Lab: Polymorphism and Abstraction

Problems for in-class lab for the Python OOP Course @SoftUni.
Submit your solutions in the SoftUni judge system at https://judge.softuni.bg/Contests/1942.

1. Robots

Refactor the provided code, so we do not need to do any type-checking. The **classes** should implement the method to return the number of sensors for **each type** of robot.

2. ImageArea

Create a class called **`ImageArea`** which will store the **`width`** and the **`height`** of an image. Create a **method** called **`get_area()`** which will return the **area** of the image. We have also to implement all the magic methods for **comparison** of two image areas (**`>`**, **`>=`**, **`<`**, **`<=`**, **`==`**, **`!=`**), which will compare their areas.

Examples

Test Code	Output

`a1 = ImageArea(7, 10)`
`a2 = ImageArea(35, 2)`
`a3 = ImageArea(8, 9)`
`print(a1 == a2)`
`print(a1 != a3)`
	`True`
	`True`

`a1 = ImageArea(7, 10)`
`a2 = ImageArea(35, 2)`
`a3 = ImageArea(8, 9)`
`print(a1 != a2)`
`print(a1 >= a3)`
	`False`
	`False`

`a1 = ImageArea(7, 10)`
`a2 = ImageArea(35, 2)`
`a3 = ImageArea(8, 9)`
`print(a1 <= a2)`
`print(a1 < a3)`
	`True`
	`True`

3. Playing

Create a function called **`start_playing`** which will receive an instance and will return its **`play()`** method.

Submit only the `start_playing` function in the judge system

Examples

Test Code	Output
`class Guitar:` `    def play(self):`	`Playing the guitar`

`        return "Playing the guitar"`
 	
`guitar = Guitar()`
`print(start_playing(guitar))`	

`class Children:`
`    def play(self):`
`        return "Children are playing"`
 	
`children = Children()`
`print(start_playing(children))`
	`Children are playing`

4. Shapes

Create an abstract class **Shape** with abstract methods **`calculate_area`** and **`calculate_perimeter`**. Create classes **`Circle`** (receives radius upon initialization) and **`Rectangle`** (receives height and width upon initialization) that implement those methods (returning the result). The fields of **`Circle`** and **`Rectangle`** should be **private**.

Submit all the classes and your imports in the judge system

Examples

Test Code	Output

`circle = Circle(5)`
`print(circle.calculate_area())`
`print(circle.calculate_perimeter())`
	`78.53981633974483`
	`31.41592653589793`

`rectangle = Rectangle(10, 20)`
`print(rectangle.calculate_area())`
`print(rectangle.calculate_perimeter())`
	`200`
	`60`