Lab: Classes and Instances

1. 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, 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 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}. Memory left: {left_memory}"**

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`

2. 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 of each vet**; **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 clinic**, **remove** it from the **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. {left_space_in_clinic} space left in the clinic"**

Examples

Test Code	Output

`peter = Vet("Peter")`
`george = Vet("George")`
`print(peter.register_animal("Tom"))`
`print(george.register_animal("Cory"))`
	`Tom registered in the clinic`
	`Cory registered in the clinic`
	`Fishy registered in the clinic`
	`Bobby registered in the clinic`

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())	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

3. 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 **`"{left_space} 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