## 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 which 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 |