AI-ASSISTED CODING
LAB TEST

SUBGROUP-B

H.NO: 2403A51236

NAME: S. SIRI

BATCH: 11

B.1 — [S18B1] Apply surge/penalty rules (conditionals)

Context:

Pricing in the real estate listings platform app uses a base per-km rate and time-based surge after peaks. Finance wants a deterministic calculator for receipts and audits.

Your Task:

Implement a fare function: fare = km * base_per_km * surgeMultiplier, where surge applies strictly after 18:00 local time.

Data & Edge Cases:

Input is a list of rides with `time` as HH:MM (24h) and `km` as float. Exactly 18:00 is non-surge; > 18:00 is surge.

AI Assistance Expectation:

Prompt AI to outline parsing HH:MM, applying conditionals, and rounding to 2 decimals; then implement and write a quick test.

Constraints & Notes:

No external libraries; round each fare to 2 decimals; do not mutate input.

Sample Input

[{'time': '07:45', 'km': 2.8}, {'time': '18:45', 'km': 6.2}]

Sample Output

[39.2, 108.5]

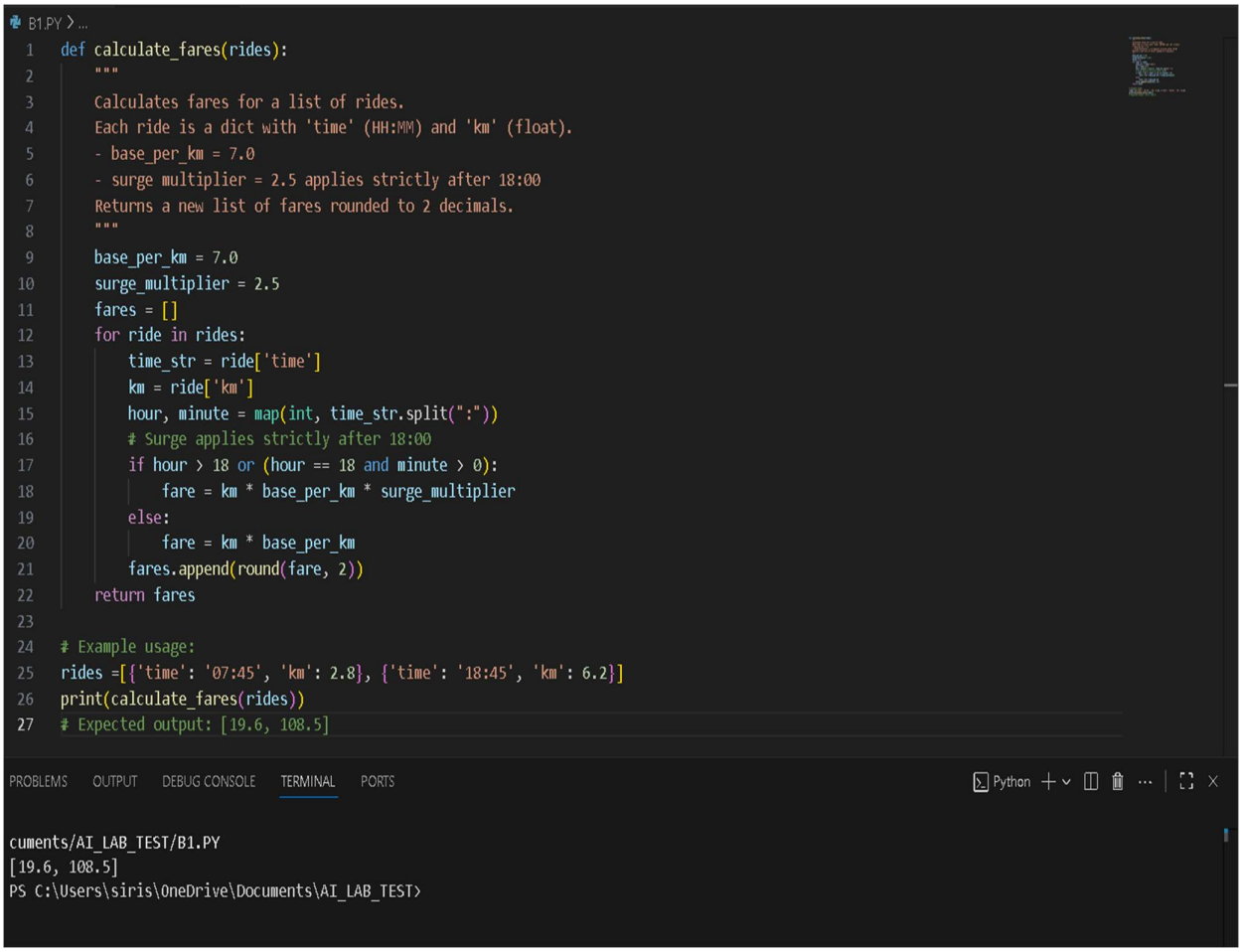
Acceptance Criteria: Correct surge threshold and rounding.

PROMPT:

Write a Python function calculate_fares(rides) where each ride is a dict with 'time' in HH:MM and 'km' as float.

➔ base_per_km = 7.0, surge multiplier = 2.5 applies strictly after 18:00. Round fares to 2 decimals and do not mutate input.

CODE AND OUTPUT:



B.2 — [S18B2] Debug rolling mean (off-by-one)

Context:

A team in real estate listings platform noticed off-by-one bugs in a rolling KPI computation (moving averages) that undercount windows.

Your Task:

Use AI to identify the bug and fix the window iteration so all valid windows are included.

Data & Edge Cases:

For xs=[4, 5, 7, 10] and w=2, number of windows should be len(xs)-w+1.

AI Assistance Expectation:

Ask AI to add a failing test first, propose the minimal fix, and verify with the sample.

Constraints & Notes:

Guard invalid w (<=0 or >len(xs)); preserve O(n*w) simple solution.

Sample Input

xs=[4, 5, 7, 10], w=2
Buggy code:

```
def rolling_mean(xs, w):
  sums = []
  for i in range(len(xs)-w):
    window = xs[i:i+w]
    sums.append(sum(window)/w)
  return sums
```

Sample Output

[4.5, 6.0, 8.5]

Acceptance Criteria: All valid windows included; passes tests; no index errors.

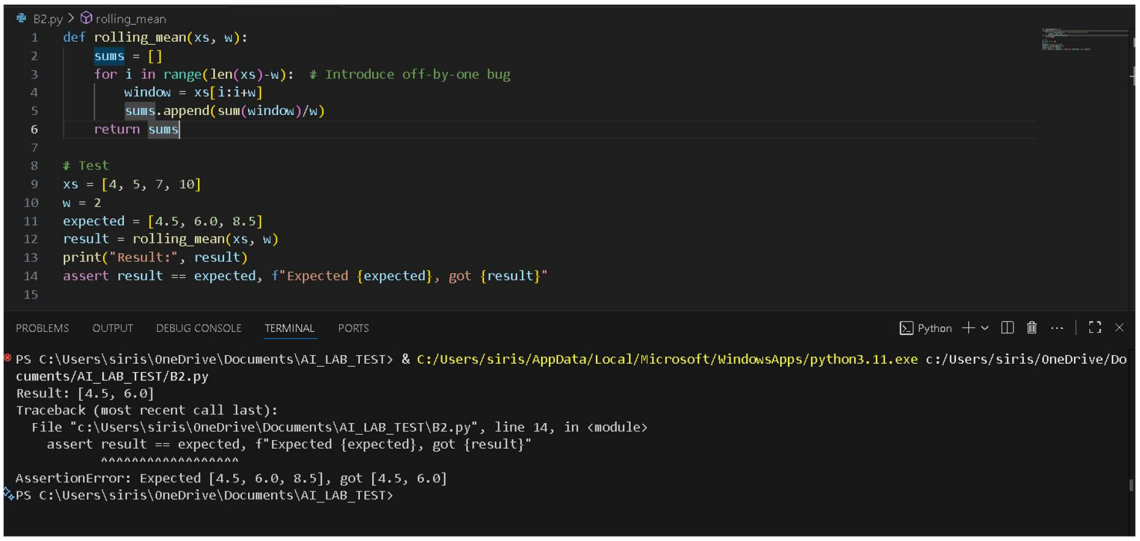
PROMPT:

The function below has an off-by-one bug in its rolling mean window iteration.

Write a failing test for xs = [4,5,7,10], w=2 expecting [4.5,6.0,8.5],

then fix the function minimally so the test passes.

INITIAL CODE (BUGGY CODE) WITH FAILING TEST:



CORRECTED CODE WITH OUTPUT: