# Lab: Polymorphism and Abstraction

Problems for in-class lab for the [Python OOP Course @SoftUni](https://softuni.bg/courses/python-oop).

Submit your solutions in the SoftUni judge system at <https://judge.softuni.bg/Contests/1942>.

## Robots

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

## Encryption Generator

Create a class called **EncryptionGenerator**. Upon initialization it will receive a **text** (str). Develop a functionality that **encrypts the given text** changing **each character** with the next **n**th one when adding **n** (number) to the text. The valid ASCII characters you should use are between **32nd** and **126th** decimal ASCII character. If **n** is not a number raise **ValueError** with text **"You must add a number."**. For example:

* If **n** **= 10**, for character **"a"** the encrypted one is **"k"**
* If **n = 2**, for character **"}"** the encrypted one is **" "**
* If **n = -3** for character **"!"** the encrypted one is **"}"**

|  |  |
| --- | --- |
| **Test Code** | **Output** |
| some\_text = EncryptionGenerator('I Love Python!')  print(some\_text + 1)  print(some\_text + (-1)) | J!Mpwf!Qzuipo"  H~Knud~Oxsgnm |
| example = EncryptionGenerator('Super-Secret Message')  print(example + 20)  print(example + (-52)) | g\*%y'Agyw'y)4ay((u{y  ~A<1>X~1/>1@Kx1??-31 |

## 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 to also 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 |

## 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 |