

Lecture 9 - Exercises

Collections and Nested Data: Guided, Laboratory, and Individual practice

Exercise Set Structure

- **Guided:** Course Registration Summary - tuple unpacking, sets, dictionaries
- **Lab 1:** Club Membership Comparison - set operations and inclusion checks
- **Lab 2:** Weekly Sales Table - nested lists, rows, columns, nested loops
- **Individual 1:** Product Catalogue and Price Report - dictionaries and sorting
- **Individual 2:** Student Results Analyzer - list of dictionaries and sets
- **Individual 3:** Multi-Course Grade Report - nested dictionaries and lists
- The three Individual exercises together simulate an exam-style practice block.

Guided Exercise: Course Registration Summary - Problem

- A course office receives a list of registrations. Each registration is represented by a pair containing the student name and the course name.
- The same student may appear more than once because they can register for different courses. The program must summarize the registrations by identifying unique students and counting enrolments per course.
- Input/data format:
 - `registrations = [("Alice", "Programming"), ("Bob", "Mathematics"), ...]`
 - Each element is a tuple: (student, course).

Guided Exercise: Course Registration Summary - Tasks

- Tasks:
 - iterate over the list of registration tuples;
 - use *tuple unpacking* to obtain student and course;
 - build a **set** containing all **distinct student names**;
 - build a **dictionary** that **counts registrations for each course**;
 - print the unique students in alphabetical order;
 - print each course and its number of registrations;
 - identify and print the course with the **largest number of registrations**.
- Lecture 9 hints:
 - Use a set when only uniqueness matters.
 - Use a dictionary when a key identifies a value.
 - Tuple unpacking makes pairs easier to read.

Use functions to keep it readable and well structured

Laboratory 1: Club Membership Comparison - Problem

- Two university clubs store the names of their members. Some students belong to both clubs, while others belong to only one of them.
- The program must compare the two member groups and produce a short report describing overlaps and differences. This is a direct application of mathematical set operations.
- Input/data format:
 - `coding_club` = {"Alice", "Bob", "David", "Emma"}
 - `robotics_club` = {"Bob", "Carol", "David", "Frank"}
 - Both collections are sets of names.

Laboratory 1: Club Membership Comparison - Tasks

- Tasks:
 - compute students who are in **at least** one club (all the name);
 - compute students who are in **both** clubs;
 - compute students who are **only** in the Coding Club;
 - compute students who are **only** in the Robotics Club;
 - compute students who are in **exactly one** of the two clubs;
 - display the **total** number of distinct students involved;
 - check whether all members of one club are also members of the other;
 - print set results in alphabetical order.
- Lecture 9 hints:
 - Match “at least one”, “both”, “only”, and “exactly one” to set operations
 - Use `.subset/ .superset` checks for inclusion questions
 - Sort only for display, not to change the set itself.

Use functions to keep it readable and well structured

Laboratory 2: Weekly Sales Table - Problem

- A shop records how many units of each product are sold over five days of a week (Monday - Friday). The data are stored as a table: each row represents one product and each column represents one day.
- The program must analyze the table by product and by day. This requires reasoning about rows, columns, and nested loops.
- Input/data format:
 - `sales = [[12, 15, 9, 18, 20], [5, 7, 6, 4, 8], [20, 18, 25, 22, 19]]`
 - `products = ["Notebook", "Pen", "Backpack"]`
 - Rows correspond to products; columns correspond to days.

Laboratory 2: Weekly Sales Table - Tasks

- Tasks:
 - print the table in a readable form;
 - calculate the **weekly total for each product**;
 - build a list containing these product totals;
 - identify and print the product with the **highest weekly total**;
 - calculate the **total sales for each day**;
 - identify and print the day with the highest total sales;
 - calculate the overall total number of units sold.
- Lecture 9 hints:
 - Use the outer loop for rows/products
 - Use fixed column indexes when computing daily totals
 - Keep row meaning and column meaning separate.

Use functions to keep it readable and well structured

Individual 1: Product Catalogue and Price Report - Problem

- A small shop stores its catalogue as a dictionary in which each product name is associated with its price. The user wants to search for a product and also inspect products within a chosen price interval.
- The program must use dictionary lookup, safe key handling, iteration, and sorting by both keys and values.
- Input/data format:
 - `prices = {"Notebook": 4.50, "Pen": 1.20, "Backpack": 32.00, ...}`
 - User input: minimum price, maximum price, product to search.

Individual 1: Product Catalogue and Price Report - Tasks

- Tasks:
 - read and validate the **minimum** and **maximum** price;
 - read the **product name** to search;
 - report the product price if it exists, otherwise show a suitable message;
 - print all products whose price is inside the interval;
 - print the catalogue sorted alphabetically by product name;
 - print the catalogue from most expensive to least expensive;
 - show the cheapest and most expensive products;
 - calculate and display the average price.
- Lecture 9 hints:
 - A dictionary is appropriate because product names identify prices
 - Sorted iteration by key is different from sorted iteration by value
 - A dictionary method can be used as the sorting key.

Use functions to keep it readable and well structured

Individual 2: Student Results Analyzer - Problem

- A lecturer stores exam results as a list of student records. Each record is a dictionary with the same fields: student ID, name, and grade.
- The program must analyze the class, search by student ID, and use a set to identify distinct grades. It should be structured into small functions for search and analysis.
- Input/data format:
 - `students = [{"id": 1042, "name": "Alice", "grade": 18}, ...]`
 - User input: one student ID to search.

Individual 2: Student Results Analyzer - Tasks

- Tasks:
 - calculate the **class average**;
 - show the student with the **highest grade** and the one with the **lowest grade**;
 - build a new list containing only **passing students**, using 10 as passing threshold;
 - print passing students in **alphabetical order**;
 - ask the user for a student ID and display the matching record if present;
 - count how many distinct grades appear in the class;
 - display the distinct grades in increasing order;
 - define at least two functions, separating search and analysis tasks.
- Lecture 9 hints:
 - A list of dictionaries keeps each student record together
 - A set is useful when only distinct values matter
 - Use functions to separate lookup from report computations.

Use functions to keep it readable and well structured

Individual 3: Multi-Course Grade Report - Problem

- A course coordinator stores the results of several students across multiple courses. The data are organized as a nested dictionary: the first key selects the **student**, the second key selects the **course**, and each course contains a **list of grades**.
- The program must navigate this nested structure to compute student-level and course-level summaries.
- Input/data format:
 - `results = {"Alice": {"Programming": [18, 17, 20], "Mathematics": [16, 19, 18]}, ...}`
 - User input: one student name to inspect.

This is the most complete exam-style exercise of the set

Individual 3: Multi-Course Grade Report - Tasks

- Tasks:
 - ask for a student name and verify that it exists;
 - for that student, print the average grade for each course;
 - calculate the student overall average;
 - identify the course with the best average for that student;
 - calculate the average of each course considering all students;
 - find the student with the highest overall average;
 - build a set containing all distinct grades in the dataset;
 - use functions for at least: average of a list, overall student average, and best student search.
- Lecture 9 hints:
 - Follow the structure level by level: student → course → grades
 - When processing nested data, decide which loop corresponds to each level
 - Keep small computations in functions before building the full report.

Use functions to keep it readable and well structured