Lab: Attributes and Methods

Problems for in-class lab for the Python OOP Course.
Submit github link.

1. Store (10 points)

Create a class called **Store**. Upon initialization it should receive a **name (str)**, **type (str)**, **capacity (int)**. The store should also have an **attribute** called **items** (**dictionary** that stores **name** of an item and its **quantity**). The class should have **4 methods**:

- **`from_size (name:str, type:str, size:int)`** – a **new instance** should be created with capacity which is **50% of the size**
- **`add_item(item_name:str)`** – adds **1** to the quantity of the given **item**. On **success**, the method should **return `"{item_name} added to the store"`.** If the addition is **not possible**, the following message should be returned **`"Not enough capacity in the store"`**
- **`remove_item(item_name:str, amount:int)`** – **removes** the given amount from the **item**. On **success**, it should return **`"{count} {item_name} removed from the store"`**. **Otherwise**, the method should return **`"Cannot remove {count} {item_name}"`**
- **`__repr__()`** - returns a string representation in the format **`"{store_name} of type {store_type} with capacity {store_capacity}"`**

Examples

Test Code

`first_store = Store("First store", "Fruit and Veg", 20)`
`second_store = Store.from_size("Second store", "Clothes", 500)`
 	
`print(first_store)`
`print(second_store)`
 	
`print(first_store.add_item("potato"))`
`print(second_store.add_item("jeans"))`
`print(first_store.remove_item("tomatoes", 1))`
`print(second_store.remove_item("jeans", 1))`

Output

`First store of type Fruit and Veg with capacity 20`
`Second store of type Clothes with capacity 250`
`potato added to the store`
`jeans added to the store`
`Cannot remove 1 tomatoes`
`1 jeans removed from the store`

2. Integer (20 points)

Create a class called **Integer**. Upon initialization it should receive a single parameter **value** (**int**). It should have **4 methods**:

- **`from_float(value)`** - creates a **new instance** by **flooring** the provided floating number. If the value is **not a float** return a message **`"value is not a float"`**
- **`from_roman(value)`** – creates a **new instance** by converting the **roman** number (**as string**) to an integer

- **`from_string(value)`** - creates a **new instance** by converting the **string** to an integer (if the value **cannot be converted**, return a message **`"wrong type"`**)
- **`add(integer:Integer)`** – adds the values of the **two instances** and returns the result (if the integer is **not an instance of Integer**, return the message **`"number should be an Integer instance"`**)

Examples

Test Code

`first_num = Integer(10)`
`second_num = Integer.from_roman("IV")`
 	
`print(Integer.from_float("2.6"))`
`print(Integer.from_string(2.6))`
`print(first_num.add(second_num))`

Output

`value is not a float`
`wrong type`
`14`

3. Calculator (30 points)

Create a class called **`Calculator`** that has the following **static methods**:

- **`add(*args)`** – **sums** all the arguments passed to the function and **returns the result**
- **`multiply(*args)`** – **multiplies** all the numbers and **returns the result**
- **`divide(*args)`** – **divides** all the numbers and returns the **result**
- **`subtract(*args)`** – **subtracts** all the numbers and returns the **result**

Examples

Test Code

`print(Calculator.add(5, 10, 4))`
`print(Calculator.multiply(1, 2, 3, 5))`
`print(Calculator.divide(100, 2))`
`print(Calculator.subtract(90, 20, -50, 43, 7))`

Output

`19`
`30`
`50.0`
`70`

4. Hotel Rooms (40 points)

In a folder called **`project`** create two files: **`hotel.py`** and **`room.py`**

In the **`room.py`** file create a class called **Room**. Upon **initialization** it should receive a **`number`** (**int**) and a **`capacity`** (**int**). It should also have an **attribute** called **`guests`** (**0** upon initialization) and **`is_taken`** (**`False`** upon initialization). The class should have **2 methods**:

- **`take_room(people)`** – if the room is **not taken**, and there is **enough space**, the guests take the room. Otherwise, the method should return **`"Room number {number} cannot be taken"`**

- **`free_room()`** – if the room **is taken**, free it. Otherwise, return **`"Room number {number} is not taken"`**

In the **`hotel.py`** file create a class called **`Hotel`**. Upon initialization it should receive a **`name`** (**`str`**). It should also have 2 **more attributes**: **`rooms`** (empty **`list`** of rooms) and **`guests`** (**`0`** upon initialization). The class should have **5 more methods**:

- **`from_stars(stars_count)`** – creates a new instance with name **`"{stars_count} stars Hotel"`**
- **`add_room(room)`** – add the room to the list of rooms
- **`take_room(room_number, people)`** – find the room with that **number** and try to **accommodate** the **guests** in the room
- **`free_room(room_number)`** – find the room with that **number** and **free it**
- **`print_status()`** – prints information about the hotel in the following format:

  `Hotel {name} has {guests} total guests`
  `Free rooms: {numbers of all free rooms separated by comma and space}`
  `Taken rooms: {numbers of all taken rooms separated by comma and space}`

Examples

Test Code

`from project.hotel import Hotel`
`from project.room import Room`
 	
`hotel = Hotel.from_stars(5)`
 	
`first_room = Room(1, 3)`
`second_room = Room(2, 2)`
`third_room = Room(3, 1)`
 	
`hotel.add_room(first_room)`
`hotel.add_room(second_room)`
`hotel.add_room(third_room)`
 	
`hotel.take_room(1, 4)`
`hotel.take_room(1, 2)`
`hotel.take_room(3, 1)`
`hotel.take_room(3, 1)`
 	
`hotel.print_status()`

Output

`Hotel 5 stars Hotel has 3 total guests`
`Free rooms: 2`
`Taken rooms: 1, 3`