# Question Paper Set A13

AI-Assisted Coding Exam — Python (1 hour)  
- You may either (a) write the exact prompt to have your AI assistant generate code, or (b) write the code yourself. If you use AI, submit the prompt and the final code you executed.  
- Use VS Code Copilot / Gemini / CursorAI. Credit your prompts.  
- Prefer Python standard library; write clear, tested, well-documented code.  
- Each subgroup (A–O) has two tasks. Attempt all. Medium complexity, use-case based.  
- Syllabus pillars:  
 A–C: AI completion (classes, loops, conditionals)  
 D–E: Debugging with AI  
 F–G: TDD with AI  
 H–I: Documentation & Code Review  
 J–K: Code Quality & Performance  
 L–M: Files/CSV & Regex  
 N–O: Algorithms & Data Structures

Deliverables for each question  
1) If using AI: the exact prompt you issued. If manual: note 'manual' and a brief design reason.  
2) solution.py  
3) tests.py (unittest/pytest; write tests first for TDD items)  
4) Docstrings & inline comments (AI-assisted allowed)  
5) Short README.md (approach, assumptions, complexity, run tests)  
6) For debugging/refactor: brief before/after note

Question Paper Set A13

AI-Assisted Coding Exam — Python (1 hour)

- You may either (a) write the exact prompt to have your AI assistant generate code, or (b) write the code yourself. If you use AI, submit the prompt and the final code you executed.

- Use VS Code Copilot / Gemini / CursorAI. Credit your prompts.

- Prefer Python standard library; write clear, tested, well-documented code.

- Each subgroup (A–O) has two tasks. Attempt all. Medium complexity, use-case based.

- Syllabus pillars:

A–C: AI completion (classes, loops, conditionals)

D–E: Debugging with AI

F–G: TDD with AI

H–I: Documentation & Code Review

J–K: Code Quality & Performance

L–M: Files/CSV & Regex

N–O: Algorithms & Data Structures

Deliverables for each question

1) If using AI: the exact prompt you issued. If manual: note 'manual' and a brief design reason.

2) solution.py

3) tests.py (unittest/pytest; write tests first for TDD items)

4) Docstrings & inline comments (AI-assisted allowed)

5) Short README.md (approach, assumptions, complexity, run tests)

6) For debugging/refactor: brief before/after note

## Subgroup J

### J.1 — [S13J3] Normalize sensor readings

**Scenario (agritech):**  
**Context:** Sensor data logs in agritech are stored as CSV-like text. Each line contains sensor\_id, timestamp, value. For downstream ML, values must be normalized.

**Your Task:**  
Parse the text into structured data, compute **z-score normalization** for each sensor’s values independently (per sensor\_id), and return results as a dict mapping sensor\_id → list of (timestamp, z\_value).

**Data & Edge Cases:**

* Input text may have blank lines.
* Each sensor\_id should be normalized independently.
* If a sensor has only one value, return z=0 for all its rows.

**AI Assistance Expectation:**  
Ask AI to suggest pandas/numpy vs pure Python approaches, discuss numerical stability, and test edge cases.

**Constraints & Notes:**

* Output as dict[str, list[tuple[str, float]]].
* Round z-scores to 3 decimals for consistency.

**Sample Input**

s1,2025-01-01T10:00,10

s1,2025-01-01T11:00,20

s2,2025-01-01T10:30,100

s2,2025-01-01T11:30,100

**Sample Output**

{

's1': [('2025-01-01T10:00', -1.0), ('2025-01-01T11:00', 1.0)],

's2': [('2025-01-01T10:30', 0.0), ('2025-01-01T11:30', 0.0)]

}

**Acceptance Criteria:**

* Correct per-sensor z-score calculation.
* Handles one-value case with zeros.
* Ignores blank lines safely.
* A screen shot of a computer program

  AI-generated content may be incorrect.

A screen shot of a computer program

AI-generated content may be incorrect.

A screen shot of a computer program

AI-generated content may be incorrect.

Output:A screenshot of a computer program

AI-generated content may be incorrect.

### J.2 — [S13J4] Detect overlapping irrigation schedules

**Scenario (agritech):**  
**Context:** Smart irrigation systems schedule watering by time slots per field. Overlaps between schedules can cause overwatering or pressure issues.

**Your Task:**  
Given a list of schedules (field\_id, start\_time, end\_time in ISO format), detect **pairs of fields with overlapping watering times**.

**Data & Edge Cases:**

* Input is a list of dicts.
* Times are inclusive at start, exclusive at end.
* Return list of tuples (fieldA, fieldB) where overlap occurs.
* If no overlaps, return empty list.
* Ensure each pair is unique (no duplicates, order sorted lexicographically).

**AI Assistance Expectation:**  
Ask AI to propose interval overlap detection strategies (sorting by start, sweep-line, or O(n²) check depending on data size).

**Constraints & Notes:**

* Return list[tuple[str,str]].
* Assume all times in same timezone (no conversion needed).

**Sample Input**

[

{'field': 'F1', 'start': '2025-01-01T08:00', 'end': '2025-01-01T10:00'},

{'field': 'F2', 'start': '2025-01-01T09:30', 'end': '2025-01-01T11:00'},

{'field': 'F3', 'start': '2025-01-01T11:00', 'end': '2025-01-01T12:00'}

]

**Sample Output**

[('F1', 'F2')]

**Acceptance Criteria:**

* Correct detection of overlaps.
* Unique, sorted pairs.
* Works for multiple overlaps or none.

A screen shot of a computer program

AI-generated content may be incorrect.

A screen shot of a computer program

AI-generated content may be incorrect.

Output: