Week 3 — Putting the Station's Data in Order

Course: Scientific Programming with Python
Timebox: Tutorial 75 min · Exercise 5 ≤ 30 min · Exercises 6–10 = home tasks for ~1 week

Narrative setup (read first)

Your rooftop weather station now streams many small facts: minute-by-minute readings, crew rosters, thresholds, and notes. This week you'll **organize** those facts so they're easy to look up, combine, and check—like a field notebook that never loses a page.

Exercise 1 — Packing a Reading

Type: Scientific · **Estimated time:** 6–8 min

A technician dictates a single snapshot: (day index, hour, temperature °C). Capture the three pieces together so they travel as one and print them back neatly.

What you must produce. One script that builds one **packed record** from three inputs and prints: `day=.. hour=.. tempC=...`

Inputs/Outputs.
• day ∈ `[1,31]`, hour ∈ `[0,23]`, temp ∈ `[-40,60]`.
• Print on one line; temp with **1 dp**.

Reflection. When do you keep related values "stuck together," and why?
Why this matters. Measurements make sense only with their context.

Hint: use tuple to store day, hour, temp.

Exercise 2 — Minute Bucket

Type: Scientific · **Estimated time:** 7–9 min

A burst of **5** wind-speed values arrives for the same minute. Store them **in order received** and then show the first, last, and how many arrived.

What you must produce. One script that builds a small **ordered bucket** from five floats and prints: `first=.. last=.. count=5`.

Inputs/Outputs.
• Five speeds in km/h (each `0–200`).
• Print first and last with **1 dp**, and `count`.

Reflection. Why preserve arrival order here?
Why this matters. Later smoothing/median needs chronological order.

Hint: Which data structure is appropriate: list, tuple, dictionary, or set?

Exercise 3 — Unique Terms from a Note

Type: General · **Estimated time:** 6–8 min

The crew writes a short note like `light rain alarm test alarm`. Extract the **distinct** words used in the note and show how many unique terms there are.

What you must produce. One script that takes a short text (≤100 chars) and prints `unique_terms=N`.

Inputs/Outputs.
• Input: one line of text; split on spaces; case-sensitive.
• Output: exact count of **distinct** tokens.

Reflection. When do you care about uniqueness over counts?
Why this matters. Quick de-dup helps spot repeated flags like “alarm”.

Hint: Which data structure is appropriate: list, tuple, dictionary, or set?

Exercise 4 — Little Lookup

Type: Scientific · **Estimated time:** 10–12 min

The station has three named thresholds: `t_ok_low=18`, `t_ok_high=26`, `hum_low=20`. Prepare a **name→value** look-up so other parts of the sheet can fetch by name and print the three values in one aligned report.

What you must produce. One script that creates a small **name→value** table and prints lines like `t_ok_low: 18`.

Inputs/Outputs.
• Exactly the three names above with integer values.
• Print in any order, one per line.

Reflection. What makes name-based access safer than “remembering positions”?
Why this matters. Readable configuration beats magic numbers.

Hint: Which data structure is appropriate: list, tuple, dictionary, or set?

Exercise 5 — Mini “Minute Record”

Type: Scientific · **Estimated time:** ≤ 30 min

For one minute, you receive: an **ID** (text), a **(day, hour)** pair, and a tiny bundle of numbers (**temperature °C**, **humidity %**). Produce a compact, single-line summary that includes: the packed time, the numbers **in arrival order**, a **distinct-term count** from a short note, and whether the temperature falls inside your earlier threshold names.

What you must produce. One script that reads: ID, day, hour, temp, hum, and a note; then prints one line:
`ID | time=(day,hour) | values=[temp,hum] | distinct_note_terms=N | in_temp_range=True/False`

Inputs/Outputs.
• day ∈ `[1,31]`, hour ∈ `[0,23]`, temp ∈ `[-40,60]`, hum ∈ `[0,100]`.
• Thresholds from Exercise 4.
• Treat words by splitting on spaces; case-sensitive.

Reflection. How did separating “structure” from “values” make the line easier to build?
Why this matters. Field summaries are trustworthy when structure is consistent.

Exercise 6 — Minute-to-Hour Box (home task)

Type: Scientific · **Estimated time:** Home task

Over an hour, you collect **60** temperature snapshots. Organize them so you can report: the **first three** readings, the **last three**, and the hour’s size.

What you must produce. One script/cell that takes 60 numbers and prints 7 lines: `first1..first3`, `last1..last3`, and `count=60`.

Inputs/Outputs.
• Each value in `[-40,60]`; exactly **60** inputs.
• Print each chosen reading with **1 dp**.

Reflection. Why choose a fixed-size container here?
Why this matters. Fixed windows are standard in QC and smoothing.

Exercise 7 — Crew Phone Book (home task)

Type: General · **Estimated time:** Home task

Build a small “name→phone” roster for **4** crew members and print the contact for a queried name; if the name isn’t present, print `not found`.

What you must produce. One script/cell that first builds the roster, then reads a queried name and prints either the phone number or `not found`.

Inputs/Outputs.
• Names are case-sensitive; phone is a short string like `+49-421-123456`.

Reflection. Why is key-based access more direct than scanning a list?
Why this matters. Lookups are everywhere—from configs to contacts.

Exercise 8 — De-dup the Parts Order (home task)

Type: General · **Estimated time:** Home task

A vendor list contains repeated item codes for gaskets and bolts. From a line of space-separated codes, report the **distinct** codes and how many distinct items to order.

What you must produce. One script/cell that reads a single line of codes like `GA12 GA12 B07 GA12 B07 X1` and prints: `distinct=N` and a second line with the codes in any order, separated by spaces.

Inputs/Outputs.
• Codes are non-empty words without spaces; case-sensitive.
• Order of the output codes may be arbitrary.

Reflection. What are the trade-offs of losing multiplicity information?
Why this matters. Prevents double-ordering and reduces waste.

Exercise 9 — Quick Indices (home task)

Type: General · **Estimated time:** Home task

Prompt. The station team wants a small “index” telling where a crew member appears in the day’s duty list. Given a position-ordered roster and a name to find, print the position(s) where it appears (0-based).

What you must produce. One script/cell that reads a roster line like `Asha Ben Asha Chen` and a query name, then prints a line `positions: i j ...` (empty list if not present).

Inputs/Outputs.
• Names are case-sensitive; report positions **smallest→largest**.

Reflection. How did order help you answer the question?
Why this matters. Indices are the bridge between order and location.

Exercise 10 — Sensor Glossary (home task)

Type: Scientific · **Estimated time:** Home task

Prompt. Build a small "term→explanation" glossary for **5** station terms (e.g., `dewpoint`, `gust`, `hPa`, `calibration`, `checksum`). Allow one query term; print the definition if present, or `unknown`.

What you must produce. One script/cell that constructs the glossary, asks for a query, and prints exactly one line with either the definition or `unknown`.

Inputs/Outputs.
• Free-text definitions ≤ 80 chars.

Reflection. Why separate raw terms from their explanations?
Why this matters. Shared vocabulary reduces misinterpretation.