AI ASSISTED CODING LAB TEST -02

NAME : M.SHIVA

HALL.NO : 2403A52377

DATE : 17-09-2025

BATCH.NO : AI 14

Prompt a 1 : perplexity.ai

"Write a Python function that parses raw CSV text, calculates per-id average output_kw skipping malformed/non-numeric lines, and returns both per-id averages as a dict and overall average rounded to 2 decimals."

Code :

```
def turbine_averages(log_text):
    sums, counts = {}, {}
    for line in log_text.strip().split('\n'):
        parts = line.strip().split(',')
        if len(parts) != 3:
            continue
        tid, ts, val = parts
        try:
            val = float(val)
        except:
            continue
        sums[tid] = sums.get(tid, 0) + val
        counts[tid] = counts.get(tid, 0) + 1
    avgs = {tid: round(sums[tid]/counts[tid], 2) for tid in sums}
    overall = round(sum(sums.values())/sum(counts.values()), 2) if counts else 0
    return avgs, overall

txt = '''tu121,2025-01-01T08:00,31.7
```

tu122,2025-01-02T09:00,33.2

tu123,2025-01-03T010:00,34.7'''

avgs, overall = turbine_averages(txt)

print(avgs, overall)

output :

{'tu121': 31.7, 'tu122': 33.2, 'tu123': 34.7} 33.2

Explanation :

- Split each line and check it has 3 fields.
- Try converting output_kw to float, skip bad lines.
- Use dictionary to collect sum and count per turbine.
- Calculate per-turbine average and round to 2 decimals.
- Calculate overall average from all turbines.

Prompt a 2 : **perplexity.ai**

"Write a minimal Python class GridMonitor that can add/remove float values by ID, and provide count and average (rounded 2 decimals). Safe on missing IDs, average None if empty."

Code :

```
class GridMonitor:
    def __init__(self):
        self.store = {}
    def add(self, id, value):
        self.store[id] = value
    def remove(self, id):
```

```
        self.store.pop(id, None)
    def summary(self):
        n = len(self.store)
        avg = round(sum(self.store.values())/n, 2) if n else None
        return n, avg

mon = GridMonitor()
mon.add('a1', 21)
mon.add('b2', 17)
mon.remove('a1')
mon.add('c3', 18)
print(mon.summary())
```

Output :

(2, 17.5)

Explanation :

- Uses dict for {id: value}, add overwrites.
- remove ignores missing IDs.
- summary: returns (count, avg), avg None if empty.
- Average rounded, all ops O(1).