# Lab: Polymorphism

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>

## Execute

Create a function called execute that receives a **function** as first argument and then **all the other arguments**

Return the **result** of the execution of the passed function with that arguments

***Submit only the execute function in the judge system***

### Examples

|  |  |
| --- | --- |
| **Test Code** | **Output** |
| def say\_hello(name, my\_name):  print(f"Hello, {name}, I am {my\_name}")  def say\_bye(name):  print(f"Bye, {name}")  execute(say\_hello, "Peter", "George")  execute(say\_bye, "Peter") | Hello, Peter, I am George  Bye, Peter |

## Instruments

Create a function called play\_instrument which will receive an instance of an **instrument** and will print it's play() method

***Submit only the play\_instrument function in the judge system***

### Examples

|  |  |
| --- | --- |
| **Test Code** | **Output** |
| class Guitar:  def play(self):  print("playing the guitar")  guitar = Guitar()  play\_instrument(guitar) | playing the guitar |
| class Piano:  def play(self):  print("playing the piano")  piano = Piano()  play\_instrument(piano) | playing the piano |

## 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 |

## 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 |