

# 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 :

The screenshot shows a Python script named `parse_telemetry_data.py` in a code editor. The code defines a function `parse_telemetry_data` that takes raw text as input and returns a tuple containing a dictionary of per-cell averages and an overall average. The code uses a defaultdict to accumulate sum and count for each cell ID. It processes lines of CSV data, ignoring empty lines and ensuring exactly three fields per line. The terminal below shows the execution of the script and its output, which includes overall and per-cell average latencies.

```
9.1 task8.py .python from typing import Dict, Tuple, Optional.py Untitled-1.py
python > parse_telemetry_data
1 def parse_telemetry_data(raw_text):
2     """
3         Parse telecom network telemetry data and compute per-cell and overall averages.
4
5     Args:
6         raw_text (str): Raw text containing CSV lines with id,timestamp,latency_ms
7
8     Returns:
9         tuple: (dict of {id: average_latency}, overall_average)
10    """
11    from collections import defaultdict
12
13    # Initialize data structures
14    cell_data = defaultdict(lambda: {'sum': 0.0, 'count': 0})
15    total_sum = 0.0
16    total_count = 0
17
18    # Process each line
19    lines = raw_text.strip().split('\n')
20
21    for line in lines:
22        # Clean up the line
23        line = line.strip()
24        if not line:
25            continue
26
27        # Split into fields
28        fields = line.split(',')
29        if len(fields) != 3:
30
Overall average: 35.2
PS C:\Users\VAMSHI\ai 1123> & C:/Users/VAMSHI/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/VAMSHI/ai 1123/.python"
Per-cell averages: {'ce141': 33.7, 'ce142': 35.2, 'ce143': 36.7}
Overall average: 35.2
PS C:\Users\VAMSHI\ai 1123>
```

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 :

```
9.1 • .python Untitled-1.py X
❶ Untitled-1.py > ...
❷ class PacketBuffer:
❸     """
❹         A class to track values keyed by unique identifiers with add, remove, and summary capabilities.
❺
❻     Attributes:
❼         data (dict): A dictionary storing id-value pairs
➋         total (float): The sum of all current values
⌽         count (int): The number of current entries
⌾
⌿
⌽     """
⌾
⌽     def __init__(self):
⌽         """Initialize an empty PacketBuffer."""
⌽         self.data = {}
⌽         self.total = 0.0
⌽         self.count = 0
⌽
⌽     def add(self, id: str, value: float) -> None:
⌽         """
⌽             Add or update a value for the given ID.
⌽
⌽             Args:
⌽                 id (str): The unique identifier
⌽                 value (float): The value to associate with the ID
⌽
⌽             Note:
⌽                 If the ID already exists, the value will be overwritten and the total adjusted accordingly.
⌽
⌽         if id in self.data:
⌽             # Update existing value: subtract old value, add new value
⌽
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\VAMSHI\ai 1123> & C:/Users/VAMSHI/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/VAMSHI/ai 1123/Untitled-1.py"
Python
Python
count=2, avg=18.5
PS C:\Users\VAMSHI\ai 1123>
```

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.