Day 4
Prompt Design Patterns (ReAct, Self-Reflect)

Why Prompt Patterns Matter

Large Language Models (LLMs) interpret instructions with extreme literalness. This means that poorly constructed prompts inevitably lead to suboptimal or inaccurate results. Conversely, well-designed prompts are the cornerstone of effective LLM interaction.

Enhanced Accuracy

Reduces "hallucinations" and improves factual correctness.

Consistent Outputs

Ensures predictable structure and formatting every time.

Automation-Ready

Generates outputs that can be seamlessly integrated into automated workflows.

Superior Reasoning

Facilitates better logical thinking and effective tool utilization.

The prompt we design directly influences the model's behavior and utility.

> “Prompting is programming the LLM with language.”

Persona / Role Prompting

Persona or Role Prompting involves instructing the LLM to adopt a specific identity and behavioral style. This technique is invaluable for eliciting responses with an expert tone, targeting a particular audience, or ensuring domain-specific accuracy.

Template

```
Act as a . Your audience is . Use
```

Structured Output Prompting

For seamless integration into production systems, LLM responses often need to be machine-readable. Structured Output Prompting ensures the model delivers information in predefined formats like JSON, Tables, XML, or Markdown Lists.

Template

```
Return ONLY valid JSON: {"skill": "", "rating": "", "years_of_experience": ""}
```

Example

Input text: “Kiran has 3 years experience in Python and intermediate SQL skills.”

Output:

```
{"skill": "Python", "rating": "Intermediate", "years_of_experience": 3}
```

⚙ **Best Practice:**

- Clearly state the required **format**.
- Specify **validation requirements**.
- Use **ALL CAPS** for non-negotiable rules (e.g., ONLY, NO explanation).

ReAct Prompting (Reason + Act)

ReAct (Reason + Act) Prompting is particularly effective for tasks requiring tool utilization, mathematical calculations, search operations, or multi-step reasoning. It guides the LLM through a structured problem-solving process.

Format

```
Thought:Action:Observation:Final Answer:
```

Example

Question: What is 234 * 77?

```
Action: calculator("234 * 77")Observation: 18018Thought: Calculation doneFinal Answer: 18,018
```

⚡ **Used in:** LangChain Agents, search-connected bots, and Retrieval Augmented Generation (RAG) tools to avoid superficial guesses and enhance reliability.

Self-Critique / Self-Refine

Self-Critique and Self-Refine prompting empowers the LLM to act as its own editor, significantly improving output quality and reducing factual inaccuracies or "hallucinations." This involves a two-phase prompting approach:

01

Phase 1: Generate

The model first creates the initial response or content based on the given instructions.

02

Phase 2: Critique & Improve

The model then critically reviews its own generated output, identifying areas for enhancement and making revisions.

Example

Write a 40-word summary of AI. Then review clarity & improve structure. Return improved version only.

📌 **Tip:** Instruct the model to only return the improved version, avoiding the inclusion of the critique text itself, for a cleaner and more concise output.

Evidence-Based Prompting (Grounded)

Evidence-Based Prompting is crucial in applications like Retrieval Augmented Generation (RAG), where the model is strictly forbidden from inferring or generating information not present in the provided context. It ensures that every claim is directly supported by the given data.

Template

```
Use ONLY the context below. If answer not found, say “I don’t know.” Provide supporting sentence.
```

Example

Context: “Gold is a good conductor of electricity.”

Question: Is gold a better conductor than copper?

Answer: I don’t know. (No information found in the provided context)

✓ Improves Reliability

Ensures responses are factually accurate and sourced.

✓ Prevents Hallucination

Eliminates the generation of fabricated or unverified information.

Decomposition Prompting

Decomposition Prompting is a powerful technique for tackling complex problems by breaking them down into manageable, sequential steps. This approach significantly improves the LLM's reasoning accuracy, especially for lengthy and intricate tasks.

Template

```
Break the task into clear steps. Solve each step. Then combine into a final result.
```

Example

Explain how a transformer model works:

- Step 1: Input → tokens
- Step 2: Attention mechanism…

Final Answer: Summary of above steps.

🔍 **Great for:** Coding tasks, detailed analysis, and complex mathematical problems that require a methodical approach.

Compare & Evaluate Prompting

This technique involves instructing the model to generate multiple potential outputs and then critically compare and evaluate them to select the best option. This leads to a significant boost in the accuracy and quality of the final answer, particularly in subjective or creative tasks.

Example

Generate 3 taglines for a fitness brand. Then select the best one based on clarity & emotion. Return only the best.

Tagline 1:

"Sweat. Achieve. Repeat."

Tagline 2:

"Unleash Your Inner Athlete."

Tagline 3 (Selected):

"Transform Your Body, Elevate Your Mind."

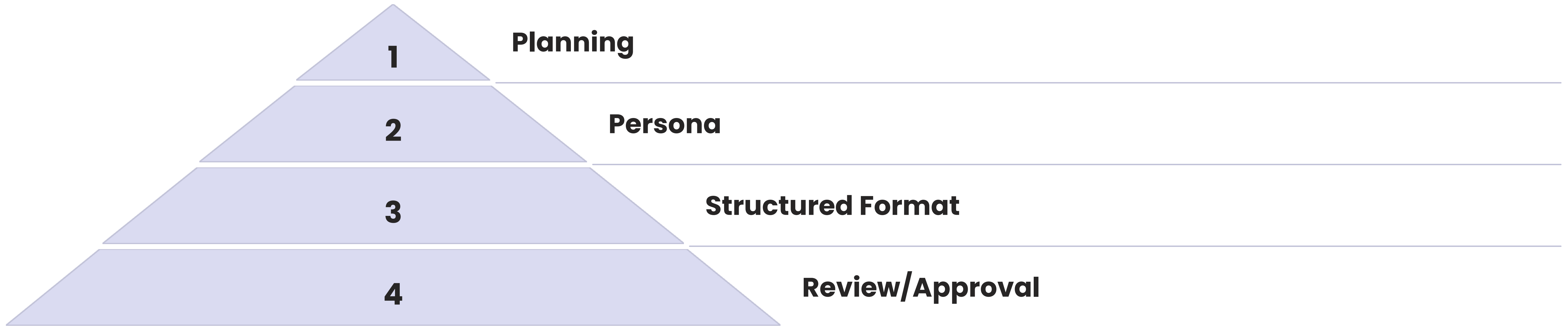
🎯 **Useful for:** Creative endeavors, ambiguous problems, and situations where multiple valid solutions exist.

Planning + Persona + Format Prompt

Combining multiple prompt patterns is the key to achieving the best real-world performance from LLMs, especially for sophisticated applications like Agents and RAG systems. This "pattern stacking" allows for highly nuanced and effective instructions.

Example for writing a report:

Act as a cybersecurity expert. Plan first: list the key sections for a ransomware report. Ask for approval. After approval, write a detailed report in table format.



🧩 **Pattern Stacking:** This synergistic approach is optimal for developing sophisticated AI agents and advanced RAG applications, yielding more comprehensive and accurate results.

Good vs Bad Prompt Summary

"Explain blockchain."	"Explain blockchain in 5 bullets for a 12-year-old with a real-life example."
No specific format	JSON, bullets, table, XML
No defined role	Role defined = domain accuracy
Guessing allowed	Ground answer strictly in provided context

👉 Good prompts are characterized by:

→ **Clear Role Definition**

→ **Structured Format Requirements**

→ **Explicit Constraints**

→ **Verification Mechanisms**

Choosing the best approach to write prompts

Zero-Shot: Simple tasks; no examples needed.*Wrong use → vague/inaccurate results.*

Few-Shot: Specific formats or styles; examples guide output.*Wrong use → formatting drift.*

Chain-of-Thought (CoT): Multi-step reasoning.*Wrong use → shallow or wrong logic.*

ReAct: Tasks needing tools/search + reasoning.*Wrong use → missing or incorrect tool actions.*

Critique / Self-Refinement: High-quality, polished outputs.*Wrong use → unreviewed errors, lower quality.*

What happens if you choose the wrong method

- Lower accuracy
- Missing steps or faulty reasoning
- Poor formatting or inconsistency
- Tool calls not triggered when needed
- Overall weaker performance

Summary

Pattern	What It Improves	Real Benefit
Persona / Role Prompting 🎭	Tone + Expertise	Domain-accurate and audience-specific responses
Structured Output 🧱	Format + Consistency	Directly usable in apps (JSON, tables)
ReAct Prompting 🧠 ⚙️	Step-by-step reasoning + tool use	Better correctness for math, search, agents
Self-Refine / Critique 🔁	Clarity + Coherence	Higher quality with fewer hallucinations
Evidence-Based Prompting 📚	Source grounding	Trustworthy responses (critical for RAG)