Exercise #02

IT University of Copenhagen (ITU)
Data Mining KSD (DAMIN)
(Autumn 2025)

02 September 2025

Introduction This exercise list will provide you with hands-on experience in Python programming and fundamental data analysis, reinforcing the concepts covered during the lecture. The tasks are approachable for beginners while offering challenges that require critical thinking and problem-solving skills. Feel free to implement your solution using Python Scripts or Jupyter Notebooks. The learning objectives for this exercise encompass:

- Write simple Python programs that perform basic input/output operations.
- Understand and apply fundamental Python syntax and constructs, including variables, data types, and control flow.
- Manipulate and manage lists, tuples, and dictionaries to store and retrieve data.
- Load and manipulate data, including reading data from files, handling missing data, and performing fundamental data transformations.
- Apply basic data analysis techniques, such as calculating summary statistics and filtering data based on conditions.

Exercise 02.01. *Simple Calculator* (15-20 minutes) – Write a Python program that takes two numbers as input from the user and performs basic arithmetic operations (addition, subtraction, multiplication, and division). – *Homework Exercise*

- **Instructions:**
  - Prompt the user to enter two numbers.
  - Perform the four basic arithmetic operations.
  - Display the results.
- **Example Output:**
  - If the user enters 5 and 3, the program should output:

```
5 + 3 = 8
5 - 3 = 2
5 * 3 = 15
5 / 3 = 1.67
```

Exercise 02.02. *Grade Evaluation* (15-20 minutes) – Create a Python program that evaluates students' grades based on their scores. Feel free to use the grading scale from your preference (e.g., see grading systems by country).

- **Instructions:**
  - Prompt the user to enter a score between 0 and 100.
  - Use conditional statements to assign a grade (A, B, C, D, F) based on the score.
- **Example Output:**
  - If the user enters 85, according to the academic grading in the United States, the program should output:

    ```
    Grade: B
    ```

Exercise 02.03. *Manage a List of Students* (15-20 minutes) – Create a Python program that manages a list of student names. The program should allow users to add new names, remove existing ones, and display all ones in the list. *Optionally*, you can save the list to a file and load it when the program starts.

- **Instructions:**
  - Initialize an empty list to store student names.
  - Provide options for the user to add a new name, remove a name, or display all names in the list.
  - Use a loop to offer these options until the user exits repeatedly.
- **Menu Options:**
  - This is an example of the menu options you can provide:

    ```
    1. Add a name
    2. Remove a name
    3. Display all names
    4. Exit
    ```

Exercise 02.04. *Simple Contact List with Dictionaries* (15-20 minutes) – Develop a Python program that simulates a simple contact list using dictionaries. The program should allow users to add new contacts, remove existing contacts, search for a contact by name, and display all contacts. *Optionally*, you can save the dictionary to a text file and load it when the program starts. – *Homework Exercise*

- **Instructions:**
  - Use a dictionary to store contacts; the keys are the contacts' names, and the values are their phone numbers.
  - Provide options for the user to add a new contact, remove a contact, or search for one.
  - Display the contact list in a readable format.
- **Example Output:**
  - If the user wants to display all contacts, the program should output:

```
Contacts:
Alice: +45 12 34 56 78
Bob: +45 98 76 54 32
Charlie: +45 23 45 67 89
```

Exercise 02.05. *Analyze Student Grades* (20-30 minutes) – Work with a dataset of student grades to perform fundamental data analysis. Load the dataset into a Pandas `DataFrame` and calculate each student's basic statistics (e.g., average grade, standard deviation, minimum, maximum). Identify students who scored above a certain threshold and handle any missing data in the dataset.

- **Instructions:**
  - Load a CSV file named `student_grades.csv` containing columns for student names and grades for distinct subjects into a Pandas `DataFrame`.
  - Display the first few rows of the data.
  - Calculate the average grade for each student.
  - Identify and print the students who scored above a certain threshold (e.g., 85).
  - Handle any missing data in the dataset.
- **Example Output:**
  - If the dataset contains the following information:

Student	**Math**	**Science**
Alice	90	92
Bob	75	92
Charlie	85	78

– The program should output:

```
Average Grades
Alice: 91
Bob: 83.5
Charlie: 81.5

Students with Grades Above 85
Alice: 90 (Math), 92 (Science)
Charlie: 85 (Math)
Bob: 92 (Science)
```

Exercise 02.06. *Sales Data Analysis* (20-30 minutes) – Analyze a small sales information dataset. Load the dataset into a Pandas `DataFrame` and calculate the total sales for each product. Determine the day with the highest sales and generate summary statistics for the dataset.

- **Instructions:**
  - Load a CSV file named `sales_data.csv` containing columns for the date, product name, quantity sold, and sales amount into a Pandas `DataFrame`.
  - Display the first few rows of the data.
  - Calculate the total sales for each product.
  - Determine the day with the highest sales.
- **Perform a more detailed analysis:**
  - Generate summary statistics (mean, median, mode).
  - Write a brief summary (as comments in the code) explaining the findings.