Lab 9 – Classes, and Object-Oriented Programming

Q1: Write a python class to represent a Square in two-dimensional space. The Square class should have the following (private) attributes (or data fields) and (public) methods:

Private Attributes

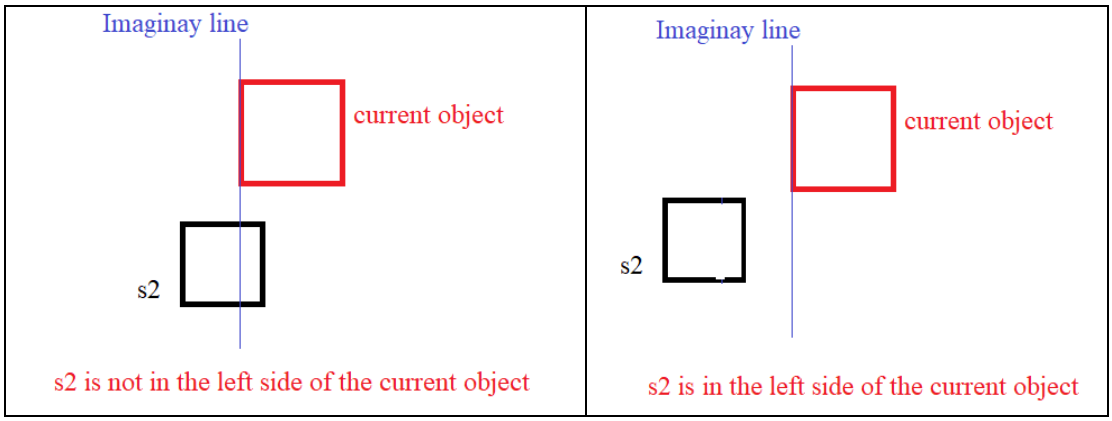
- `x` coordinate (an int), can be negative
- `y` coordinate (an int), can be negative
- `side` (a float), must be greater than 0.0

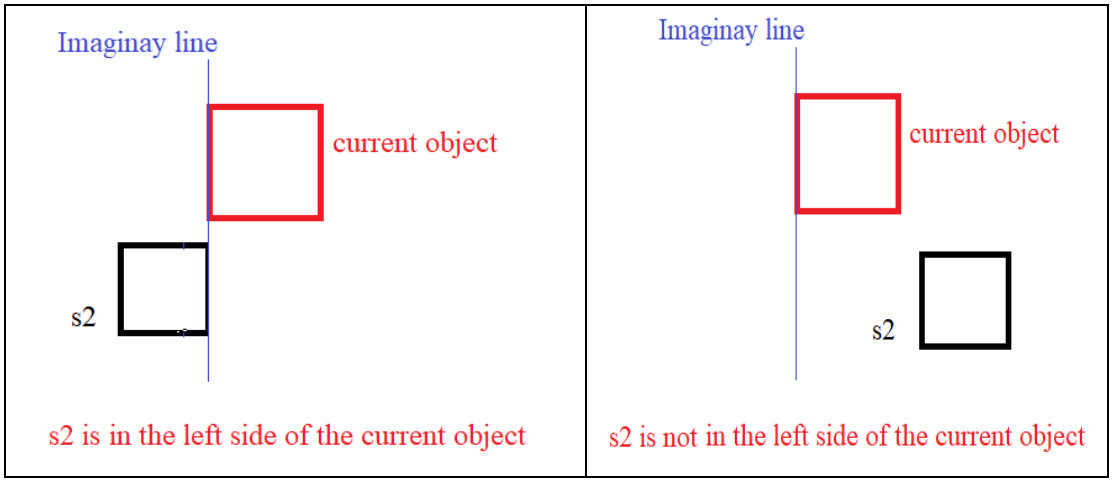
x and y-coordinates represent the location of bottom-left corner of the given Square.

Public Methods

- A constructor with parameters for x, y to represent the bottom-left corner of Square, and side to represent the side of the square. By default, it creates a Square of side 1 at position (0,0). Default values are used if there is invalid or no value.
- `getX()` - returns the Square's x coordinate
- `getY()` - returns the Square's y coordinate
- `getSide()` - returns the Square's side
- `translateXY(X, Y)` – Updates the x and y coordinate of the Square with adding X and Y to the old values, respectively.
- `setSide(s)` - changes the Square's side to s, or to 1.0 if s is invalid (negative or zero)
- `computeArea()` - computes and returns the Square's area
- `str()` - returns a string including the Square's attributes like this: "[x=12, y=17, side=10.0]"
- `isLeftside(s)` – takes a Square object, s, and returns True if square s is in the left side of the current square (calling square). In other words, imagine a vertical line that covers the left side of current square, if s is in left side of the imaginary line, it is in the left side of current square.

  The following figure explains the definitions of left side with example.





Use the following main function to test your class.

```
1  # This is a test program for class Square in shapes module
2  import shapes
3
4  def main():
5      s1 = shapes.Square()
6      print('s1='+ s1.str())

       s2 = shapes.Square(2, -1, 4)
       print('s2='+ s2.str())

       print('s1 is left side of s2:', s2.isLeftside(s1))

       s2.translateXY(-6, -1)
       print('s2={x:', s2.getX(),', y:', s2.getY(), ', side:', s2.getSide(),'}', sep='')

       print('s2 is left side of s1:', s1.isLeftside(s2))

       s2.setSide(6.5)
       print('s2='+ s2.str())

       print('s2 is left side of s1:', s1.isLeftside(s2))

   # call the main function
   main()
```

The following is the output of running the above main function:

```
= RESTART: D:/Assignments/Asgn 10/testSquare.py =
s1=[x=0, y=0, side=1.0]
s2=[x=2, y=-1, side=4]
s1 is left side of s2:  True
s2={x:-4, y:-2, side:4}
s2 is left side of s1:  True
s2=[x=-4, y=-2, side=6.5]
s2 is left side of s1:  False
>>>
```

```
class Square:
    # constructor with parameters and defualt values
    def __init__(self, x=0, y=0, s=1.0):
        self.__x = x
        self.__y = y
        if s > 0 :
            self.__side = s
        else :
            self.__side = 1.0

    # Accessor to private attribute __x
    def getX(self):
        return self.__x

    # Accessor to private attribute __y
    def getY(self):
        return self.__y

    # Accessor to private attribute __side
    def getSide(self):
        return self.__side

    # updates __x and __y by adding the corresponding parameters
    def translateXY(self, X, Y):
        self.__x += X
        self.__y += Y

    # updates __side with new value
    def setSide(self, s):
        if s > 0 :
            self.__side = s
        else :
            self.__side = 1.0

    # returns the area of the square
    def computeArea(self):
        return self.__side ** 2

    # returns information about object
    def str(self):
        return '[x=' + str(self.__x) +  \
               ', y=' + str(self.__y) + \
               ', side=' + str(self.__side) +']'

    # checkes if the parameter is in left side of this object
    def isLeftside(self, s2):
        if self.__x < s2.getX() +  s2.getSide() :
            return False
        return True
```