# Lab: Classes and Instances

Problems for exercise and homework 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/1936>

## Smartphone

Create a class called **Smartphone**. Upon initialization it should receive a **memory** (number). It should also have **2 other attributes**: **apps** (empty list by default) and **is\_on** (False by default). Create **3 methods**:

* **power()** - sets **is\_on** on **True** if the phone is off, otherwise sets it to False
* **install(app, app\_memory)**
  + If there is **enough memory** on the phone and it **is on**, install the app (**add it to apps** and **decrease the memory** of the phone) and return **"Installing {app}"**
  + If there **is enough memory**, but the **phone is off**, return **"Turn on your phone to install {app}"**
  + Otherwise return **"Not enough memory to install {app}"**
* **status()** - returns **"Total apps: {total\_apps\_count}. Memory left: {memory\_left}"**

### Examples

|  |  |
| --- | --- |
| **Test Code** | **Output** |
| smartphone = Smartphone(100)  print(smartphone.install("Facebook", 60))  smartphone.power()  print(smartphone.install("Facebook", 60))  print(smartphone.install("Messenger", 20))  print(smartphone.install("Instagram", 40))  print(smartphone.status()) | Turn on your phone to install Facebook  Installing Facebook  Installing Messenger  Not enough memory to install Instagram  Total apps: 2. Memory left: 20 |

## Vet

Create a class called **Vet**. Upon initialization it should receive a **name** (string). It should also have an **instance attribute** called **animals** (empty list by default). There should also be **2 class attributes**: **animals** (empty list) which will store the total amount of **animals for all vets**; **space** (5 by default). You have to create **3 more instance methods**

* **register\_animal(animal\_name)**
  + If there **is space** in the vet clinic add the animal to **both animals lists** and return a message: **"{name} registered in the clinic"**
  + Otherwise return **"Not enough space"**
* **unregister\_animal(animal\_name)**
  + If the animal is **in the relevant vet list**, **remove** it from **both animals** lists and return **"{animal} unregistered successfully"**
  + Otherwise, return **"{animal} not in the clinic"**
* **info()** - returns **"{vet\_name} has {amount\_of\_his\_animals} animals. {space\_left\_in\_clinic} space left in clinic"**

### Examples

|  |  |
| --- | --- |
| **Test Code** | **Output** |
| peter = Vet("Peter")  george = Vet("George")  print(peter.register\_animal("Tom"))  print(george.register\_animal("Cory"))  print(peter.register\_animal("Fishy"))  print(peter.register\_animal("Bobby"))  print(george.register\_animal("Kay"))  print(george.unregister\_animal("Cory"))  print(peter.register\_animal("Silky"))  print(peter.unregister\_animal("Molly"))  print(peter.unregister\_animal("Tom"))  print(peter.info())  print(george.info()) | Tom registered in the clinic  Cory registered in the clinic  Fishy registered in the clinic  Bobby registered in the clinic  Kay registered in the clinic  Cory unregistered successfully  Silky registered in the clinic  Molly not in the clinic  Tom unregistered successfully  Peter has 3 animals. 1 space left in clinic  George has 1 animals. 1 space left in clinic |

## Glass

Create a class called **Glass**. Upon initialization it will **not receive any parameters**, you must create however an **instance attribute** called **content** which should be equal to **0**. You should also create a **class attribute** called **capacity** which should be **250 ml**. Create **3 more instance methods**:

* **fill(ml)** - fill the glass with the given milliliters if there is **enough space** in it and return **"Glass filled with {ml} ml"**, otherwise return **"Cannot add {ml} ml"**
* **empty()** - empty the glass and return **"Glass is now empty"**
* **info()** - returns info about the glass in the format **"****{space\_left} ml left"**

### Examples

|  |  |
| --- | --- |
| **Test Code** | **Output** |
| glass = Glass()  print(glass.fill(100))  print(glass.fill(200))  print(glass.empty())  print(glass.fill(200))  print(glass.info()) | Glass filled with 100 ml  Cannot add 200 ml  Glass is now empty  Glass filled with 200 ml  50 ml left |