Assignment 1: AI Teaching Assistant Prompt Solution

Use Case: AI Teaching Assistant for Computer Science.

Problem Understanding: Students often find abstract concepts like "Recursion" or "Dynamic Programming" intimidating. A standard AI response might provide a direct solution, which hinders learning. The goal is to design a **Socratic-style Teaching Assistant** that guides the student through the logic without providing the final code immediately.

Structured Prompt Design: To solve this, I have designed a prompt using the **Persona-Task-Constraint (PTC)** framework and **Chain-of-Thought (CoT)** prompting.

> **System Prompt:** "You are an expert Computer Science Teaching Assistant. Your **Persona** is patient, encouraging, and highly technical. **Task:** Your goal is to help students understand coding logic through the Socratic method. **Constraints:**
>
> 1. Never provide the full code solution in your first response.
> 2. Break down the student's problem into smaller, manageable logical steps.
> 3. Ask one guiding question at a time to lead the student to the answer.
> 4. Use analogies to explain complex memory or logic concepts. **Chain-of-Thought Requirement:** Before responding, think step-by-step about where the student's logic might be failing."

Iterative Improvement:

- **Initial Output Issue:** In early testing, the model was too "helpful" and gave away the base case of recursive functions too quickly.
- **Improvement:** I added a "Negative Constraint" to the prompt: *"Do not define the base case for the student; instead, ask them what happens when the input reaches its smallest possible value."*

Link to chat:

https://chatgpt.com/share/69597d8c-1280-8012-9651-d4d094b02913

Assignment 2: Three Advanced Techniques for Improving AI Responses

Note: The following techniques are from external industry standards and were not explicitly detailed in the provided sources.

1. **Chain-of-Density (CoD) Prompting:** This technique is used primarily for summarization. The model is instructed to generate an initial summary and then iteratively identify and incorporate missing "entity-dense" information without increasing the word count, making the output increasingly information-rich.
2. **Skeleton-of-Thought (SoT):** This technique addresses latency and structure. The model first generates a "skeleton" (an outline) of the answer and then expands each segment. This is particularly useful for long-form content generation where logical flow is critical.
3. **Self-Consistency Sampling:** Instead of taking a single output, the model generates multiple different reasoning paths (Chain-of-Thought) for the same problem. The final answer is determined by a "majority vote" among the generated paths, which significantly improves performance on arithmetic and commonsense reasoning tasks.

Assignment 3: Prompt Engineering vs. Context Engineering

Introduction

In the evolving landscape of Generative AI, the performance of a Large Language Model (LLM) depends heavily on how it is guided. Two primary methodologies for this guidance are **Prompt Engineering** and **Context Engineering**.

Defining Prompt Engineering

Prompt Engineering focuses on the **instructional input** provided to the model. It is the art of "talking" to the AI by refining the specific wording, tone, and structure of the query.

- **Primary Goal:** To elicit a specific style or type of response from the model's pre-existing knowledge base.
- **Focus:** It prioritizes the **logic of the request**, utilizing techniques like Few-Shot prompting (providing examples) or Chain-of-Thought (asking the model to "think step-by-step").

Defining Context Engineering

Context Engineering is a more data-centric approach. It focuses on the **environment and information** surrounding the prompt. Rather than just changing the "ask," it changes the "knowledge" available to the model for that specific session.

- **Primary Goal:** To provide the model with specific, often private or updated, data it was not originally trained on.
- **Focus:** It prioritizes **relevance and grounding**. This often involves **Retrieval-Augmented Generation (RAG)**, where external documents are injected into the model's "context window" to prevent hallucinations and ensure accuracy.

Key Differences

Feature	Prompt Engineering	Context Engineering
Focus	How the question is asked.	What information the model has.
Method	Wording, structure, and persona.	Data retrieval, RAG, and window management.
Source	Relies on the model's internal training.	Relies on external, provided data sources.
Use Case	Adjusting tone or logic style.	Answering questions about specific company files.

Conclusion

While **Prompt Engineering** is about the **"how"** (the strategy of communication), **Context Engineering** is about the **"what"** (the specific data the model uses to form its answer). Effective AI solutions usually require a hybrid of both: a well-structured prompt acting on high-quality, relevant context.

Analogy for Understanding: Think of an LLM as a **highly skilled chef**.

- **Prompt Engineering** is like giving the chef **specific cooking instructions** (e.g., "Make this spicy and use a French style").
- **Context Engineering** is like providing the chef with **specific ingredients** from your own pantry (e.g., "Use these specific tomatoes and this brand of flour") to ensure the meal tastes exactly like your family recipe.