# Lab: Polymorphism and Abstraction

This document defines the exercises for the ["Python OOP" course at @Software University.](https://softuni.bg/modules/74/python-advanced)

Please, submit your source code solutions for the described problems to the [Judge System](https://judge.softuni.org/Contests/1942/Polymorphism-and-Abstraction-Lab).

## Robots

**Refactor** the [provided code](https://softuni.bg/downloads/svn/python-advanced/May-2024/Python-OOP/06-Polymorphism-and-Abstraction/01_Robots.zip), 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.

\*This task is **not included** in the Judge System. You are not supposed to submit a solution for it.

## 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 the **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 |

## 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):  return "Playing the guitar"  guitar = Guitar()  print(start\_playing(guitar)) | Playing the guitar |
| class Children:  def play(self):  return "Children are playing"  children = Children()  print(start\_playing(children)) | Children are playing |

## 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 |