GitHub Copilot Practice Assignments Day-1

Practice Assignment 1 (JavaScript or C#): Prompting Techniques Explorer

Objective:

Build a Simple Loan Calculator application using different prompting styles to guide GitHub Copilot. Use the following prompting techniques

1. Few-shot
2. Zero-shot
3. Instructional
4. Language-style
5. Comment-based
6. Comment chaining

Tasks & Steps:

- Takes loan amount, interest rate, and duration (months) as input.
- Calculates the monthly EMI, total repayment, and total interest.
- Displays the result in a user-friendly format.

Practice Assignment 2 (JavaScript): Build a Smart String Formatter Utility

Objective:

Create a smart string formatting utility in JavaScript that:

- Formats user input strings to title case, sentence case, and camelCase.
- Removes special characters, trims extra spaces, and validates input type.

Tasks & Steps:

1. Priming GitHub Copilot
2. Implement the function
3. Use GitHub Copilot Inline Chat
4. Perform Code Review using Copilot Chat
5. Prompt Engineering Practice
6. Final Test

Expected Outcome:

- Efficient formatting utility

Practice Assignment 3 (JavaScript): Expression Evaluator with Custom Syntax Support

Objective:

Develop a powerful expression evaluator that:

- Parses and evaluates mathematical expressions as strings (e.g., "4 + 5 * (2 - 3)")
- Supports custom operators like # for power (2 # 3 = 8) and @ for modulo
- Handles variables (e.g., "x + y * 2" where x = 5, y = 2)
- Implements error handling for invalid expressions

Tasks & Steps:

1. Priming GitHub Copilot
2. Implement the function
3. Use GitHub Copilot Inline Chat
4. Prompt Engineering Techniques
5. Instructional Prompting and Context specific prompting
6. Code review
7. Final Testing

Expected Outcome:

- Modular and error-resilient evaluator
- Demonstrated use of inline chat and Copilot for parsing and logic)

Practice Assignment 4 (C#): AI-based To-Do Task Analyzer

Objective:

Create a function in C# that analyses a list of to-do tasks and returns:

- A list of categorized tasks (e.g., Work, Personal, Urgent).
- A priority order for each task based on urgency (ASAP, today, etc.).
- Detection of incomplete or vague task descriptions (e.g., "Finish project" should be flagged as unclear).

Tasks & Steps:

1. Prompt Priming
2. Function Plan
    a. Categorize Tasks: Check for keywords like "work," "personal," "urgent.
    b. Priority Detection: Use keywords like "ASAP," "today," "next week."
    c. Vague Task Detection: Flag tasks with unclear descriptions (e.g., missing actionable verbs).
3. Prompt Engineering Techniques
    a. Step-by-step prompting
    b. Few-shot prompting
4. Use copilot chat
5. Code performance and review
6. Ensure proper validation
7. Export results to csv

Expected Outcome:

- Takes a list of task descriptions and categorizes them into Work, Personal, Urgent, or Unclear.
- Detects priority levels (e.g., High, Medium, Low).
- Flags vague or incomplete tasks (e.g., missing verbs, incomplete descriptions).
- Optionally, exports the results to a CSV file.
- Code Review and Testing