In this project you will define a class MovingBox which is used to model a moving box and its content. The finished class must have methods that can be used to add items into a moving box, remove them, search for a particular item etc.

The assignment includes a relatively large code template [movingbox.py](https://plus.tuni.fi/graderS/static/compcs100-f2021/_downloads/movingbox1.py), which contains a ready user interface for program using the MovingBox class. Your job is to define the said class in a way which is compatible with the way code template uses it.

**Note**

In the code template there is a clearly marked place (the class definition) where you are expected to add your code. You are not allowed to modify the other parts of the template.

**Program Behavior**

To understand what code you need to write it is important to first understand what the code given in the template does. This is because the ready given code makes assumptions about the MovingBox class' interface (i.e. the methods).

The user interface in the template implements the following 9 commands. All of them besides quit use the methods of the MovingBox class to create the expected behavior. With every command, study carefully the helper function connected to it (add ⟶ add\_to\_box, list ⟶ list\_box\_content etc.) to understand what kind of methods (name, parameters, and return value) you need to implement for the MovingBox class.

**quit**

Ends the program. The program will also end, if the user enters an empty line.

**newbox**

Creates a new moving box with the given name and stores it internally into a data structure for the future use (see the descriptions of the rest of the commands).

next command> newbox bathroom-box-03

next command> newbox kitchen-box-01

next command> newbox bedroom-box-05

next command>

If a moving box with the same name already existed, it and its content will be lost.

**add**

Adds the given amount of named items into a named moving box. If the box already contained items with the same name, the new items will be combined with the old ones.

next command> add bedroom-box-05 pillow 1

next command> add kitchen-box-01 knife 4

next command> add bedroom-box-05 pillow 3

next command> add kitchen-box-01 fork 6

next command>

**list**

Lists in alphabetical order the items and their amounts stored in the named box.

next command> list kitchen-box-01

Box "kitchen-box-01" contains 10 items.

6 fork

4 knife

next command> list bedroom-box-05

Box "bedroom-box-05" contains 4 items.

4 pillow

next command> list bathroom-box-03

Box "bathroom-box-03" contains 0 items.

next command>

**list\_all**

Prints in alphabetical order the boxes and their contents on the screen. The output format for each box is exactly the same as in list command above.

next command> list\_all

Box "bathroom-box-03" contains 0 items.

Box "bedroom-box-05" contains 4 items.

4 pillow

Box "kitchen-box-01" contains 10 items.

6 fork

4 knife

next command>

**remove**

Removes a given amount of a named item from a named moving box.

next command> remove kitchen-box-01 fork 3

next command> remove bedroom-box-05 pillow 4

next command> list\_all

Box "bathroom-box-03" contains 0 items.

Box "bedroom-box-05" contains 0 items.

Box "kitchen-box-01" contains 7 items.

3 fork

4 knife

next command>

**transfer**

Moves a given number of named items from one box to another.

next command> transfer kitchen-box-01 bedroom-box-05 fork 2

next command> transfer kitchen-box-01 bathroom-box-03 knife 4

next command> transfer kitchen-box-01 bathroom-box-03 fork 1

next command> list\_all

Box "bathroom-box-03" contains 5 items.

1 fork

4 knife

Box "bedroom-box-05" contains 2 items.

2 fork

Box "kitchen-box-01" contains 0 items.

next command>

**search**

Goes though all the boxes in alphabetical order and searches them for a given item. Prints the information about all the boxes where the item was found in. In addition to this, the number of that item in each box is shown.

next command> search knife

Box bathroom-box-03: 4

next command> search fork

Box bathroom-box-03: 1

Box bedroom-box-05: 2

next command> search rubber\_duck

next command>

**count**

Counts the total number of a particular item found collectively in all the boxes.

next command> count rubber\_duck

There are total 0 rubber\_duck(s) in the boxes.

next command> count fork

There are total 3 fork(s) in the boxes.

next command> count knife

There are total 4 knife(s) in the boxes.

next command>

Please note, that the colorful text in the examples is used to emphasize that there is user input from the keyboard at that point. You are not expected to have your Python code to print colored text.

Another point worth noting is, that the code template takes care of all the error messages: no need for you to worry about them. Although, some error conditions require that the methods you implement return a particular value if the error happens. That is something you do need to take care of.

Even though the behaviour of the different commands was explained in detail above, remember that you only need to implement the MovingBox class so that it is compatible with the code template. This is also a good example of the main benefit of using classes: the way we use the class in the template code has no interest in how the class is actually implemented: use and implementation of the class are separated from each other.