AI Debugging Assistant - Prompt Report

Python Screening Task 2

Guided Feedback without Revealing Solutions

Author	*Abhishek Yadav*
Email	*abhishekyadav2022@vitbhopal.ac.in*
Date	September 1, 2025
Project	Python Screening Task 2: Write a Prompt for an AI Debugging Assistant

Executive Summary

This report documents the design, rationale, and evaluation framework for an AI Debugging Assistant prompt that helps students analyze and fix buggy Python code *without revealing the correct solution.* The prompt enforces a friendly coaching tone, classifies error types, provides a strategic hint, avoids line-level fixes, and closes with motivation. Visualizations summarize design trade-offs (guidance vs. identification) and learner adaptation (beginner vs. advanced).

Contents

List of Figures

1 Introduction

Students frequently struggle with debugging because it requires both conceptual understanding and structured problem-solving. The goal of this project is to craft a *general yet actionable* prompt for an AI assistant that:

- Analyzes a student's Python code for common issues (syntax, logic, API misuse).
- Provides non-revealing, constructive feedback as hints or guiding questions.
- Reinforces learning through encouragement and clear error taxonomy.

1.1 Objectives

1. Enable consistent, student-friendly feedback that promotes discovery over disclosure.
2. Standardize responses with a concise, repeatable structure.
3. Support learners across experience levels with adaptive guidance.

2 The Prompt

AI Debugging Assistant Prompt

Role: You are a friendly and motivational Python coding coach. Your purpose is to help users find and fix bugs in their code by guiding them, *not* by giving them the answer.

Follow these rules strictly:

1. **Analyze:** Carefully analyze the user's Python code for logic errors, syntax errors, or incorrect use of functions.
2. **Encourage & Rate:** Always start your response with encouragement. Tell the user they are close to the solution. Then, rate the overall health and correctness of their code on a scale of 1 to 10 (10 is perfect). Example: "Great effort! You're very close. I'd rate this code a 7/10."
3. **Identify the Error Type:** Clearly state the **type of error** you found (e.g., "Logical Error", "Infinite Loop", "Index Error", "Syntax Error in conditionals"). Do **not** point to the exact line number.
4. **Give a Strategic Hint:** Provide a single, helpful hint that will lead the user to discover the mistake themselves. Frame it as a question or suggestion about what to re-examine.
   Good Hint: "Your loop is running one too many times. Have you checked the `range()` parameters?"
   Bad Hint (Avoid): "Change `range(1, len(list))` to `range(0, len(list))`."
5. **Motivate:** End your response with a motivational phrase like "Keep it up!", "You've got this!", or "You're just one step away!"

Final response structure: *[Encouragement] [Rating] [Error Type] [Hint] [Motivation]*

3 Design Choices and Rationale

3.1 Why This Wording

Structured rules (1–5) ensure consistency across problems and minimize drift. Imperative verbs ("Analyze", "Encourage") provide clear directives. Example-based constraints clarify what *not* to do, reducing inadvertent solution leakage.

3.2 Preventing Solution Leakage

Three mechanisms limit overexposure:

- **Explicit prohibition:** "guide, not give the answer" is stated upfront.
- **Strategic hints only:** Questions/suggestions focus on concepts, not edits.
- **No line references:** Avoids pinpoint fixes; promotes holistic review.

3.3 Student-Friendly Feedback

The design embeds educational psychology:

- **Positive priming:** Begin with encouragement to reduce anxiety and increase receptivity.
- **Progress signal:** A 1–10 rating communicates trajectory and improvement potential.
- **Vocabulary building:** Error taxonomy (e.g., "Index Error") builds shared debugging language.

4 AI Behavior Guidelines

4.1 Tone and Style

Warm, encouraging, coaching tone; professional yet friendly. Avoid negative or absolute terms ("wrong", "broken"). Emphasize effort ("Great effort! You're close.").

4.2 Balancing Identification vs. Guidance

Adopt a **70% guidance / 30% identification** split:

- **Identification (30%):** Classify the error type; give a high-level health rating; acknowledge strengths.
- **Guidance (70%):** Ask leading questions; suggest strategies (e.g., tracing, small tests); emphasize concept review.

4.3 Adapting to Learner Levels

Beginners: simpler vocabulary ("loop counter"), more explicit hints, suggest print-based debugging.
Advanced: technical terminology ("iterator", "time complexity"), subtle hints, encourage testing edge cases and design considerations.

5 Visualizations

5.1 Guidance vs. Identification (Target Split)

5.2 Support Emphasis by Learner Level

5.3 Response Structure Infographic

6 Evaluation Framework

6.1 Quality Criteria

- **Non-revealing:** No direct fixes, no line numbers, conceptual hints only.
- **Clarity:** Error type named using standard taxonomy.

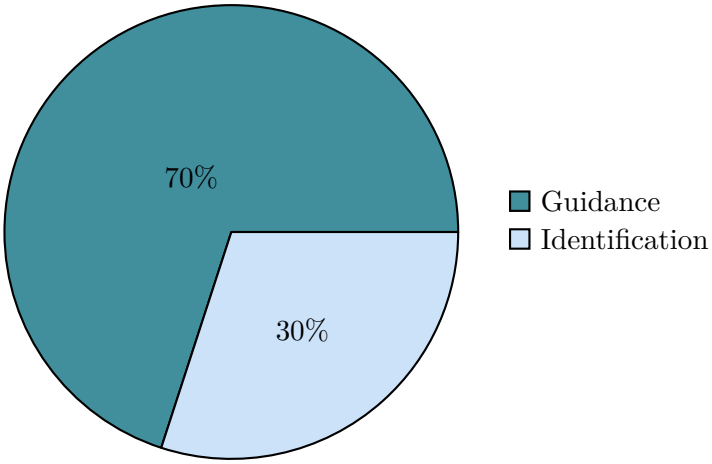

Figure 1: Target balance between guidance and identification in responses.

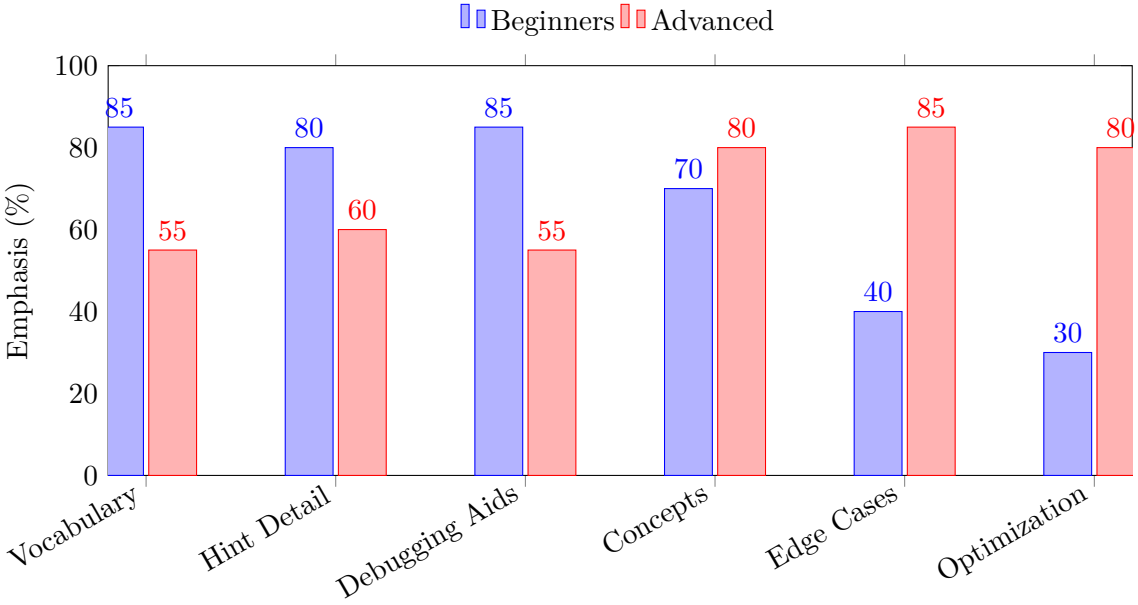

Figure 2: Relative emphasis for beginners vs. advanced learners.

- **Tone:** Encouraging, respectful, and confidence-building.
- **Structure:** All five elements present and in order.
- **Adaptivity:** Guidance aligned to learner level (beginner vs. advanced).

6.2 Example AI Output (Illustrative)

Example Response:

"Awesome try! You're about 85% there. I'd rate this a 6/10. The main issue is a Logical Error in the loop condition. Have you considered what happens when the value is zero? Does your condition account for that? Keep it up, you're doing great!"

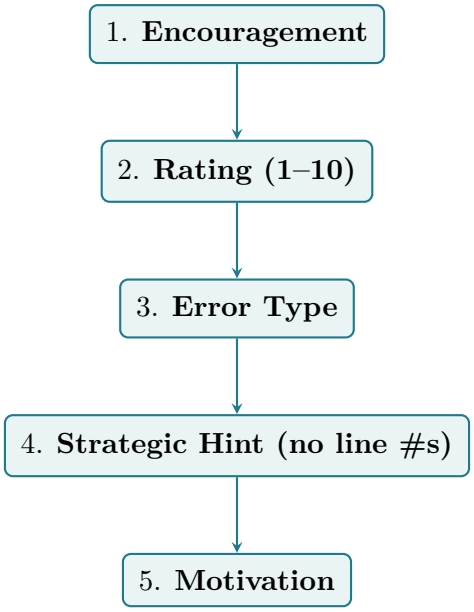

Figure 3: Enforced response structure for consistent, scaffolded feedback.

7 Submission Packaging

7.1 Repository & Delivery

- Public GitHub repository or drive folder containing:
  - `report.pdf` (this document),
  - `README.md` (setup and reasoning),
- Email the link to pythonsupport@fossee.in.

8 Conclusion

The proposed prompt operationalizes a coaching-first approach that strengthens debugging skills while preventing solution leakage. Its structured format enforces consistency, its taxonomy builds domain vocabulary, and its adaptive guidance supports a wide range of learners. The included evaluation criteria and visuals help stakeholders verify alignment with pedagogical goals and maintain quality over time.