Exercises – Part 2: Containers

Do not use built-in functions except when stated otherwise! Name your functions exactly like in the specification. Upload your code as a .py file, e.g. **`exercises_part_2.py`**, on Moodle.

Do not copy and paste code from others or the internet! Find your own solution, there are many different correct ways to complete each exercise.

Exercise 1: Counting

Write a function **`count_a_number(numbers, number)`** that accepts a list of integers named **`numbers`** and an integer named **`number`** as arguments. It counts the occurrence of the integer number in the list and returns the count as an integer. Do not use-built-in functions! Use loops to solve the problem.

Exercise 2: Playing with lists

Write a function **`play_with_lists(numbers, number)`** that accepts a list of integers named **`numbers`** and an integer named **`number`** as arguments. Use built-in list functions to achieve the following:

- Print out the list in reverse order but leave the original list in order
- Replace the given integer **`number`** within the list with the number **`1`** and print it on the console
- Print out a sorted version of the list in descending order

Don't forget that lists are mutable. You probably will have to make a copy if you want the leave the original list untouched. Investigate the difference between using the build-in function **`sorted(list)`** and the **`sort`** function of the class **`list`**. Write a comment why you chose the sorting function you used.

The function does not return anything.

Exercise 3: Comparing list elements

Write a function **`compare_lists(list1, list2)`** that accepts two list as arguments. The function looks for elements that the two lists have in common. It returns a list containing all those elements. This list may be empty if there are no common elements.

Exercise 4: No duplicates!

Write a function **`remove_duplicates(items)`** that accepts a list of strings named **`items`** as argument and removes all duplicate values from the list. There is an easy way to do this using another container. Search for this way and implement it.

Then write another function named **`remove_duplicates_my_way(items)`**. This time find a way to accomplish the task without using another container.

Both functions return the list of strings without duplicates.

Exercise 5: Computer description

Write a function **`describe_computer(computer)`** that accepts a dictionary named **`computer`** as argument. The given dictionary contains the keys **`"Type"`**, **`"Brand"`** and **`"Price"`**. The function prints out the values of the dictionary in the following format:

```
You have a TYPE from BRAND that costs PRICE€.
```

The capitalized words like TYPE and BRAND represent the values assigned to the keys in the dictionary.

Then the function adds the key **`"OS"`** to the dictionary and defaults it to **`"Linux"`**. Afterwards it prints the dictionary to the console.

Example:

```
my_notebook = {'Type': 'Notebook', 'Brand': 'Dell', 'Price': 2000}

describe_computer(my_notebook)
```

Example output:

```
You have a Notebook from Dell that costs 2000€.

{'Type': 'Notebook', 'Brand': 'Dell', 'Price': 2000, 'OS': 'Linux'}
```

If one of the keys is not present, the value used in the output defaults to a custom text like “unknown brand” for the key “Brand”.

The function does not return anything.