**Prompt Engineering — Detailed Guide**

*Sources used: OpenAI Agents Python docs; prompting guides for Chain-of-Thought and Tree-of-Thought; sampling parameter references.*

**Overview**

This document explains prompt engineering topics , with practical examples and short code snippets that use the OpenAI Agents Python SDK style where applicable.

**1. Sampling parameters: Temperature, top\_k, and top\_p**

What they are:

• **Temperature** — A float (usually between 0 and 1, but can be higher) that controls randomness in token sampling.

Lower values (e.g., 0 or 0.1) make output more deterministic and focused.

higher values (e.g., 0.8, 1.0) increase creativity and diversity but can produce inconsistent results.

• **Top-k** — A sampling strategy that limits the token pool to the top k most likely tokens at each step.

If k=40, the model samples only from the 40 highest-probability tokens.

• **Top-p** (nucleus sampling) — Chooses from the smallest set of tokens whose cumulative probability mass is at least p (e.g., p=0.9). This dynamically adapts the number of candidate tokens and often works well in practice.

Effects and tradeoffs:

• Determinism: temperature=0 and small top\_p (e.g., 0.1) tends to make outputs deterministic and repeatable. Useful for factual or structured tasks.

• Creativity: higher temperature and top\_p close to 1 produce more varied outputs — good for brainstorming, creative writing.

• Safety and hallucination: aggressive randomness can increase hallucinations; use lower randomness for safety-critical outputs.

Practical recommendations:

• For templates, code, or instructions: temperature between 0 and 0.2; top\_p low (e.g., 0.5) or left default.

• For creative tasks: temperature 0.7–1.0, top\_p 0.9–1.0.

Example (OpenAI Agents SDK-style model settings):

```python  
from agents import Agent, ModelSettings  
  
assistant = Agent(

name='Assistant', instructions='You are precise and concise assistant.', model= ModelSettings(temperature=0.2, top\_p=0.9)

)  
```

**2. Safe system messages for sensitive data**

**What are system (developer) messages?**

• System or developer messages set high-level behavior and rules for the assistant. They appear before user messages and can include privacy rules, tone, and security constraints.

**How to use them for sensitive data:**

• Explicitly instruct the model to refuse, redact, or **protect PII (Personally Identifiable Information)** and sensitive content.

Example phrases: 'Do not reveal personal data', 'If a user asks for sensitive personal information, refuse and provide resources', 'Mask or redact names and identifiers unless explicitly authorized.'

• Combine system messages with runtime checks in your application: do not rely solely on the model to enforce legal or compliance requirements. Log and audit requests containing PII and use automated detectors when possible.

Example system message (Agent creation):

```python  
assistant = Agent(  
 name='SecureAssistant',  
 instructions="""You are a security-first assistant.  
 - Never output unredacted PII (names, emails, SSNs).  
 - If asked for sensitive data, refuse and suggest secure alternatives.  
 - Follow company policy: escalate to human review when uncertain."""  
)  
```

**3. Chain of Thought (CoT) prompting**

**What it is:**

Chain of Thought prompting asks the model to show intermediate reasoning steps, improving performance on tasks requiring multi-step logical reasoning (math, puzzles, complex decisions). Rather than only asking 'What is the answer?', you ask 'Show your reasoning and steps.'

**Why it helps**:

• By exposing intermediate steps, the model is guided to perform internal computations and logical progression explicitly, which can increase accuracy on reasoning tasks.

**How to prompt for CoT (few strategies):**

• Direct instruction: 'Explain step-by-step how you arrived at your answer.'

• Few-shot CoT: Provide a couple of examples that include step-by-step reasoning, then ask the new question.

Simple example prompt:

Prompt:  
"""Solve: If there are 3 boxes with 2, 4, and 7 apples respectively, how many apples in total?  
Show your steps."""  
  
Expected model behavior:  
"""Step 1: Add 2 + 4 = 6.  
Step 2: Add 6 + 7 = 13.  
Answer: 13."""

Caveats: Revealing chains of thought can have safety and confidentiality implications (it may reveal internal heuristics). Also, not all models reliably follow CoT prompts—few-shot examples often increase reliability.

**4. Tree of Thoughts (ToT) prompting**

**What it is:**

Tree of Thoughts extends CoT by exploring multiple reasoning paths in parallel. Instead of a single linear chain, the model proposes several 'thoughts' at each step, evaluates them, and expands the most promising branches. This can significantly improve performance on problems that benefit from search and evaluation (planning, complex puzzles).

**How it works (high-level):**

• Propose: At a node, the model generates several candidate thoughts/partial solutions.

• Evaluate: Score or rank those candidates according to heuristics or a learned evaluator.

• Expand: Continue exploration from the top candidates, pruning less

promising branches.

• Terminate and select: When a solution reaches the goal or resource limits are hit, select the best final thought.

Example (conceptual prompt flow):

1) 'Propose 3 possible next steps and briefly justify each.'

2) 'Rank these 3 proposals by likelihood of success.'

3) 'Expand on the top-ranked proposal with 2 more sub-steps.'

Simple illustrative example prompt (planning a 3-step experiment):

Prompt:  
"""You are solving a complex puzzle. Propose 3 different approaches to start, with a one-line justification each. Then rank them from most to least promising."""

Notes: Implementing ToT in production often requires orchestration outside the base LLM—keeping track of candidate states, scoring functions, and budget for exploration.

**5. Practical snippets and tips**

• Example: forcing deterministic answer for structured output:

Use low temperature and strict prompts:  
model\_settings = ModelSettings(temperature=0.0, top\_p=0.1)  
instructions = "Respond in JSON with keys: status, message. Do not add any extra text."

• Example: encouraging creative brainstorming:

Use higher temperature and open-ended prompt:  
model\_settings = ModelSettings(temperature=0.9, top\_p=0.95)  
instructions = "List 10 out-of-the-box product ideas for remote work tools. Be creative and brief."

• Tip: Combine CoT with ToT when appropriate — use CoT for reliable step-by-step verification and ToT when you want to explore multiple distinct solution strategies before selecting one.