Sarika Palle

2403A51L33

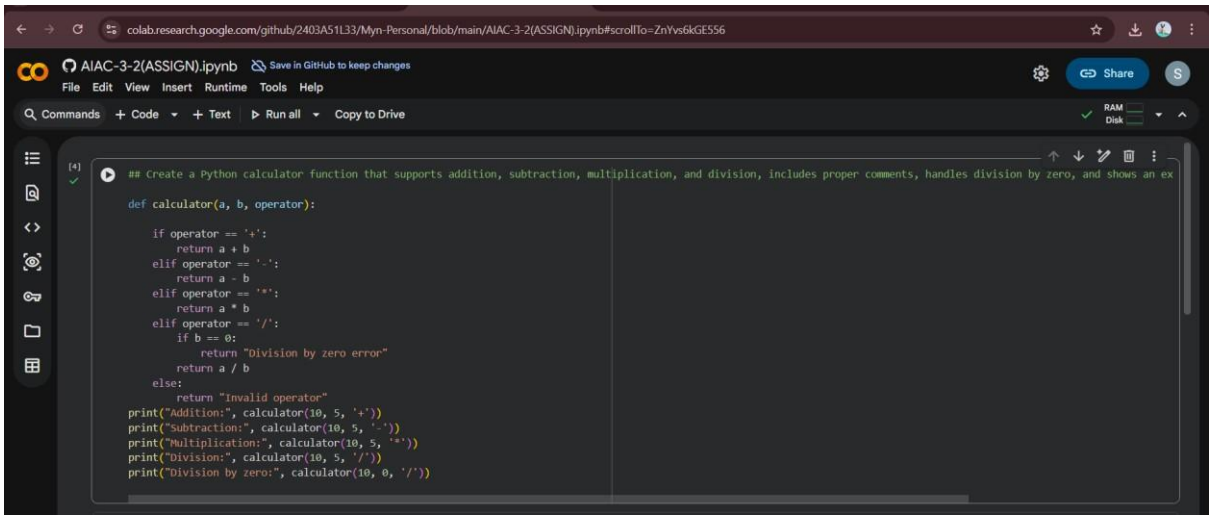
B-52

ASSIGNMENT – 3.2

Lab 3: Prompt Engineering – Improving Prompts and Context Management

Task– 1: Progressive Prompting for Calculator Design

Prompt: Create a Python calculator function that supports addition, subtraction, multiplication, and division, includes proper comments, handles division by zero, and shows an example of how the function is used.



OUTPUT:

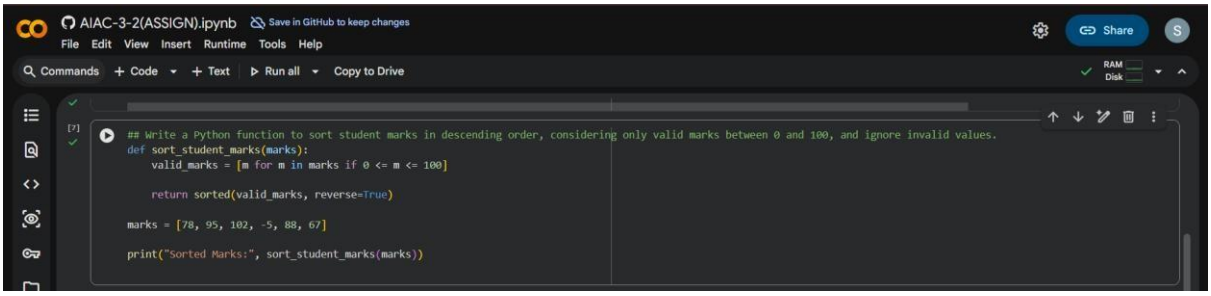
Explanation:

Initially, limited prompt information leads to a simple implementation. Adding comments, examples, and constraints helps the AI generate structured logic with proper error handling.

This shows how prompt refinement improves code quality.

Task – 2: Refining Prompts for Sorting Logic

Prompt: Write a Python function to sort student marks in descending order, considering only valid marks between 0 and 100, and ignore invalid values.



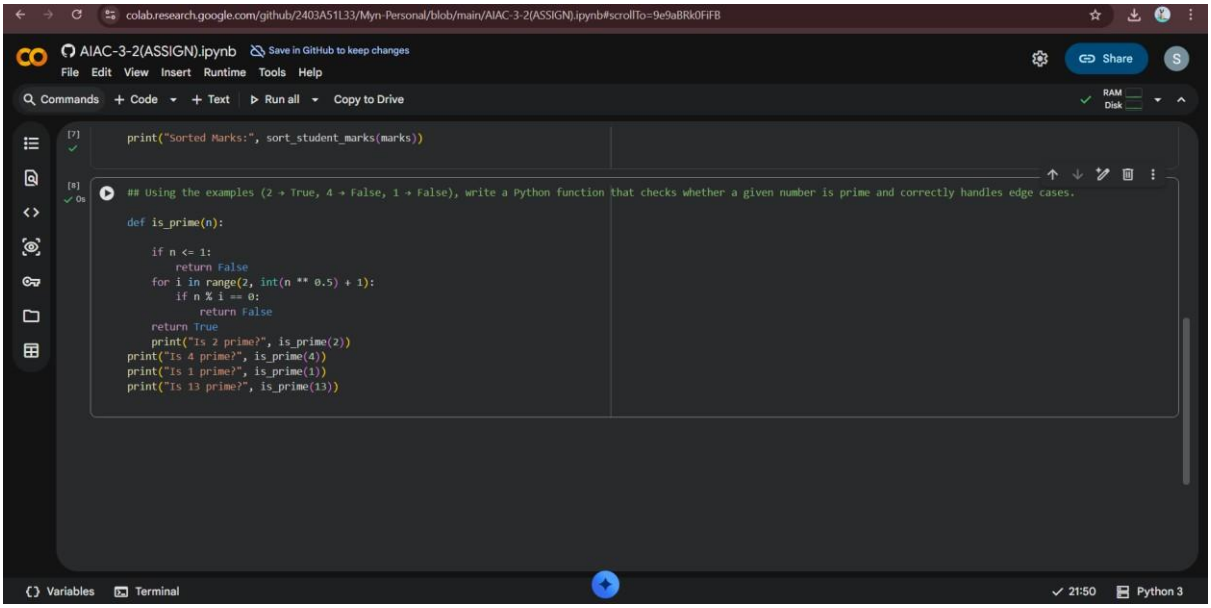
OUTPUT:

Explanation:

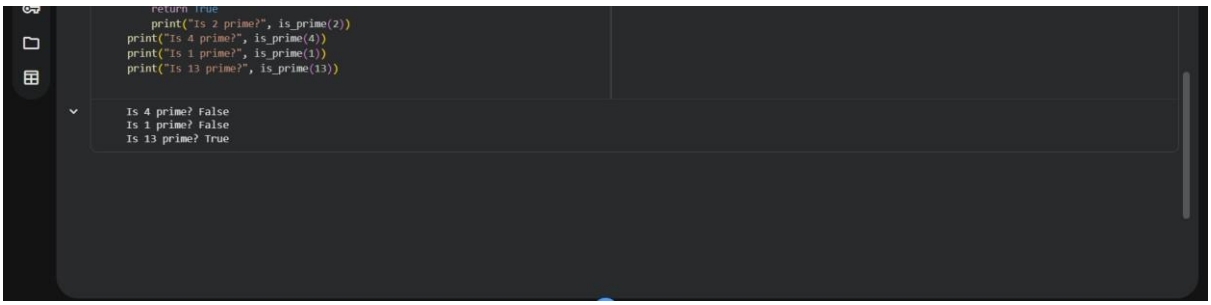
A vague prompt results in generic sorting without validation.
Providing clear constraints such as order and valid range enables the AI to produce accurate and meaningful logic.
Prompt clarity removes ambiguity in implementation.

Task– 3: Few-Shot Prompting for Prime Number Validation

Prompt: Using the examples (2 → True, 4 → False, 1 → False), write a Python function that checks whether a given number is prime and correctly handles edge cases.



OUTPUT:



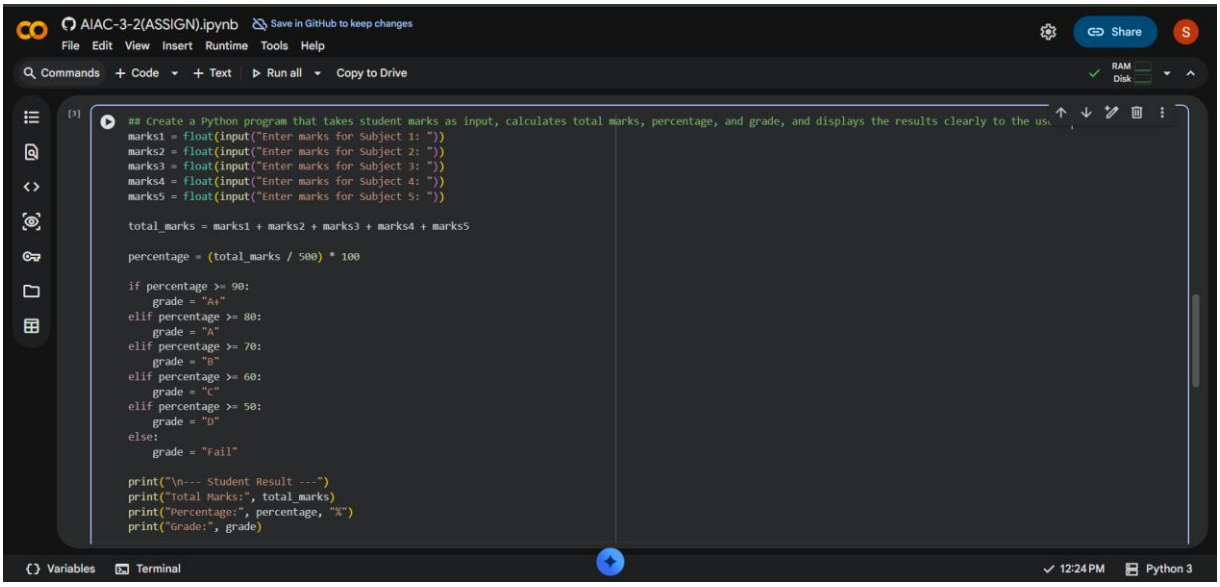
Explanation:

Few-shot prompting provides clear expectations through examples. This helps the AI understand edge cases and generate correct primechecking logic.

Accuracy improves compared to zero-example prompts.

Task– 4: Prompt-Guided UI Design for Student Grading System

Prompt: Create a Python program that takes student marks as input, calculates total marks, percentage, and grade, and displays the results clearly to the user.



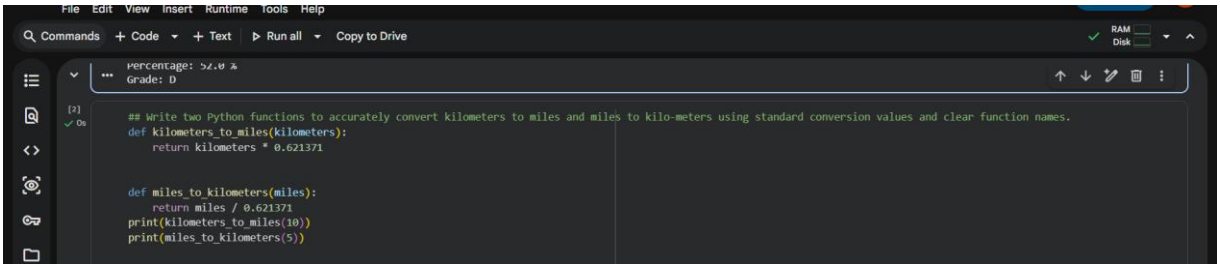
OUTPUT:

Explanation:

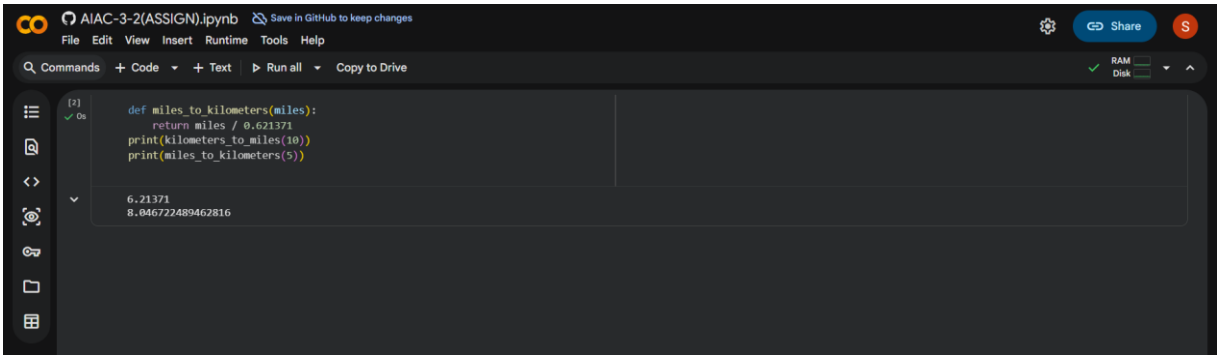
Clear prompt instructions guide the AI to generate a structured and interactive program.
The code correctly handles user input, calculations, and result display. Prompt guidance improves usability and readability.

Task– 5: Analysing Prompt Specificity in Unit Conversion Functions

Prompt: Write two Python functions to accurately convert kilometers to miles and miles to kilo-meters using standard conversion values and clear function names.



OUTPUT:



Explanation:

Specific prompts lead to accurate and well-defined conversion functions.
Clear instructions ensure correct formulas and readable function names.
This demonstrates how prompt specificity improves program correctness.