Prof. Dr. Jörg Müller
Arthur Fleig

AI VIII – Serious Games

UNIVERSITÄT BAYREUTH

Exercises for **Programming, Data Analysis, and Deep Learning in Python** (SoSe 2021)

Exercise Sheet no. 2, *Deadline*: Monday, April 26, 10:15

Notes

- Always explain your solution.
- Handing in in groups of two is possible and desired. Only *one* submission per group! (All participants will get the credit).
- Unless stated otherwise, submissions should be uploaded on e-Learning. Name your file with the first and last names of all participants. Additionally, add these names as a comment at the beginning of your file.
  Example: Names of team mates: Erika Mustermann, Rudi Ratlos ⇒ File name: `ex1_Erika-Mustermann_Rudi-Ratlos.ipynb`
- Programming exercises should be submitted as `.py` or `.ipynb` files. Write the answers to text questions in programming tasks as a comment in the file.
- Exercises that do not have the "programming exercise" tag can be submitted as `.py`, `.ipynb`, `.txt`, or `.pdf` files.

Exercise 5 Troubleshooting (programming exercise) (6 points)

In a biochemistry research facility, the enzymes X, Y, and Z are mixed in various not necessarily integer mixing ratios. For safety reasons, each mixing ratio should first be checked by a computer program before it is entered into the machine. It is of utmost importance that the enzymes X and Y differ in quantity by less than $\varepsilon := 10^{-4}$. For cost reasons, an intern was assigned to write that program. The result is a Python code that you can find on e-Learning. Unfortunately, the internship has ended, but the program is not yet operational.

Find all (syntax) errors and make the program executable. Examine the code for logic errors and correct them. Justify every fix.

Hint: To find logic errors, try different values of X, Y, and Z and check whether the output of the program matches the desired output.

Exercise 6 Iterating over lists (programming exercise) (6 points)

The management of your company wishes to redesign workflows. There are four tasks A, B, C, and D. Task C cannot be completed until task A is completed. The management wants you to list all possible work sequences that take this into account. For example, (A, B, C, D) means that first task A, then B, then C, and finally D will be executed. An invalid sequence would be, e.g., (C, B, D, A).

a) Check whether the script provided on e-Learning really only outputs valid workflows. If it does not, adjust the script accordingly. Comment on your changes.

b) Extend the script such that all valid workflows are not only printed on screen but also written to a file called `valid_combinations.txt`.

Hint: The `print_list` function and creating all permutations with `itertools.permutations` are error-free. Reading the documentation[1] might help but is not required for this exercise.

Exercise 7 On lists, tuples, dictionaries, and iterators (6 points)

a) Briefly explain when to use a list, when to use a tuple, and when to use a dictionary.

b) Explain why a list cannot be used as a "key" in a dictionary in Python.

c) Name two advantages of a dictionary compared to a list.

d) Is there a difference in the output between `range(10)`, `range(0, 10)`, and `range(0, 10, 1)`? Explain each argument in `range(0, 10, 1)`.

Exercise 8 Python – Guess the number (programming exercise) (6 points)

Implement the following in Python:

1) Ask the user for an integer number between 1 and 100 and store the information in an integer variable `z`. (You do not have to verify whether a valid number was entered.) Multiply `z` by 9 and store the result in `z`.

2) Calculate the sum of digits of `z` and save the result in a variable.
Hint: To do this, you can cast `z` to a string. Then iterate over that string to get the respective digits.
(If you cannot solve this part, assume that the sum of digits is 9 in the following.)

3) Check whether the sum of digits is odd. If it is, do nothing. If it is not, divide the sum of digits by 2.

4) Square the result of step 3), then add 14 and finally, divide by 5. Print the final result to screen.

Try various integer numbers between 1 and 100. What stands out?

[1] `https://docs.python.org/3/tutorial/datastructures.html`