Assignment I

Programming with Python

Introduction

This is the first of two course assignments and here you will practice concepts from the first part of the course from lecture 01 to 06.

Grading of the Assignment

It is graded pass/fail and at least 50 % of the tasks in the assignments have to be passed in oder to pass the assigment. You may work in groups of up to three students and you should submit one assignment per group via email to vlcek@beyondsimulations.com before the start of Lecture 7 on scientific programming.

You may use online resources and generative AI (gAI) for assistance. However, you must not copy code from other groups, and you must include a short paragraph titled `How I used generative AI` that explains where and how you used gAI.

> How I used generative AI (template)
>
> - Provider(s) used (e.g., “OpenAI, Anthropic, Google”)
> - Whether you used a browser or an IDE to communicate with the provider
> - Your key prompts (1–3 lines), or a brief description of them
> - How you verified correctness (tests, docs checks, manual reasoning)

Note, you are fully responsible for the correctness and originality of submitted work, regardless of gAI use.

> Warning
>
> If you use gAI without including a paragraph on its use (use of gAI is typically easy to spot), the assignment will be marked as failed.

Use comments and docstrings to explain your code. Do not include gAI-generated meta-comments describing what gAI changed. Thus, keep comments focused on explaining the code itself.

> Warning
>
> If I find multiple instances of such gAI meta-comments, the assignment will be marked as failed. Keep your comments clean and relevant.

Use descriptive variable names and format your code consistently to maximize readability. Read the instructions carefully and follow them exactly. gAI may hallucinate additional features that are not required.

> ⚠ Warning
>
> Ensure the tasks of the assignment run without any errors or mistakes just by calling `uv run [your filename]`. If you used external packages, include your `pyproject.toml` so I can replicate your environments.

Submission format

- Submit a single ZIP named `a1_group-<number>_<lastname1-lastname2_lastname3>.zip`
- Include:
  - `assignment_1_task1.py`, `assignment_1_task2.py`, and so on
  - Any data files used in a `data/` folder (if used)
  - Any images in an `assets/` folder (if used)
- Email subject: `Intro to Python – Assignment 1`

> 💡 Submission checklist
>
> - ☐ Runs without errors from a clean environment by using `uv`
> - ☐ No unexplained gAI meta-comments, all comments explain code
> - ☐ "How I used generative AI" paragraph included (if gAI used)
> - ☐ Data/assets included and paths correct (if used)
> - ☐ `pyproject.toml` included (if external packages are used)
> - ☐ File naming and email subject follow the conventions

Tax refund for donations

In this excercise, you will create a program that is able to calculate the amount of tax a person can get back for a donation. First, research the rules for tax refunds for donations in Germany. Then, create a program that is able to calculate the amount of tax a person can get back for a donation. The program should ask the user for the amount of the donation and the yearly income and then calculate the amount of tax the user can get back. The program should then print the amount of tax to the console. Note, that the program should also handle wrong inputs from the user, e.g. if the user enters a negative donation amount or a non-numeric income.

```
# Tax refund for donations
# TODO: Create a program that is able to calculate the amount of tax a
person can get back for a donation.
# Your code here
```

> Tip
>
> Use the `input()` function to get user input in order to let the user interact with your program.

> Warning
>
> Do research yourself on how german taxes and donations work so you use the correct rules in your program!

Text based adventure game

In this excercise, you will create a text based adventure game. You are completely free in the choice of the story, but it must at least include 4 choices, 3 different endings, the choice to restart the game and yourself as a character of the game. Try to use the concepts of conditionals, loops, and functions to create a game that is both fun and interactive for the user. Note, that the program should also handle wrong inputs from the user, e.g. if the user enters a choice that is not available.

```
# Test based adventure game
# TODO: Create a text based adventure game with at least 3 choices, 2
different endings and the choice to restart the game.
# Your code here
```

Hangman

In this excercise, you will create a program that is able to play the game Hangman. The program should ask the game master for a secret word. Afterward, it should ask the player to guess a letter and then check if the letter is in the word. The program should then print the word with the guessed letters and the number of tries left. The program should also handle wrong inputs from the user, e.g. if the user enters a non-letter or a letter that is already guessed.

```
# Hangman
# TODO: Create a program that is able to play the game Hangman with a game
master and one player.
# Your code here
```

Caesar Cipher

In this excercise, you will create a program that is able to encrypt and decrypt messages using the Caesar Cipher. The program should ask the user for a message and a key and whether to encrypt or decrypt the message. The program should then encrypt the message by shifting each letter of the message by the key or decrypt the message by shifting each letter of the message back by the key. The program should then print the encrypted or decrypted message and asks the user whether to continue with the next

message or to quit the program. Note, that you only need to encrypt or decrypt letters, keep other characters unchanged.

```
# TODO: Create a programm able to encrypt and decrypt messages using the
Caesar Cipher.
# Your code here
```