Programming Assignment: Rectangle and Point classes

Problem Statement:

In this assignment you will write two classes: A Point class and a Rectangle class to represent a point and a rectangle in a 2 dimensional coordinate space. You will write the definitions for these two classes so that the client code provided in RectangleClient.py file works as expected. See Appendix for more information on the coordinate system these points and rectangles are expected to be placed. Specifically,

1. Create a Point class with the following requirements:
   - Attributes:
     - **__x**: x-coordinate of the point
     - **__y:** y-coordinate of the point
   - Two readonly Properties:
     - x: x-coordinate of the point
     - **y:** y-coordinate of the point
   - A constructor:
     - This takes two required arguments: x and y coordinates
   - One Method:
     - **translate(dx,dy)**: method moves the point by dx in x-direction and by dy in y-direction (it updates the __x and __y attributes by adding dx and dy to them respectively)
2. Create a Rectangle class with the following requirements:
   - Three static attributes:
     - **DEFAULT_WIDTH** (initialized to 1)
     - **DEFAULT_HEIGHT** (initialized to 1)
     - **rectangleCount** (initialized to 0, represents the count of rectangles instantiated so far)

   - Attributes:
     - **__topLeft**: an instance of Point class that denotes the top-left corner of the rectangle.
     - **__width:** width of the rectangle (cannot be negative or 0)
     - **__height:** height of the rectangle (cannot be negative or 0)
   - Three properties:
     - **topLeft**: Getter should return the __topLeft attribute and the setter should update the __topLeft attribute.
     - **width**: The getter method should return the __width attribute. Knowing that __width cannot be negative or 0, the setter should check if client tries to set an

invalid value, if so, the setter should print a meaningful message, and not update the __width attribute. If the new value is valid, the setter should update the __width attribute.
    - **height**: The getter method should return the __height attribute. Knowing that __height cannot be negative or 0, the setter should check if client tries to set an invalid value, if so, the setter should print a meaningful message, and not update the __height attribute. If the new value is valid, setter should update the __height attribute.
- Three **readonly** properties:
    - **bottomRight**: Returns a Point object representing the bottom right corner of the Rectangle. This will be calculated using the topLeft, width and height property values.
    - **area:** returns the area of the rectangle calculated using the width and height property values.
    - **perimeter:** returns the perimeter of the rectangle calculated using the width and height property values.
- A constructor:
    - This takes three required arguments:
        - An object of Point class representing the top-left corner of the rectangle.
        - the width of the rectangle
        - the height of the rectangle

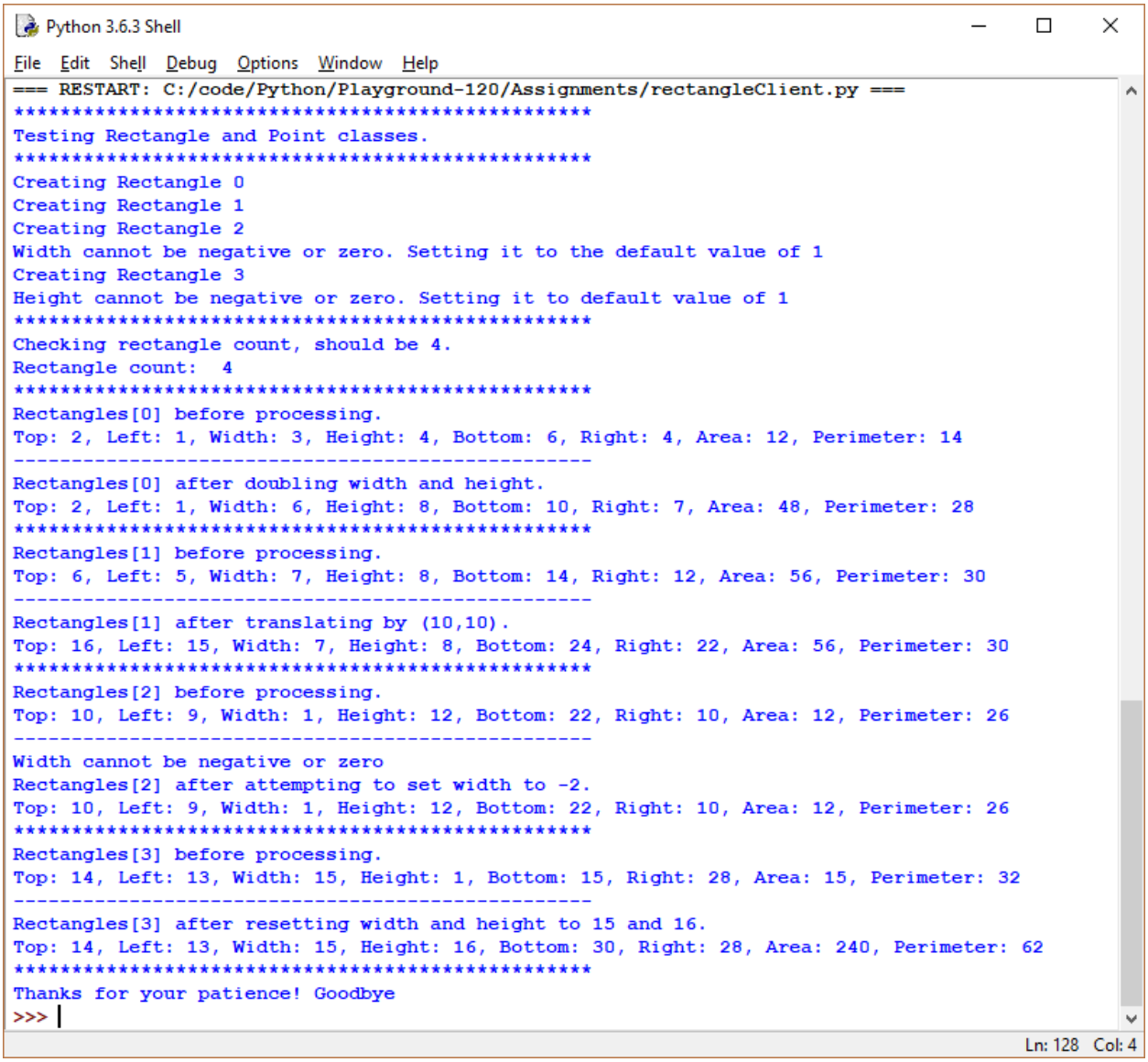
      Width and height arguments cannot be negative or 0. If either of them is invalid, the constructor should print a meaningful message. In case of error, the constructor should set the value of the attribute to the corresponding static attribute (either **DEFAULT_WIDTH** or **DEFAULT_HEIGHT** respectively.)

      The constructor should increment the static attribute **rectangleCount** (which keeps track of the number of rectangles created) by 1.

- One Method:
    - **translate(dx,dy)**: method just calls the translate method of the top left point, effectively moving the rectangle by dx in x-direction and by dy in y-direction.

Write both the classes in a single file called rectangle.py. Now run the provided rectangleClient.py file. It should run without errors and should give the output as shown in the screenshot of the sample run below. ***NOTE: you shouldn't modify the client code. It just has to run correctly and give output as shown in the screenshot below. This file has a couple of methods that verify actual values against expected ones using assert statements. If the check fails, it will raise an AssertionError exception. You can disable these checking methods by setting checkAll global variable to False in the client file.*** Submit the program in a file with a name first_last_rectangle.py. Make sure to add a block-comment at the start of the file that lists assignment title, class name, date, your name, and assignment description. Follow naming conventions.

Sample Run:



For your convenience, the same expected output is listed below in text form, with lines that will be checked highlighted in yellow.

=== RESTART: C:/code/Python/Playground-120/Assignments/rectangleClient.py ===

****

Testing Rectangle and Point classes.

****

Creating Rectangle 0

Creating Rectangle 1

Creating Rectangle 2

Width cannot be negative or zero. Setting it to the default value of 1

Creating Rectangle 3

Height cannot be negative or zero. Setting it to default value of 1

****

Checking rectangle count, should be 4.

Rectangle count: 4

****

Rectangles[0] before processing.

Top: 2, Left: 1, Width: 3, Height: 4, Bottom: 6, Right: 4, Area: 12, Perimeter: 14

Rectangles[0] after doubling width and height.

Top: 2, Left: 1, Width: 6, Height: 8, Bottom: 10, Right: 7, Area: 48, Perimeter: 28

****

Rectangles[1] before processing.

Top: 6, Left: 5, Width: 7, Height: 8, Bottom: 14, Right: 12, Area: 56, Perimeter: 30

Rectangles[1] after translating by (10,10).

Top: 16, Left: 15, Width: 7, Height: 8, Bottom: 24, Right: 22, Area: 56, Perimeter: 30

****

Rectangles[2] before processing.

Top: 10, Left: 9, Width: 1, Height: 12, Bottom: 22, Right: 10, Area: 12, Perimeter: 26

Width cannot be negative or zero

Rectangles[2] after attempting to set width to -2.

Top: 10, Left: 9, Width: 1, Height: 12, Bottom: 22, Right: 10, Area: 12, Perimeter: 26

****

Rectangles[3] before processing.

Top: 14, Left: 13, Width: 15, Height: 1, Bottom: 15, Right: 28, Area: 15, Perimeter: 32

Rectangles[3] after resetting width and height to 15 and 16.

Top: 14, Left: 13, Width: 15, Height: 16, Bottom: 30, Right: 28, Area: 240, Perimeter: 62

****

Thanks for your patience! Goodbye

>>>

Appendix: Coordinate System

Here's the coordinate system where the points and rectangles are placed with a sample Rectangle drawn with top-left point at (20, 20) and bottom right point at (60,60) with a width of 40 and height of 40.