Assignment II

Programming with Python

Introduction

This is the second of the two assignments, where you will practice the concepts you learned in the second part of the course.

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 `a2_group-<number>_<lastname1-lastname2_lastname3>.zip`
- Include:
  - `assignment_2_task1.py`, `assignment_2_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 2`

> 💡 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

Redact secret information

In this excercise, you will create a program that is able to redact secret information in a text. The program should be able to redact the following: zip codes, names, email addresses and phone numbers. The program should ask the user for a filename and then read the file and redact the secret information. The program should then print the redacted text to the console and write it to a new file called `redacted.txt`. You can find a file with secret information in the git repository under `assignments/secret-text.txt`.

```
# Secret information redactor
# TODO: Create a program that is able to redact secret information in a
text.
# YOUR CODE HERE
```

Dice roll simulator

In this excercise, you will create a program that is able to simulate dice rolls and visualizes their distribution. The program should ask the user for the number of sides on the dice and the number of dices to roll. Then, the program should simulate the dice rolls 10000 times and visualize the distribution of the dice rolls using a histogram.

```
# Dice roll simulator
# TODO: Create a program that is able to simulate a dice roll.
# YOUR CODE HERE
```

Future Self Predictor

In this exercise, you will create an interactive program that predicts a users future based on his or her answers. The program should ask 5 questions about users where user can either add the answers in the terminal freely or choose from a number of options. It should then generate 3 entertaining predictions, and save them to a separate file with the user name as filename. The predictions should be a 5-year, 10-year, and 30-year prediction of where the user is going to be in life. Note, the program should be able to handle user input errors gracefully.

```
# Future Self Predictor
# TODO: Create a program that is able to predict the future.
# YOUR CODE HERE
```

Dashboards

In this exercise, you will create a program that visualizes a data set of your choice in an interactive dashboard. For this excercise, you can choose any data set of your interest. The data should be visualized in a dashboard with at least two plots. You can use one of the dashboard libraries we discussed in the lecture.

> **i** Note
>
> The dashboard should be programmed in a separate file I can call for evaluation.

```
# Dashboards
# TODO: Create a program that visualizes a data set of your choice in a
dashboard.
# YOUR CODE HERE
```