General Technical Exercise

Shared Exercises

These are technical exercises, that guide in going through the content of the first part of the workshop again. The first ones should be directly solvable with what you saw in the first part. However, the most important skill in programming is arguably to know how to find and read resources like documentations and code examples. So try to solve each exercise without looking at the hints, but feel free to use whatever else information source. If you feel a bit stuck look at the hint. If you are still stuck, do not hesitate to reach out to one of the trainers - that is what they are there for! After finishing an exercise, you might still want to read the hints - sometimes they might give you an inspiration how to approach the exercise differently. After finishing a few (e.g. 5 or so) of the exercises, you might find it useful to also reach out to one of the trainers to discuss it.

Note, that each of these exercises (as pretty much any programming problem) has multiple correct solutions. It can be very instructive to implement several of them and discuss what the advantages and disadvantages they might have.

Exercise 1

Create a (Python) list of your team's members names, e.g. `names = ['Paul', 'Marie', 'Isabell', 'Jochem', ]`. Print it out.

Exercise 2

Using the same list, print each name on an individual line.

Exercise 3

Repeat exercise 2, but print the names in reverse order.

Exercise 4

Have the user `input` a number, that you assign to `n`. Print the first `n` names of the list. Ask the user for a different number if the input is no (positive) number or `n` is larger than the list is long!

Exercise 5

Research *'python built in list methods'*. Find the functions required to sort a list in ascending and descending order, find the amount of occurrences for a certain element as well as the ones to remove and add elements to the list. You can use this list as a test-object:
`weird_list = [5, 3, 5, 2, 4, 5, 7, 0, 1, 'Paul']`
(You can for instance count how many 5's and 0's there are, remove *Paul* and add *Pauline*.)

Exercise 6

For the same list, print all elements that are larger than three.

Exercise 7

Take the list
`weird_list = [{}, [], 5.1, 2e5, None, 5, 1j, True, "Pauline", 'Paul', 'c']`
Print the type of each element on a separate line.

Exercise 8

In the previous exercise we saw, that we can place a list within a list (second entry). Inspect this list
`nested_list = [[1,2], [3,5]]`
This is called a nested list. Write a piece of code, that replaces the five with a four!

Exercise 9

Write a piece of code, that multiplies all entries of the nested list from the previous exercise. Ensure that the same code also works for nested lists of larger dimensions, e.g. for
`bigger_nested_list = [[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12]]` .
Make sure it also works for
`weird_nested_list = [[1, 2, 3,], [4, 5, 6, 7], [8,]]` ?