Day X Python Challenge — Commission & Bonus Calculator

Scenario:
You work for a school store that pays sales staff a **commission** on monthly sales. Your manager wants a program that greets the salesperson by name, calculates their commission, and tells them whether they qualified for a monthly **bonus**.

Your program should:

1. **Ask for inputs** with `input()` and store them in variables:
    - The salesperson's **name** (string).
    - The salesperson's **total monthly sales** (float).
2. **Calculate the commission** based on **tiered rates**:
    - 13% if sales are **below $10,000**
    - 15% if sales are **$10,000 or more**
3. **Round** the commission to **two decimals** and **print** a sentence with the **name** and the **commission** using an **f-string**.
4. **Use comparison + logical operators** to report a simple status:
    - If sales are **≥ $8,000 and** commission is **≥ $1,000**, print: `"Bonus unlocked!"`
    - Otherwise print: `"Keep pushing—you're close!"`
5. **Do a quick identity tag** using **string slicing** on the name:
    - Create `emp_code` from the **first 3 letters** of the name (uppercase) + the **last 2 digits** of the year `"25"`.
    - Example: `"Marvin"` → `"MAR25"`. Include this in your output line.

6. **(Mini list task)** Ask for **three product sales** (as three separate `float` inputs). Store them in a **list**, compute their **sum**, and print the list and its total.
    - If the **list total** is within **$1** of the monthly sales, print: `"Sales breakdown verified."` else `"Sales breakdown does not match."`

Guidelines (read before you code)

- Remember that `input()` returns a **string**. Convert numeric inputs with `float()` where needed.
- Use **f-strings** for your final messages.
- Use **`round(value, 2)`** or formatted f-strings (e.g., `f"{value:.2f}"`) to display money.
- Keep your code clear: name your variables meaningfully and add a couple of comments.

Sample Output:

Enter your name: Marvin

Enter total monthly sales: 10450.5

Enter product #1 sales: 4000

Enter product #2 sales: 3500.5

Enter product #3 sales: 2950

Hello MAR25 (Marvin)! Your commission is $1567.58 at a 15% rate.

Bonus unlocked!

Sales items: [4000.0, 3500.5, 2950.0] → total $10450.50

Sales breakdown verified.

Personal Data Analyzer — Project Rubric

Category	Exemplary (4)	Proficient (3)	Developing (2)	Beginning (1)
1. Program Functionality	The program runs smoothly with **no errors**. All required features (input, math, logic, lists, f-strings, slicing) are included and functional.	The program runs with **minor errors** or missing one feature but still demonstrates understanding.	The program runs partially; multiple features missing or not functioning correctly.	The program does not run or shows little evidence of understanding Python syntax.
2. Logical & Comparison Operators	Correct and efficient use of multiple logical/comparison operators (`and`, `or`, `<`, `>=`, etc.) for decision-making.	Correct use of logical/comparison operators in most cases; few syntax or logic errors.	Attempted use of logical/comparison operators but with incorrect syntax or meaning.	Minimal or no use of logical/comparison operators.
3. Math & Computation Accuracy	Math calculations are accurate, variables are clearly named, and expressions demonstrate understanding of operator precedence.	Most calculations are correct; a few minor mistakes or unclear variable names.	Some understanding of math operations but frequent syntax or logic errors.	Limited or incorrect use of math; unclear purpose in calculations.
4. Lists & Data Handling	Creates, manipulates, and prints lists accurately (e.g., using `append`, `sum`, or slicing). Demonstrates understanding of list purpose.	Creates a list and accesses elements correctly.	Partial understanding of list creation or manipulation.	No evidence of list usage or incorrect implementation.

5. String Formatting & Concatenation	Uses **f-strings** and **concatenation** effectively to produce clean, readable output; includes **string slicing** for creative display.	Mostly correct use of f-strings or concatenation; may miss slicing or have small formatting issues.	Some use of f-strings or concatenation; output messy or incomplete.	No use of f-strings or concatenation; raw print statements only.
6. Creativity & Integration	Program feels polished and creative — integrates multiple elements smoothly and shows original thinking.	Program is complete and clear, with minor creative or design elements.	Program demonstrates effort but feels incomplete or unoriginal.	Program lacks structure or creativity.
7. Code Clarity & Comments	Clear variable names, consistent indentation, and meaningful comments explaining logic throughout.	Mostly clear code; some comments missing..	Inconsistent naming/indentation; few comments.	Code disorganized and difficult to read; no comments.
8. Reflection / Explanation	Student provides a detailed explanation of how the program works and what each part does. Can explain **why** specific operators or structures were used.	Student can explain most of the code clearly; misses minor details.	Student explanation is vague or partially incorrect.	Student cannot explain how the program works