiple parents? [1]

Your Task

You must write four classes: `Parallelogram`, `Rectangle`, `Rhombus` and `Square`, ensuring that the class hierarchy follows your diagram from the previous step. You may create additional classes if you wish (are there any good reasons why you might want to do this?). The parameters of `__init__` methods should always be input in the following order: `(base, side, theta)` (theta being given in degrees), though of course, not every class' init will take all three. So for example, a `Rectangle` will only take `(base, side)`.

Objects of these classes must have the following methods:

- `area()` - returns the area of the shape
  - Note: The area of a parallelogram is computed by $base * side * sin(theta)$
  - Warning: function `math.sin` in Python expects its argument to be an angle given in **radians** - take a look at the function `math.radians` to convert between degrees and radians.
- `bst()` - returns a list of three floats: [`base`, `side`, `theta`]. Even if a shape doesn't need one of the parameters for its input, it should still be able to return it. (e.g., a 10 x 10 square would return: `[10.0, 10.0, 90.0]`).
- When printed, each shape should return a string with text in the following format "I am a `shape` with area `area`". Four example, a 10 x 10 square would return the string: "I am a Square with area 100" [2].

Be Lazy

One way to solve this would be to write four completely independent classes, and have each class completely implement all of their own functions. This would be a bad idea (why?). If you use inheritance correctly, you should find the exercise much simpler. Remember, you should never calculate something when you can just get another method to do the work for you. Hint: It's possible to set more than one parent for your class. Are there any shapes here for which that would be a sensible thing to do?

What to Submit

All of your code should be submitted to MarkUs in a file called `ex9.py` that doesn't import anything other than `math`, and does not use print or input. As usual your test cases should be in a file called `ex9.test`.

[1] In Python, you can inherit from multiple parents. Just add them separated by commas: `class MyClass(ParentA, ParentB)`

[2] "I am a Square with area 100.0" would also be acceptable, you don't need to do any rounding