Shapes

Problem ID: a19p01shapes

The main file, which handles input and output, is already provided. - Please only submit your class definitions, without any code outside the classes!

Implement three classes, `Shape`, `Rectangle` (which inherits from `Shape`), and `Square` (which inherits from `Rectangle`) in their own separate files, `Shape` in `shape.py`, `Rectangle` in `rectangle.py` and `Square` in `square.py`.

You should be able to figure out which methods you need to implement in these classes from the example execution code given below, which demonstrates what functionality these classes should offer.

Note:

- The `__str__()` method should return the string: "{C} with area of {A} and perimeter of {P}." where C is the name of the class, A is the area of the shape, and P is the perimeter. Both A and P should be limited to 2 decimal places in this string. It does not matter whether you skip trailing zeros.
- To get the name of the class you can use `type(self).__name__`. You could even hide that detail within yet another helper method.

This is an exercise in reusing methods in subclasses, so think about at which level in the hierarchy to implement each method, so that it can be used by other classes without the need to write the same code in multiple places. Also try to make your code extendable. For example, imagine we were to add a `Circle` class as well, or a `Triangle` class. What functionality would those have in common with the others?

Example usage

The following are examples of a main program. These two examples are also included in the supplied `main.py` file:

Example 1

```
rect = Rectangle(2, 3)
print(rect.get_area())
print(rect.get_perimeter())
print(rect)

sq = Square(2)
print(sq.get_area())
print(sq.get_perimeter())
print(sq)
```

Corresponding output:

```
6
10
Rectangle with area of 6.00 and perimeter of 10.00
4
8
Square with area of 4.00 and perimeter of 8.00
```

Example 2

```
rect = Rectangle(5, 10)
print(f"Rectangle area: {rect.get_area()}")
print(f"Rectangle perimeter: {rect.get_perimeter()}")
print(rect)

square = Square(7)
```

```
print(f"Square area: {square.get_area()}")
print(f"Square perimeter: {square.get_perimeter()}")
print(square)
```

And the corresponding output:

```
Rectangle area: 50
Rectangle perimeter: 30
Rectangle with area of 50.00 and perimeter of 30.00
Square area: 49
Square perimeter: 28
Square with area of 49.00 and perimeter of 28.00
```

Hints

- You only have to write the functions `get_area()` and `get_perimeter()` once such that it works for both `Rectangle` and `Square`.
- The `Shape` class should not implement the `get_area()` and `get_perimeter()` methods, as we have no information on how to calculate the area and perimeter of a general undefined shape.
- Only implement the `__str__()` method in the `Shape` class. If it calls methods that are only defined in the subclasses, that is fine as long as we never attempt to call the `__str__()` method directly on an instance of the `Shape` class, which we will not do.

  This class is not intended to be used directly, only serve as a placeholder for common functionality of its derived classes.
- The constructor of `Shape` can be more or less empty, using the `pass` statement.

Testing

Provided is a test file with unit tests that you can and should use to test your code locally before submitting your solution. It will provide you with quicker feedback than Gradescope does, and more detail on what is going wrong, if anything.

These are the old test cases from before we moved the assignments to Gradescope. The autograder on Gradescope uses different tests that are randomly generated and more numerous, and does not use exatly the same types of tests as the ones you can see in the old unit test file.

Below we include a description of the format of the test cases in the autograder, how the input and output is structured.

This is just FYI, and mostly serves as a documentation for us, the authors. You do not really need to worry about these details, because you do not need to write any code for dealing with this input, that is something that the autograder handles for you. We only include this description to explain what it is you are seeing in the samples, because we can not include test cases that are not secret (samples) without them appearing in the pdf automatically.

Input

In the input files for the test cases, the first line states the type of test case, which class and method the test is aimed at testing.

Then, depending on whether the class being tested is Rectangle or Square, the second line will contain one or two sidelengths, separated by a comma.

Output

The ouput should be rather self-explanatory.

Sample Input 1

```
Rectangle.get_area
2, 3
```

Sample Output 1

```
Initializing Rectangle with width=2 and height=3.
Rectangle created without errors.
Area: 6
```

Sample Input 2

```
Rectangle.get_perimeter
2, 3
```

Sample Output 2

```
Initializing Rectangle with width=2 and height=3.
Rectangle created without errors.
Perimeter: 10
```

Sample Input 3

```
Rectangle.__str__
2, 3
```

Sample Output 3

```
Initializing Rectangle with width=2 and height=3.
Rectangle created without errors.
String representation:
Rectangle with area of 6.00 and perimeter of 10.00
```

Sample Input 4

```
Square.get_area
7
```

Sample Output 4

```
Initializing Rectangle with side length 7.
Square created without errors.
Area: 49
```

Sample Input 5

```
Square.get_perimeter
7
```

Sample Output 5

```
Initializing Rectangle with side length 7.
Square created without errors.
Perimeter: 28
```

Sample Input 6

```
Square.__str__
7
```

Sample Output 6

```
Initializing Rectangle with side length 7.
Square created without errors.
String representation:
Square with area of 49.00 and perimeter of 28.00
```