Exercise: Defining Classes

Problems for exercise and homework for the Python OOP Course @SoftUni. Submit your solutions in the SoftUni judge system at https://judge.softuni.bg/Contests/1935

1. Car

Create a class called **Car**. Upon initialization it should receive a **name**, **model** and **engine** (all strings). Create a method called **get_info()** which will return a string in the following format:
"This is {name} {model} with engine {engine}".

Examples

Test Code	Output
`car = Car("Kia", "Rio", "1.3L B3 I4")` `print(car.get_info())`	`This is Kia Rio with engine 1.3L B3 I4`

2. Shop

Create a class called **Shop**. Upon initialization it should receive a **name** (string) and **items** (list). Create a method called **get_items_count()** which should return the **amount of items** in the store.

Examples

Test Code	Output
`shop = Shop("My Shop", ["Apples", "Bananas", "Cucumbers"])` `print(shop.get_items_count())`	3

3. Hero

Create a class called **Hero**. Upon initialization it should receive a **name** (string) and **health** (number). Create two functions:

- **defend(damage)** - Deal the given **damage** to the hero; if the **health** is 0 or less, **set** it **to 0** and **return "{name} was defeated"**.

- **heal(amount)** - **Increase the health** of the hero with the given amount.

Examples

Test Code	Output

`hero = Hero("Peter", 100)`
`print(hero.defend(50))`
`hero.heal(50)`
`print(hero.defend(99))`
`print(hero.defend(1))`
	
	`None`
	`None`
	`Peter was defeated`

4. Steam User

Create a class called **SteamUser**. Upon initialization it should receive **username** (string), **games** (list). It should also have an **attribute** called **played_hours** (**0** by default). Add **three methods** to the class:

- **play(game, hours)**

- If the **game** is in the user **games increase** the **`played_hours`** by the given hours and return **`"{username} is playing {game}"`**
- Otherwise, return **`"{game} is not in library"`**

- **`buy_game(game)`**
  - If the game **is not** already in the user's **games**, **add it** and return **`"{username} bought {game}"`**
  - Otherwise return **`"{game} is already in your library"`**
- **`stats()`** - returns **`"{username} has {games_count} games. Total play time: {played_hours}"`**

Examples

Test Code	Output

`user = SteamUser("Peter", ["Rainbow Six Siege", "CS:GO", "Fortnite"])`
`print(user.play("Fortnite", 3))`
`print(user.play("Oxygen Not Included", 5))`
`print(user.buy_game("CS:GO"))`
`print(user.buy_game("Oxygen Not Included"))`
`print(user.play("Oxygen Not Included", 6))`
`print(user.stats())`
	`Peter is playing Fortnite`
	`Oxygen Not Included not in library`
	`CS:GO is already in your library`
	`Peter bought Oxygen Not Included`
	`Peter is playing Oxygen Not Included`
	`Peter has 4 games. Total play time: 9`

5. Programmer

Create a class called **`Programmer`**. Upon initialization it should receive **`name`** (string), **`language`** (string), **`skills`** (integer). The class should have **two methods**:

- **`watch_course(course_name, language, skills_earned)`**
  - If the programmer's **language** is the **equal** to the **one on the course increase his skills** with the given one and return a message **`"{programmer} watched {course_name}"`**.
  - Otherwise return **`"{name} does not know {language}"`**.
- **`change_language(new_language, skills_needed)`**
  - If the programmer **has the skills** and the **language is different from his**, **change** his language to the new one and return **`"{name} switched from {previous_language} to {new_language}"`**.
  - If the programmer **has the skills**, but the **language is the same** as his return **`"{name} already knows {language}"`**.
  - In the last case the programmer does **not have the skills**, so return **`"{name} needs {needed_skills} more skills"`** and **don't change his language**

Examples

Test Code	Output

`programmer = Programmer("John", "Java", 50)`
`print(programmer.watch_course("Python Masterclass", "Python", 84))`
`print(programmer.change_language("Java", 30))`
`print(programmer.change_language("Python", 100))`
`print(programmer.watch_course("Java: zero to hero", "Java", 50))`
`print(programmer.change_language("Python", 100))`
`print(programmer.watch_course("Python Masterclass", "Python", 84))`
	`John does not know Python`
	`John already knows Java`
	`John needs 50 more skills`
	`John watched Java: zero to hero`
	`John switched from Java to Python`
	`John watched Python Masterclass`