School of Engineering

Applied MSc in Data Science & Artificial Intelligence

Applied MSc in Data Engineering & Artificial Intelligence

Course: **Software Engineering Part 2**

Project: **Item, Suitcase and Cargo hold**

Instructor: **Hanna Abi Akl**

Project Objectives:

The project requires you to use Python Object-Oriented Programming (OOP) knowledge to create the classes **Item**, **Suitcase** and **Cargo hold**. This will let you further practice working on objects which contain references to other objects.

The project is constructed in "Parts" that are ordered and serve as a guide towards the final implementation.

Please note that this is an individual project and any suspicion in similarities between solutions will automatically result in a failing grade for the students involved. ChatGPT-like

AI-generated solutions will also be penalized (believe me, I can tell when they write your code 😊).

Part 1: Item

Please create a class named Item which is used to create items of different kinds. Each item has a name and a weight (in kilograms).

You can use the following code to test your class:

```
book = Item("ABC Book", 2)
phone = Item("Nokia 3210", 1)

print("Name of the book:", book.name())
print("Weight of the book:", book.weight())

print("Book:", book)
print("Phone:", phone)
```

Your program should print out this:

Sample output

```
Name of the book: ABC Book
Weight of the book: 2
Book: ABC Book (2 kg)
Phone: Nokia 3210 (1 kg)
```

An **Item** should provide the methods **weight** and **name**, which return the values stored in those attributes.

The name and weight should be encapsulated within the class. The following code should not work:

```
book = Item("ABC Book", 2)
book.weight = 10
```

Part 2: Suitcase

Please write a class named **Suitcase**. You should be able to pack items into a suitcase. A suitcase also has a maximum combined weight for the items stored within.

Your class should contain the following members:

- a constructor which takes the maximum weight as an argument
- a method named **add_item** which adds the item given as an argument to the suitcase. The method has no return value.
- a **__str__** method which returns a string in the format "x items (y kg)"

The class should make sure that the combined weight of the items stored within any **Suitcase** does not exceed the maximum weight set for that instance. If the maximum weight would be exceeded when the **add_item** method is called, the new item should not be added to the suitcase.

Your class should work as follows:

```
book = Item("ABC Book", 2)
phone = Item("Nokia 3210", 1)
brick = Item("Brick", 4)

suitcase = Suitcase(5)
print(suitcase)

suitcase.add_item(book)
print(suitcase)

suitcase.add_item(phone)
print(suitcase)

suitcase.add_item(brick)
print(suitcase)
```

Executing the above should print out:

Sample output

```
0 items (0 kg)
1 items (2 kg)
2 items (3 kg)
2 items (3 kg)
```

Part 3: Mind your language

The notification "1 items" is not very grammatical. Instead, it should say "1 item". Please make the required changes to your **__str__** method.

The previous example should now print out:

Sample output

```
0 items (0 kg)
1 item (2 kg)
2 items (3 kg)
2 items (3 kg)
```

Part 4: All the items

Please add the following methods to your **Suitcase** class definition:

- **print_items** prints out all the items stored in the suitcase
- **weight** returns an integer number representing the combined weight of all the items stored in the suitcase

Your class should now work with the following program:

```
book = Item("ABC Book", 2)
phone = Item("Nokia 3210", 1)
brick = Item("Brick", 4)

suitcase = Suitcase(10)
suitcase.add_item(book)
suitcase.add_item(phone)
suitcase.add_item(brick)

print("The suitcase contains the following items:")
suitcase.print_items()
combined_weight = suitcase.weight()
print(f"Combined weight: {combined_weight} kg")
```

Executing the above program should print out this:

Sample output

```
The suitcase contains the following items:
ABC Book (2 kg)
Nokia 3210 (1 kg)
Brick (4 kg)
Combined weight: 7 kg
```

If you have implemented your **Suitcase** class with more than two instance variables, please make the required changes so that each instance has only two data attributes: the maximum weight and a list of items within.

Part 5: The heaviest item

Please add a new method to your **Suitcase** class: **heaviest_item** should return the **Item** which is the heaviest. If there are two or more items with the same, heaviest weight, the method may return any one of these. The method should return a reference to an object. If the suitcase is empty, the method should return **None**.

Your class should now work with the following program:

```
book = Item("ABC Book", 2)
phone = Item("Nokia 3210", 1)
brick = Item("Brick", 4)

suitcase = Suitcase(10)
suitcase.add_item(book)
suitcase.add_item(phone)
suitcase.add_item(brick)

heaviest = suitcase.heaviest_item()
print(f"The heaviest item: {heaviest}")
```

Executing the above program should print out this:

Sample output

```
The heaviest item: Brick (4 kg)
```

Part 6: Cargo hold

Please write a class named **CargoHold** with the following methods:

- a constructor which takes the maximum weight as an argument
- a method named **add_suitcase** which adds the suitcase given as an argument to the cargo hold
- a **__str__** method which returns a string in the format "x suitcases, space for y kg"

The class should make sure that the combined weight of the items stored within any **CargoHold** does not exceed the maximum weight set for that instance. If the maximum weight would be exceeded when the **add_suitcase** method is called, the new suitcase should not be added to the cargo hold.

Your class should now work with the following program:

```
cargo_hold = CargoHold(1000)
print(cargo_hold)

book = Item("ABC Book", 2)
phone = Item("Nokia 3210", 1)
brick = Item("Brick", 4)

adas_suitcase = Suitcase(10)
adas_suitcase.add_item(book)
adas_suitcase.add_item(phone)

peters_suitcase = Suitcase(10)
peters_suitcase.add_item(brick)

cargo_hold.add_suitcase(adas_suitcase)
print(cargo_hold)

cargo_hold.add_suitcase(peters_suitcase)
print(cargo_hold)
```

Executing the above program should print out this:

Sample output

```
0 suitcases, space for 1000 kg
1 suitcase, space for 997 kg
2 suitcases, space for 993 kg
```

Part 7: The contents of the cargo hold

Please add a method named **print_items** to your **CargoHold** class. It should print out all the items in all the suitcases within the cargo hold.

Your class should now work with the following program:

```
book = Item("ABC Book", 2)
phone = Item("Nokia 3210", 1)
brick = Item("Brick", 4)

adas_suitcase = Suitcase(10)
adas_suitcase.add_item(book)
adas_suitcase.add_item(phone)

peters_suitcase = Suitcase(10)
peters_suitcase.add_item(brick)

cargo_hold = CargoHold(1000)
cargo_hold.add_suitcase(adas_suitcase)
cargo_hold.add_suitcase(peters_suitcase)

print("The suitcases in the cargo hold contain the following items:")
cargo_hold.print_items()
```

Executing the above program should print out this:

Sample output

```
The suitcases in the cargo hold contain the following items:
ABC Book (2 kg)
Nokia 3210 (1 kg)
Brick (4 kg)
```

Project Deliverables:

Each student must submit the following files:

- A **classes.py** script: The script containing the class implementations.
- A **main.py** script: The script containing the test code for class objects.

Project Evaluation:

The project will be evaluated using the following rubric:

- Part 1 **[10 point]**
- Part 2 **[20 point]**
- Part 3 **[10 point]**
- Part 4 **[20 point]**
- Part 5 **[10 point]**

- Part 6 **[20 point]**
- Part 7 **[10 point]**