QUIZ PREPARATION

OpenAI Agent SDK

Important links:

https://cookbook.openai.com/examples/gpt4-1_prompting_guide

OPENAI AGENTS SDK, TOPICS COVERED IN QUIZ, PROMPT ENGINEERING

- Sir Qasim used 2 approaches:
  - Approach 01: Official documentation
  - Approach 02: Sir Qasim has made UML diagram of OpenAI Agent SDK for better understanding
- A UML (Unified Modeling Language) diagram is a standardized visual modeling language used to specify, visualize, construct, and document the artifacts of software and other systems.
- MCP, Voice agent, A2A and design pattern will be included in Advance MCQs, basic markdown included

PROMPT ENGINEERING

Discussion

- In this class, we'll focus on prompt engineering.
- We will learn:
  - Technical Depth (Level 01):
    - OpenAI SDK's architecture (agents, tools, handoffs, runner)
    - Pydantic models, async programming, prompt engineering
  - Conceptual topics (Level 02):
    - Dynamic instruction, context management, error handling, chain of thoughts prompting
  - Code Analysis
  - Domain Knowledge

PROMPT ENGINEERING

Discussion

- **Prompt engineering in Agentic AI** refers to the strategic design of prompts to guide autonomous AI agents—such as AI assistants or multi-agent systems—to perform tasks effectively, often with minimal human intervention. Prompt refers to “asking questions”.
- We are discussing OpenAI ChatGPT 4.1 model.
- In today’s world, there is a different approach to do prompt engineering of today’s agentic framework, previously while using generative AI, it didn’t support reasoning and context is not rich.

PROMPT ENGINEERING

Discussion

- LLMs give you 2 returns
  - Either final output
  - Or ask you to call some tool
- Previously, performance of LLMs respect to tool calling is not good.
- OpenAI work on 3 things to improve it and make it better for developers

PROMPT ENGINEERING

Discussion

- LLMs give you 2 returns
  - Either final output
  - Or ask you to call some tool
- Previously, performance of LLMs respect to tool calling is not good.
- OpenAI work on 3 things to improve it and make it better for developers:
  - First, they reached the **contexts**, now we can process up to 1 M token in 1 query, check link for detail https://cookbook.openai.com/examples/gpt4-1_prompting_guide#2-long-context

PROMPT ENGINEERING

Discussion

- Secondly, they improve **tool calling**, OpenAI gives guidelines in tool calling schema like we do things 1) when to used tool, define in system prompt and 2) when work tool will do? How you can use it? What are their examples? Define these in tool schema
  - OpenAI tried to make persistence with the help f prompt engineering, **Persistence** in Agentic AI using the **OpenAI SDK** means the AI can **remember things over time**—even after a task ends or the system restarts.
- COT (**Chain of thoughts**), GPT-4.1 is not a reasoning model, but prompting the model to think step by step (called “chain of thought”) can be an effective way for a model to break down problems into more manageable pieces, solve them, and improve overall output quality

STRUCTURED FLOWCHART REPRESENTATION OF THE GPT-4.1 PROMPTING GUIDE

Start

↓

[Context & Best Practices]

- Use examples (few-shot)
- Be specific & clear
- Encourage planning

↓

[Agentic Workflows?] — Yes ⟶ Use 3 System-Prompt Reminders:

1. Persistence ("keep going…")
2. Tool-calling ("don't guess, call tools…")
3. Planning (optional "plan before calling tools…")

↓

[Workflow Initiation]

- Agent enters multi-message exchange
- Uses tools via API definitions
- Optionally reasons via planning

↓

STRUCTURED FLOWCHART REPRESENTATION OF THE GPT-4.1 PROMPTING GUIDE

[Chain-of-Thought Prompting?] — Yes ⟶ Add “think step by step” cue

↓

[Instruction Literalness]

- GPT-4.1 follows instructions very literally
- Must repeat or restate key rules
- Front-load & back-load instructions

↓

[Common Failure Modes]

- Check for conflicting or ambiguous instructions
- Align examples with rules
- Avoid unnecessary caps or incentives

↓

[Iteration & Evaluation]

- Test prompts with evals
- Migrate old prompts (see “Prompt Migration Guide”)
- Refine and repeat

↓

End

GPT-4.1 PROMPTING GUIDE

Discussion

1. Agentic Workflow:

a) System Prompt Reminder:

i. **Persistence:** Remember stuff overtime, even after a task ends or the system restarts. We make agent persistence using prompt

ii. **Tool calling:** This encourages the model to make full use of its tools, and reduces its likelihood of hallucinating or guessing an answer.

Example:

If you are not sure about file content or codebase structure pertaining to the user's request, use your tools to read files and gather the relevant information: do NOT guess or make up an answer.

GPT-4.1 PROMPTING GUIDE

Discussion

iii. Planning: planning means chain of thoughts, You can ask the model to stop and think before using tools. Instead of just using one tool after another quickly, it will write out a short plan or explanation first –like saying what it wants to do, and why –before actually doing it.

You MUST plan extensively before each function call, and reflect extensively on the outcomes of the previous function calls. DO NOT do this entire process by making function calls only, as this can impair your ability to solve the problem and think insightfully.

GPT-4.1 follows instructions very carefully, especially in agent setups.

- By giving it three clear reminders at the start (about persistence, using tools, and planning), it works much better—about 20% better in tests.
- These reminders help the model act less like a basic chatbot and more like a smart, independent assistant that can move tasks forward on its own.

GPT-4.1 PROMPTING GUIDE

Discussion

b) Tool Calls:

- GPT-4.1 is better at using tools passed through the API.
- So, instead of putting tool details directly into your prompt and writing custom logic, you should use the tools field fields to pass tools — it's cleaner and causes fewer mistakes. In OpenAI's testing, this method worked 2% better than the old way.

Tips for Developers:

- Name your tools clearly so the model understands what they do.
- Write short but clear descriptions for each tool and its parameters.
- If tools are complex, add an # Examples section in your system prompt (not in the description itself).

GPT-4.1 PROMPTING GUIDE

Discussion

- This helps the model know:
  - When to use a tool
  - What to include
  - Which parameters to pick
- You can also use "Generate Anything" in the Prompt Playground to help write tool definitions easily.

GPT-4.1 PROMPTING GUIDE

Discussion

c) Prompting-Induced Planning & Chain-of-Thought

- GPT-4.1 doesn’t naturally “**think step-by-step**,” but you can tell it to plan out loud in your prompt.
- Instead of quickly calling tools one after another, you can prompt it to pause, explain what it’s doing, and then act.
- This is called "**Prompting-Induced Planning**" or using "**Chain-of-Thought**" prompting.
- It’s like making the model explain its thinking before doing something.
- In testing, adding this kind of step-by-step planning made the model perform 4% better on agent tasks.

GPT-4.1 PROMPTING GUIDE

Discussion

d) Sample Prompt: SWE-bench Verified:

- Link for sample prompt ➔ https://cookbook.openai.com/examples/gpt4-1_prompting_guide#sample-prompt-swe-bench-verified

2. Long Context:

GPT-4.1 can handle very long inputs — up to 1 million tokens.

This makes it great for tasks that involve a lot of information, like:

- Reading and understanding big documents
- Picking out the important parts and ignoring the rest
- Re-ranking results (like choosing the best answers)
- Doing multi-step reasoning using info from different parts of the text

In short: GPT-4.1 is good at thinking over long content and making sense of it.

GPT-4.1 PROMPTING GUIDE

Discussion

a) Optimal Context Size

- GPT-4.1 works well even with huge inputs (up to 1 million tokens) — like finding a small detail in a big pile of text (a “needle in a haystack”).
- It’s also good at tasks that mix useful and useless content, like sorting through big documents or code.
- **But there’s a limit:** If the task needs the model to remember and reason about everything at once (like doing a graph search or keeping track of many things across the whole input), its performance can start to drop.

GPT-4.1 PROMPTING GUIDE

Discussion

b) Tuning Context Reliance

- Tuning Context Reliance means deciding how much the model should rely on:
  - 📄 External context (what you give it in the prompt)
  - 🧠 Internal knowledge (what the model already knows)
- 📜 **Two instruction styles:**
  - ***Use ONLY the context given***→ "Only answer using the documents I give you. If it's not there, say you don't know — even if you think you do.
  - ***"Use both context and model knowledge***→ "Use the documents first, but if something's missing and it's basic knowledge you're sure about, go ahead and use it too."

GPT-4.1 PROMPTING GUIDE

Discussion

b) Prompt Organization

Where you put instructions in a long prompt matters.

- ✅ **Best performance:** Put instructions both before and after the context
- 👍 **Second best:** Put instructions before the context only
- ⚠️ **Not ideal:** Putting instructions only after the context

This helps GPT-4.1 follow your instructions better, especially when the prompt is long.

Tip: For long tasks, repeat key rules at the top and bottom to get the best results.

GPT-4.1 PROMPTING GUIDE

Discussion

c) Prompt Organization

Where you put instructions in a long prompt matters.

- ✅ **Best performance:** Put instructions both before and after the context
- 👍 **Second best:** Put instructions before the context only
- ⚠️ **Not ideal:** Putting instructions only after the context

This helps GPT-4.1 follow your instructions better, especially when the prompt is long.

Tip: For long tasks, repeat key rules at the top and bottom to get the best results.