Week 1 — Booting Up the Campus Weather Station

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

Exercise Sheet

Your team is bringing a low-cost **campus weather station** online on the physics building roof. Today you'll prepare small data snippets the station will use: clear identifiers, unit conversions, clean readouts, and sanity checks. Keep outputs tidy because your teammates will copy them into a shared dashboard.

Exercise 1 — Hello, Station!

Type: General · **Estimated time:** 5–7 min

Create a short station ID label combining campus code, building code, and floor, then show a friendly greeting for the field log.

What you must produce. One Python script that **creates the label** and **prints** a single-line greeting including it.

Inputs/Outputs.
• Inputs: campus code (e.g., `CU`), building code (e.g., `PHY`), floor (e.g., `2`).
• Output: one line like `CU-PHY-2 online`.
• Keep output to **one** line.

Reflection. What naming choices keep IDs readable later?

Why this matters. Clear IDs reduce confusion across devices.

Exercise 2 — Units on the Move

Type: Scientific · **Estimated time:** 7–10 min

A colleague recorded wind speed in **km/h**, but the station expects **m/s**. Prepare a conversion readout for today's gust – a sudden strong rush of wind.

What you must produce. A Python script that **accepts a wind speed value** and **prints the equivalent in m/s** with **two decimals**.

Inputs/Outputs.
• Input: a positive number in km/h within `[0, 200]`.
• Output: a single line containing the m/s value (2 dp) with SI unit label.

Reflection. Where might rounding be risky in science outputs?

Why this matters. Consistent units prevent analysis mistakes.

Exercise 3 — Precedence Puzzles [ignore this one for now]

Type: General · **Estimated time:** 8–10 min

The firmware[1] computes a "calmness score" from numbers `a`, `b`, `c`. Reproduce its exact result for given values and compare alternative groupings.

What you must produce. A Python script that **prints** results of **three** expressions using the same `a`, `b`, `c`, making operator **precedence** explicit with parentheses in some variants.

Inputs/Outputs.
• Given: `a = 6`, `b = 2`, `c = 3`.
• Output: 3 labelled lines; choose equivalent-looking expressions; avoid division by zero.

Reflection. Which expression surprised you, and why?

Why this matters. Precedence errors cause silent, wrong results.

Exercise 4 — Read, Convert, Report

Type: Scientific · **Estimated time:** 8–10 min

The technician sends a **radius** in meters to compute the **cross-sectional area** of a round vent. Provide a neat, aligned readout.

What you must produce. A cell that **asks for a radius**, computes area using `π ≈ 3.1416`, and **prints** radius and area on one line with aligned fields and **two decimals**.

Inputs/Outputs.
• Input: radius `r` in `[0.0, 5.0]` meters.
• Output: `r=… m  area=… m^2` with aligned columns (2 dp).
• Ignore negative inputs for now.

Reflection. How does formatted output improve readability?

Why this matters. Logs must be scannable at a glance.

---

[1] Software permanently programmed into hardware to control its basic functions

Exercise 5 — Mini Logger (≤ 30 min)

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

Create a **one-line status log** for later CSV append: station ID, air temperature (°C), humidity (%), and a **boolean** that indicates if temperature lies in calibration range `[18, 26]`.

What you must produce. A cell that **collects three values** (ID as text, temperature as number, humidity as number) and **prints**:
`ID, temp_C, humidity_pct, in_range`

Inputs/Outputs.
• Inputs: ID like `CU-PHY-2`, temp `[-40, 60] °C`, humidity `[0, 100] %`.
• Output: one CSV line; `in_range` must be `True` or `False`.
• Temperature shows **one decimal**; humidity shows **no decimals**.

Reflection. Where could a type mismatch sneak in?

Why this matters. Clean, typed fields enable reliable logs.

Exercise 6 — Seconds to Clock (home task)

Type: General · **Estimated time:** 30–45 min (home)

The station records uptime in **seconds**. Produce a human-friendly `HH:MM:SS` readout.

What you must produce. A cell that **reads a non-negative integer of seconds** and **prints** hours, minutes, seconds in two-digit fields.

Inputs/Outputs.
• Input: `t` in `[0, 1_000_000]`.
• Output: one line `HH:MM:SS`; hours may exceed 24 (no days).

Reflection. Why do integer division and remainder pair well here?

Why this matters. Time formatting appears in many logs.

Exercise 7 — Range Flags (home task)

Type: Scientific · **Estimated time:** 30–45 min (home)

A packet carries three sensor values: pressure (hPa), temperature (°C), humidity (%). Generate **three booleans** that tell whether each value lies inside its recommended range—**without** branching.

What you must produce. A cell that **reads three values** and **prints** three labelled booleans.

Inputs/Outputs.
• Ranges: pressure `[950, 1050]`, temp `[-10, 40]`, humidity `[20, 80]`.
• Output: lines like `pressure_ok: True`.

Reflection. How do combined comparisons express ranges succinctly?

Why this matters. Quick checks catch bad data at the source.

Exercise 8 — Bill Splitter (home task)

Type: General · **Estimated time:** 30–45 min (home)

After the installation, the team orders snacks and the total bill must be split among all members. However, everyone can only contribute whole euros (no coins). Each person pays the same number of whole euros. Because of this, there may be some cents left over that cannot be evenly split in whole euros. Your task is to compute (1) how many whole euros each person should pay and (2) how many cents are left unpaid in total, which the group must handle separately (e.g., with coins or as a tip).

What you must produce. A cell that **reads total cost in euros** and **team size**, computes **whole-euro share** per person and **remaining cents**.

Inputs/Outputs.
• Inputs: cost `[0.00, 200.00]`, team size `[1, 12]`.
• Outputs: two lines—`share_eur` (whole euros), `leftover_cents` (`0 or more`).

Reflection. Why is remainder helpful in everyday arithmetic?

Why this matters. Integer math powers budgeting tools.

Exercise 9 — Simple Circuits Readout (home task)

Type: Scientific · **Estimated time:** 30–45 min (home)

For a calibration heater, compute **current** and **power** from supplied **voltage** and **resistance**; include neat units.

What you must produce. A cell that **reads** `V` (volts) and `R` (ohms) and **prints** `I` (amps) and `P` (watts) with **two decimals** and aligned labels.

Inputs/Outputs.
• Ranges: `V ∈ [0.0, 240.0]`, `R ∈ (0.0, 200.0]`.
• Output: two labelled lines with aligned fields; avoid division by zero.

Reflection. How does formatting improve the readability of units?

Why this matters. Quick electrical checks prevent damage.

Exercise 10 — Packet Signature (home task)

Type: General · **Estimated time:** 40–60 min (home)

Create a **compact signature string** combining a day index, station ID, and a small checksum from two small integers (e.g., `(a*b) % 97`).

What you must produce. A cell that **reads** day (1–31), `ID` text, and two small ints `a`, `b`, then **prints**: `DAY-ID-CHECK`, where `CHECK` is a two-digit zero-padded number.

Inputs/Outputs.
• Day `[1, 31]`, small integers `[0, 999]`.
• Output example: `14-CU-PHY-2-05`.

Reflection. Why are fixed-width, zero-padded numbers handy in logs?

Why this matters. Compact tags keep filenames and rows consistent.

Week-1 Boundaries (for all tasks). Use only what's already taught: variables; numbers/strings/booleans/None; arithmetic, assignment, relational & logical operators; console input/output; **f**"..." formatting; numeric conversions. Avoid branching, loops, functions, lists, files, classes, and external libraries.