# Lab: Encapsulation

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/1938/Encapsulation-Lab).

## Person

Create a class called **Person**. Upon initialization, it should receive a **name** and an **age**. Name mangle **the name** and **the** **age** attributes (should not be accessed outside the class). Create two **instance methods** called **get\_name** and **get\_age** to return the values of the private attributes.

### Examples

|  |  |
| --- | --- |
| **Test Code** | **Output** |
| person = Person("George", 32)  print(person.get\_name())  print(person.get\_age()) | George  32 |

## Mammal

Create a class called **Mammal**. Upon initialization, it should receive a **name**, a **type**, and a **sound**. Create a **class attribute** called **kingdom** which should not be accessed outside the class and set it to **"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 |

## Profile

Create a class called **Profile**. Upon initialization, it should receive:

* **username: str** - the username should be between 5 and 15 characters (inclusive). If it is not, raise a **ValueError** with the message **"The username must be between 5 and 15 characters."**
* **password: str** - the password must be at **least 8 characters** long; it must contain at least **one** **upper case letter** and at least **one digit**. If it does not, raise a **ValueError** with the message **"The password must be 8 or more characters with at least 1 digit and 1 uppercase letter."**

Hint: Use **Getters** and **Setters** to name-mangle them.

Override the **\_\_str\_\_()** method of the base class, so it returns: **"You have a profile with username: "{username}" and password: {"\*" with the length of password}"**.

### Examples

|  |  |
| --- | --- |
| **Test Code** | **Output** |
| profile\_with\_invalid\_password = Profile('My\_username', 'My-password') | ValueError: The password must be 8 or more characters with at least 1 digit and 1 uppercase letter. |
| profile\_with\_invalid\_username = Profile('Too\_long\_username', 'Any') | ValueError: The username must be between 5 and 15 characters. |
| correct\_profile = Profile("Username", "Passw0rd")  print(correct\_profile) | You have a profile with username: "Username" and password: \*\*\*\*\*\*\*\* |

## 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 methods that should not be accessed** **outside** the class:

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

Create one **public method**:

* **validate(email)** - using the **three 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 |

## Account

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

* **get\_id(pin)** - if the given **pin** is correct, return the **id**, otherwise, return **"Wrong pin"**
* **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 |