# Lab: Encapsulation

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/1938>

## Person

Create a class called **Person**. Upon initialization it should receive **name** and **age**. Create **private** properties (cannot be accessed outside the class) called **name** and **age**. Create two **instance methods**:

### Examples

|  |  |
| --- | --- |
| **Test Code** | **Output** |
| person = Person("George", 32)  print(person.name)  print(person.age) | George  20 |

## Email Validator

Create a class called **EmailValidator**. Upon initialization it should receive **min\_length** (of the username; example: in **"peter@gmail.com"** **"peter"** is the **username**), **mails** (**list** of the **valid mails**; example: **"gmail"**, **"abv"**), **domains** (**list** of **valid domains**; example: **"com"**, **"net"**). Create **three private methods**:

* **validate\_name(name)** - returns whether the name is **greater than or equal to the min\_length** (True/False)
* **validate\_mail(mail)** - returns whether the **mail is in the possible mails list** (True/False)
* **validate\_domain(domain)** - returns whether the **domain is in the possible domains list** (True/False)

Create one **public method**:

* **validate(email)** - using the **three private methods** returns whether the **email is valid** (True/False)

### Examples

|  |  |
| --- | --- |
| **Test Code** | **Output** |
| mails = ["gmail", "softuni"]  domains = ["com", "bg"]  email\_validator = EmailValidator(6, mails, domains)  print(email\_validator.validate("pe77er@gmail.com"))  print(email\_validator.validate("georgios@gmail.net"))  print(email\_validator.validate("stamatito@abv.net"))  print(email\_validator.validate("abv@softuni.bg")) | True  False  False  False |

## Mammal

Create a class called **Mammal**. Upon initialization it should receive a **name**, **type** and **sound**. Create **private class attribute** called **kingdom** and set it to be **"animals"**. Create **three more instance methods**:

* **make\_sound()** - returns a string in the format **"{name} makes {sound}"**
* **get\_kingdom()** - returns the private kingdom attribute
* **info()** - returns a string in the format **"{name} is of type {type}"**

### Examples

|  |  |
| --- | --- |
| **Test Code** | **Output** |
| mammal = Mammal("Dog", "Domestic", "Bark")  print(mammal.make\_sound())  print(mammal.get\_kingdom())  print(mammal.info()) | Dog makes Bark  Animals  Dog is of type Domestic |

## Account

Create a class called **Account**. Upon initialization it should receive an **id**, **balance** and **pin** (all numbers). The **pin** and the **id** should be **private instance attributes** and the **balance** should be **public attribute**. Create **three public instance methods**:

* **get\_id(pin)** - if the given **pin** is correct, return the **id**, otherwise return **"Wrong pin"**
* **balance** - returns the balance
* **change\_pin(old\_pin, new\_pin)** - if the old pin is **correct**, **change** it to the new one and return **"Pin changed"**, otherwise return **"Wrong pin"**

### Examples

|  |  |
| --- | --- |
| **Test Code** | **Output** |
| account = Account(8827312, 100, 3421)  print(account.get\_id(1111))  print(account.get\_id(3421))  print(account.balance)  print(account.change\_pin(2212, 4321))  print(account.change\_pin(3421, 1234)) | Wrong pin  8827312  100  Wrong pin  Pin changed |