PROMPT ENGINEERING · FRAMEWORK FIELD GUIDE

Get the full framework field guide.

28 PAGES

All 11 frameworks + 30 techniques, in print. The 5 Pillars, 6 advanced frameworks, and the complete 6×5 technique matrix — the reference prompt engineering guide CPE alumni keep open at work.

- **01** 5 Pillars of Prompting reference card
- **02** 6 advanced frameworks (CoT, ReAct, RAG…)
- **03** 6×5 technique matrix (30 techniques)
- **04** Module-to-framework crosswalk
- **05** Production patterns + edge cases
- **06** Quick-reference cheat sheet

Issued by the Global Skill Development Council — Certified Prompt Engineer (CPE) program companion.

ORIENTATION

How to use this guide

This field guide is built for reference, not for reading once. Keep it open beside your editor. Every framework is presented the same way — what it is, when to reach for it, the shape of the prompt, and the failure mode to watch.

The structure

How to read each entry

WHAT A one-line definition in plain language.

WHEN The signal that tells you to reach for it.

SHAPE The skeleton of the prompt.

WATCH The most common way it breaks.

SECTION A

The 5 Pillars of Prompting

Every reliable prompt rests on the same five load-bearing elements. Drop one and quality wobbles; the Pillars are what you tune first, before reaching for anything advanced.

Context 1

Who the model is, what world it is operating in, and the background it needs.

Instruction 2

The single, unambiguous task — the verb the model must execute.

Constraints 3

The boundaries: length, tone, scope, what to avoid, what must be true.

Examples 4

Demonstrations that show, rather than tell, the target behaviour.

Output Format 5

The exact structure the answer must take so it is usable downstream.

Rule of thumb. If a prompt is failing, diagnose the Pillars in order. Most production failures are a missing constraint or an under-specified output format — not a missing technique.

5 PILLARS

Pillar 1 — Context

WHAT The role, audience, and background the model needs to answer as the right "person" for the right reader.

WHEN Always — but it matters most when the task is domain-specific, or the default voice is wrong.

SHAPE Assign a role, set the audience, supply only the facts the task requires.

WATCH Context bloat. Irrelevant background dilutes attention and burns tokens.

CONTEXT PATTERN

```
You are a senior tax analyst writing for first-time founders.
Audience: non-financial, time-poor.
Background: US C-corp, pre-revenue, Delaware incorporated.
```

Tighten it. Prefer a short, high-signal context block over a long biography. Three precise lines beat three paragraphs.

5 PILLARS

Pillar 2 — Instruction

WHAT The one task you want done, stated as a clear action verb with a defined object.

WHEN Every prompt. This is the Pillar that most often hides two tasks pretending to be one.

SHAPE One verb, one deliverable. Split compound asks into a sequence.

WATCH Ambiguous verbs ("handle", "deal with", "look at") that the model is free to interpret.

WEAK vs STRONG

```
WEAK:   Look at this contract and tell me about it.
STRONG: Extract every payment obligation from this contract.
        Return one row per obligation with amount and due date.
```

One verb test. If you can't underline a single primary verb, the instruction is doing too much.

5 PILLARS

Pillar 3 — Constraints

WHAT The guardrails: scope, length, tone, sources, and the explicit "do not" list.

WHEN Whenever an answer could drift — too long, too casual, out of scope, or invented.

SHAPE Positive constraints (must) and negative constraints (must not), stated separately.

WATCH Conflicting constraints. "Be exhaustive" and "keep it to 3 bullets" cannot both win.

CONSTRAINT BLOCK

```
- Max 120 words.
- Plain English, no jargon.
- Use only the figures in the source; do not estimate.
- If a figure is missing, write "not provided".
```

Negative space matters. The 'do not estimate / say not provided' line is what prevents hallucinated numbers.

5 PILLARS

Pillar 4 — Examples

WHAT Demonstrations of the target behaviour — the model imitates patterns far better than it follows abstract descriptions.

WHEN When format is hard to describe, edge handling matters, or tone must be exact.

SHAPE One to five input→output pairs. Include at least one tricky case.

WATCH Skewed examples. If all your examples are easy, the model fails on the hard ones.

FEW-SHOT PATTERN

```
Input:  "order shipped late, want refund"
Label:  {"intent":"refund","sentiment":"negative"}

Input:  "loved it, will buy again!"
Label:  {"intent":"praise","sentiment":"positive"}
```

Show the edge. Add a contrastive pair (a near-miss labelled correctly) to teach the boundary, not just the centre.

5 PILLARS

Pillar 5 — Output Format

WHAT The exact structure the answer must take so a human — or a program — can use it without cleanup.

WHEN Any time the output is consumed downstream: parsed, stored, displayed, or chained.

SHAPE Specify the schema and a literal template; for machines, demand strict JSON.

WATCH Prose leaking around the JSON ("Sure! Here is…") which breaks parsers.

FORMAT SPEC

```
Return ONLY valid JSON, no prose, matching:
{ "summary": string, "risks": string[], "score": 0-100 }
```

Make it parseable. "Return only JSON, no preamble" is one of the highest-leverage lines you can add to a production prompt.

TEAR-OUT

5 Pillars — Reference Card

The one-glance summary. Diagnose failing prompts top to bottom.

#	Pillar	Question it answers	Fix when…
1	**Context**	Who am I, for whom?	tone or domain is wrong
2	**Instruction**	What exactly do I do?	output answers a different question
3	**Constraints**	What are the limits?	too long, off-scope, invented facts
4	**Examples**	What does good look like?	format or edge cases are off
5	**Output Format**	What shape do I return?	result needs manual cleanup

The assembled prompt

ALL FIVE, IN ORDER

```
[Context]      You are a ... writing for ...
[Instruction]  Extract ... / Draft ... / Classify ...
[Constraints]  Max N words. Use only the source. Do not ...
[Examples]     input -> output (x2, incl. one edge)
[Output]       Return ONLY JSON matching { ... }
```

SECTION B

6 Advanced Frameworks

Once the Pillars are solid, these six frameworks handle reasoning, knowledge, action and reliability. They compose — RAG feeds ReAct; CoT lives inside Self-Consistency.

Chain-of-Thought 1

Make the reasoning explicit, step by step, before the answer.

ReAct 2

Interleave reasoning with tool actions and observations.

RAG 3

Retrieve external knowledge and ground the answer in it.

Tree-of-Thought 4

Explore several reasoning branches, then pick the best.

Meta-Prompting 5

Use the model to write and refine the prompt itself.

Self-Consistency 6

Sample many reasonings; let the majority answer win.

They stack. Most real systems are combinations: RAG + CoT for grounded reasoning, ReAct + Reflexion for self-correcting agents.

ADVANCED

Framework 1 — Chain-of-Thought (CoT)

WHAT Ask the model to reason in explicit intermediate steps before committing to a final answer.

WHEN Multi-step logic: maths, planning, diagnosis, anything where a leap-to-answer is error-prone.

SHAPE "Think step by step", or supply a worked example showing the reasoning trace.

WATCH Verbose, confident-but-wrong chains. Reasoning shown is not reasoning verified.

ZERO-SHOT CoT

```
Q: A shelf holds 3 boxes of 12; you remove 7 items. How many remain?
Let's reason step by step, then give the final number on its own line.
```

Hide it in production. Keep the chain for accuracy, but request the trace in a field you can discard, returning only the final answer to the user.

ADVANCED

Framework 2 — ReAct

WHAT A loop of *Thought* → *Action* → *Observation*: the model reasons, calls a tool, reads the result, and continues.

WHEN The answer needs live data or tools — search, code, a database, an API.

SHAPE Define the tools, then prompt the model to alternate reasoning and tool calls until done.

WATCH Loops that never terminate; always cap the number of steps.

ReAct TRACE

```
Thought: I need the current FX rate.
Action:  get_rate("USD","INR")
Observation: 83.2
Thought: Now multiply 500 x 83.2.
Answer:  41,600 INR
```

Cap the loop. Set a max-step limit and a fallback answer so a failing tool can't trap the agent.

ADVANCED

Framework 3 — RAG

WHAT Retrieval-Augmented Generation: fetch relevant documents, then have the model answer using only them.

WHEN Private, current, or large knowledge that cannot live in the prompt or the weights.

SHAPE Retrieve → rank → inject top chunks → instruct "answer only from context, cite sources".

WATCH Retrieval misses. If the right chunk isn't fetched, no prompt can save the answer.

GROUNDED ANSWER

```
Use ONLY the context below. Cite the [doc-id] for each claim.
If the answer is not in the context, reply: "Not in the provided sources."

Context:
[doc-12] ...
[doc-31] ...
```

Fix retrieval first. When RAG is wrong, the bug is usually in chunking or ranking — not in the generation prompt.

ADVANCED

Framework 4 — Tree-of-Thought (ToT)

WHAT Generate several candidate reasoning paths, evaluate them, and expand the most promising branch.

WHEN Problems with many viable approaches: puzzles, strategy, design, planning under uncertainty.

SHAPE Propose N options → score each → keep the best → deepen → repeat.

WATCH Cost. ToT multiplies token spend; reserve it for genuinely hard problems.

BRANCH + SELECT

```
Propose 3 distinct strategies for X.
Rate each 1-10 for feasibility and risk.
Expand only the highest-rated strategy in detail.
```

Breadth then depth. ToT is CoT with branching and pruning — use it when a single chain keeps dead-ending.

ADVANCED

Framework 5 — Meta-Prompting

WHAT	Use the model to design, critique, or rewrite a prompt — a prompt that produces prompts.
WHEN	You are iterating on prompt quality at scale, or onboarding a new task type.
SHAPE	"Here is my task and a draft prompt; improve it and explain each change."
WATCH	Over-engineering. Meta-prompting can inflate a simple prompt into a brittle essay.

PROMPT THAT WRITES PROMPTS

```
Act as a prompt engineer. Given this goal: ,
write a production prompt using the 5 Pillars.
Then list 3 ways it could fail and patch each.
```

Keep a human gate. Meta-prompting drafts fast — but a person should still approve the prompt that ships.

ADVANCED

Framework 6 — Self-Consistency & Reflexion

WHAT Two reliability moves: sample multiple reasonings and vote (self-consistency); or let the model critique and retry its own answer (reflexion).

WHEN High-stakes answers where a single pass is too risky.

SHAPE Run N times → take the majority; or generate → self-critique → revise once.

WATCH Diminishing returns and cost — more samples help only up to a point.

REFLEXION LOOP

```
1) Draft an answer.
2) Critique it against the constraints; list defects.
3) Produce a corrected final answer addressing each defect.
```

Vote then verify. Self-consistency raises accuracy on reasoning tasks; pair it with a hard check on the final format.

SECTION C

The 6×5 Technique Matrix

Thirty named techniques, organised into six families of five. The frameworks tell you the strategy; the matrix gives you the moves.

Family	5 Techniques
1 Structuring	Role/persona · Delimiters · Schema spec · Priming · Task decomposition
2 Reasoning	CoT · Zero-shot CoT · Tree-of-Thought · Self-consistency · Least-to-most
3 Examples	Zero-shot · One-shot · Few-shot · Example ordering · Contrastive pairs
4 Grounding	RAG · Citation enforcement · Chunking · Re-ranking · Grounded refusal
5 Action	ReAct · Tool calling · Plan-and-execute · Reflexion · Multi-agent
6 Control	Sampling control · Guardrails · Self-check · Meta-prompting · Prompt chaining

Read it as coordinates. Pick the family for the problem, then the technique for the symptom. Detailed cards follow.

30 TECHNIQUES (1–10)

Matrix — Structuring & Reasoning

Role / Persona
Assign an expert identity to set voice and depth.

Delimiters
Fence inputs with markers so the model never confuses data for instructions.

Schema spec
State the exact output keys and types up front.

Priming
Open with a short frame that sets expectations for the whole reply.

Task decomposition
Break one big ask into an ordered list of smaller asks.

Chain-of-Thought
Request explicit intermediate reasoning steps.

Zero-shot CoT
Trigger reasoning with 'think step by step' — no examples.

Tree-of-Thought
Branch into options, score, and prune.

Self-consistency
Sample multiple chains; take the majority answer.

Least-to-most
Solve easy sub-problems first; build up to the hard one.

30 TECHNIQUES (11–20)

Matrix — Examples & Grounding

Zero-shot
No examples — rely on a precise instruction.

One-shot
A single demonstration to anchor format.

Few-shot
Several pairs to teach a pattern and its edges.

Example ordering
Place the most representative example last.

Contrastive pairs
Show a near-miss labelled correctly to mark the boundary.

RAG
Inject retrieved documents as grounding context.

Citation enforcement
Require a source tag on every factual claim.

Chunking
Split sources into retrievable, coherent units.

Re-ranking
Reorder retrieved chunks by true relevance before use.

Grounded refusal
Answer only from context; otherwise decline.

30 TECHNIQUES (21–30)

Matrix — Action & Control

ReAct
Interleave reasoning with tool calls and observations.

Tool calling
Expose typed functions the model can invoke.

Plan-and-execute
Draft a full plan, then carry out each step.

Reflexion
Self-critique and retry to fix the first answer.

Multi-agent
Split roles across cooperating model instances.

Sampling control
Tune temperature / top-p for determinism or variety.

Guardrails
Hard rules and refusals encoded into the prompt.

Self-check
Have the model validate its own output before returning.

Meta-prompting
Use the model to generate and refine prompts.

Prompt chaining
Pipe one prompt's output into the next stage.

SECTION D

Module-to-Framework Crosswalk

How the CPE curriculum maps onto the frameworks and techniques in this guide. Use it to find the lesson behind any move.

CPE Module	Frameworks	Matrix families
M1 · Foundations	5 Pillars	Structuring, Examples
M2 · Reasoning	CoT, ToT, Self-Consistency	Reasoning
M3 · Agentic AI	ReAct, Reflexion	Action
M4 · Knowledge & Retrieval	RAG	Grounding
M5 · Meta & Optimisation	Meta-Prompting	Control
M6 · Production & Eval	All, applied	Control, Grounding

Two-way map. Stuck on a framework? Find its module for the full lesson. Studying a module? This is your at-a-glance index.

SECTION D

Crosswalk — Skills & Outcomes

What each module proves you can do — the language to put on a résumé or a project brief.

Module	You can demonstrate…
M1	Building reliable prompts from the 5 Pillars and diagnosing failures by Pillar.
M2	Selecting the right reasoning strategy and hiding chains in production.
M3	Designing bounded ReAct agents with tools, step caps and fallbacks.
M4	Standing up a grounded RAG flow with citation and refusal behaviour.
M5	Using meta-prompting and chaining to optimise prompts at scale.
M6	Shipping with templates, evals, guardrails and edge-case coverage.

Portfolio-ready. Each row is a project you can build and a line you can defend in an interview.

SECTION E

Production Patterns

The scaffolding around the prompt that turns a clever demo into a dependable system.

Templating & versioning

Store prompts as versioned templates with slots, not hard-coded strings.

System / user split

Keep durable rules in the system role; put per-request data in the user role.

Schema + validation

Demand JSON, then validate it; reject and retry on a parse failure.

Retry & fallback

On failure, retry with a stricter prompt, then fall back to a safe default.

Caching & cost

Cache stable sub-results; trim context to control token spend.

Eval harness

Keep a golden set; score every prompt change before it ships.

SECTION E

Production Patterns — In Practice

A hardened prompt skeleton

TEMPLATE

```
[SYSTEM] role + non-negotiable rules + output contract
[USER]   {context}  {task}  {data, fenced by ===}
[POST]   validate JSON -> on fail, retry strict -> else default
```

Checklist before you ship

- ✓ Output contract is explicit and machine-validated.
- ✓ Every external input is delimited and escaped.
- ✓ Failure path defined: retry, fallback, and a human alert.
- ✓ Prompt is versioned and covered by the eval set.
- ✓ Token budget and latency ceiling are measured.

Treat prompts as code. Version them, test them, and review changes — the same discipline you give any production dependency.

SECTION F

Edge Cases & Failure Modes

The ways prompts break in the wild — and the move that contains each one.

Failure	Symptom	Containment
Hallucination	Confident, invented facts	Ground with RAG; "say if unknown"
Prompt injection	Input overrides your rules	Delimit input; never trust it as instruction
Context overflow	Earlier content forgotten	Summarise, chunk, or trim
Instruction conflict	Ignores one of two rules	Rank rules; remove contradictions
Format drift	Extra prose around JSON	"Return only JSON"; validate + retry

SECTION F

Failure Modes — Continued

Failure	Symptom	Containment
Over-refusal	Declines safe requests	Clarify intent; tighten scope wording
Under-specification	Plausible but off-target	Add constraints and an example
Latency / cost blowup	Slow, expensive calls	Cache, trim context, cap reasoning
Looping agent	Never terminates	Step cap + forced final answer
Stale knowledge	Out-of-date facts	Retrieve current sources at run time

Diagnose in order. Most incidents trace back to a missing Pillar — usually a constraint or an output contract. Start there.

SECTION G

Quick-Reference Cheat Sheet

Reach for…

If you need…	Use
Reliable basics	The 5 Pillars, in order
Step-by-step logic	Chain-of-Thought
Live data or tools	ReAct + tool calling
Private / current knowledge	RAG + grounded refusal
Hard, branching problems	Tree-of-Thought
Higher accuracy	Self-consistency / Reflexion
Better prompts, faster	Meta-prompting
Production reliability	Schema + validate + retry

THE WHOLE GUIDE, IN ONE VIEW

Keep this open at work

11 frameworks. 30 techniques. One workflow.

5 Pillars	Context · Instruction · Constraints · Examples · Output
6 Advanced	CoT · ReAct · RAG · ToT · Meta · Self-Consistency
6×5 Matrix	Structuring · Reasoning · Examples · Grounding · Action · Control
Crosswalk	Every framework mapped to a CPE module
Production	Templates · validation · evals · guardrails
Edge cases	Diagnose by Pillar, contain by pattern