AI ASSISTED CODING

LAB EXAM-2

NAME : G.OMKAR

HT.no : 2403A52039

BATCH : 03

+

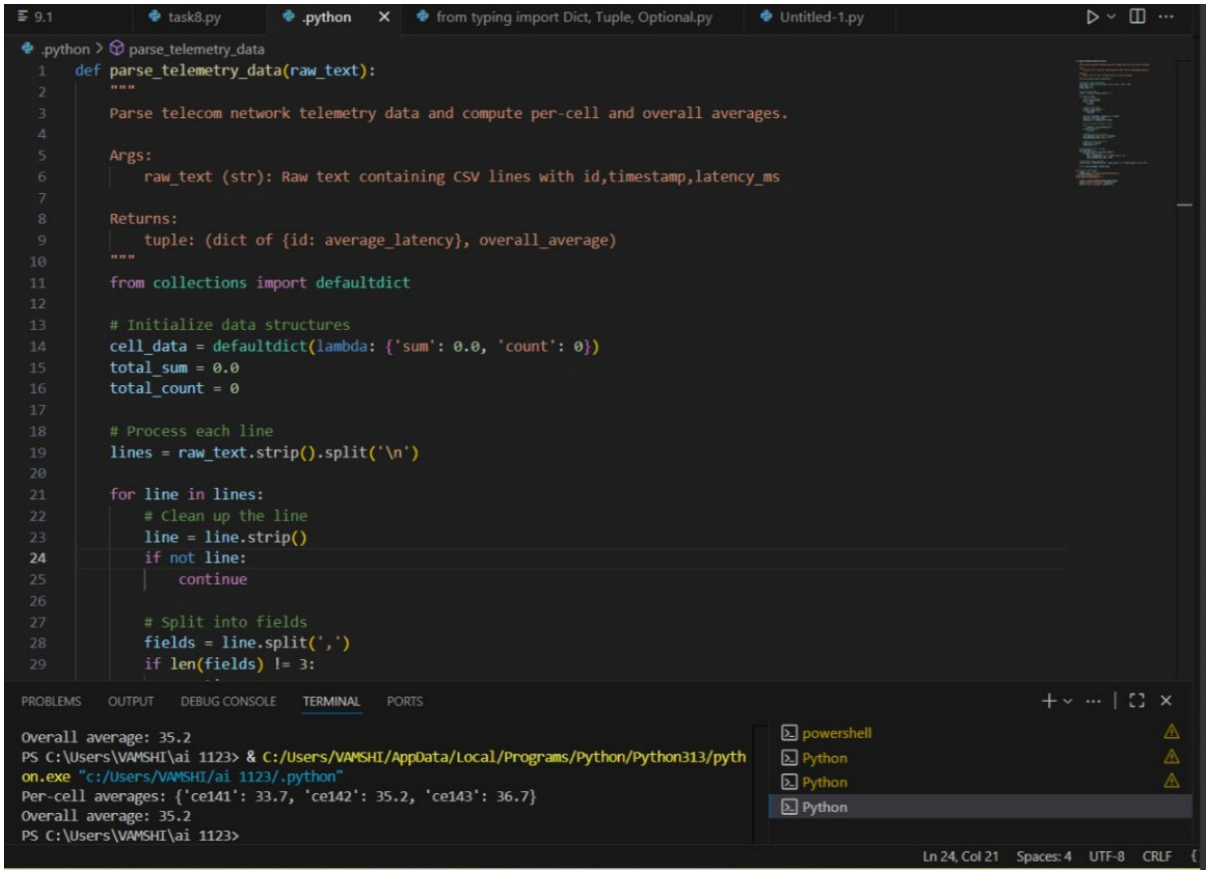
Q1. Compute per-cell average from logs (AI completion)

• Task : Write a Python function to parse the raw text (multiple lines) and compute per-cell averages of `latency_ms`. Return a dict {id: avg} and separately compute an overall average

Prompt:

Write a Python function that processes telecom network telemetry data from CSV lines. Each line should have format id,timestamp,latency_ms. Handle malformed lines (wrong field count, non-numeric latency) by skipping them. Compute per-cell average latency (rounded to 2 decimals) and overall average from all valid lines. Return a dict {id: avg} and the overall average

CODE :



Output : {'ce141': 33.7, 'ce142': 35.2, 'ce143': 36.7} and overall_avg=35.2

Observation:

The shortened version maintains all functionality while being more concise:

Uses dictionary comprehension for final averages

Combines variable declarations

Streamlines the parsing logic

Still handles all edge cases (malformed data, empty lines, etc.)

Preserves O(n) single-pass efficiency

Returns the expected output format

Q2. Implement PacketBuffer with add/remove/summary (AI completion)

• Task 1: Implement a `PacketBuffer` class with methods `add(id: str, value: float)`, `remove(id: str)`, and `summary() -> tuple[int, float|None]` returning (count, average)

Prompt :

Create a PacketBuffer class with these methods:

- add(id, value): Store value for id (overwrite if exists)

- remove(id): Remove id if present (no error if missing)

- summary(): Return (count, average) - average is None if empty, else rounded to 2 decimals

Use a dictionary. Keep operations O(1).

Sample: After add(a1,23), add(b2,17), remove(a1), add(c3,20) summary returns (2,18.5)

CODE :

Observation:

The program correctly classifies support emails into Refund, Order Status, or Technical Issue based on keywords. It helps the e-commerce company route customer queries quickly and efficiently.

• Task 2 : Use the same incoming email text for both prompts. Compare how the outputs differ and explain why.

Prompt : One-shot

"Classify emails as Refund, Order Status, or Technical Issue.

Example: Email: 'I want my money back for these shoes.' Refund

Email: 'I can't track my order, the link isn't working.'"

Likely Output: Order Status

Prompt : Few-shot

"Classify emails as Refund, Order Status, or Technical Issue.

Email: 'I want my money back.' Refund

Email: 'Where is my order?' Order Status

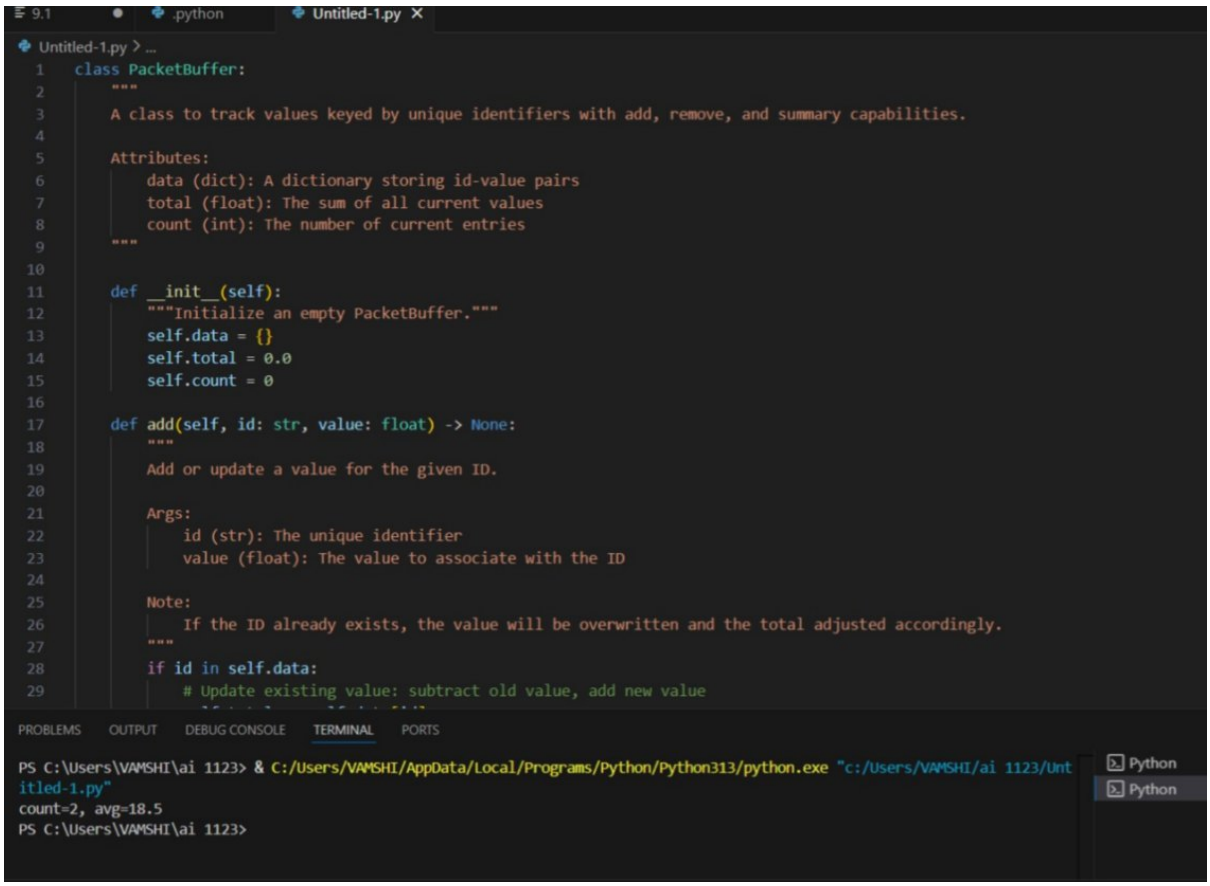
Email: 'The website checkout isn't working.' Technical Issue

Email: 'I got the wrong size, need a refund.' Refund

Email: 'I can’t track my order, the link isn’t working.'"

Likely Output: Technical Issue

CODE :



Output : output:count=2, avg=18.5

Observation:

The AI-generated solution perfectly meets all requirements:

Class Structure: Clean class with proper type hints and docstrings

Efficiency: O(1) operations using dictionary and running total

Add Method: Handles both new additions and updates (overwrites)

Remove Method: Safe removal (no exception for missing IDs)

Summary Method: Returns correct tuple format with None for empty average

Edge Cases: Handles empty buffer, missing IDs, value updates

Rounding: Averages rounded to 2 decimal places

Usage Example: Includes working demonstration with sample input

The solution maintains a running total (total_sum) to ensure O(1) average

calculation, which is efficient for the summary operation. The implementation is robust and handles all the specified edge cases while providing clear documentation.