A PRACTITIONER'S SUMMARY

5 SLIDES · ~6 MIN READ

PE—01

Prompt *Engineering*

The craft of designing inputs that guide large language models toward accurate, useful, and reliable outputs.

Distilled from the whitepaper by Lee Boonstra · Google, Feb 2025 68 pages · synthesised to 5 slides

01 / FOUNDATIONS

What is *prompt engineering?*

An LLM is a prediction engine. A prompt is your attempt to set it up to predict the right sequence of tokens . Prompt engineering is the iterative process of designing those inputs — tuning length, style, structure, and context — until the model produces what you actually want.

> *"You don't need to be a data scientist or a machine learning engineer — everyone can write a prompt."*
> — LEE BOONSTRA

A

Iterative

Tinker, evaluate, refine. The first prompt is almost never the last.

B

Model-specific

A prompt tuned for one model may need rework on another.

C

Configurable

Prompt and sampling settings work together — you tune both.

02 / THE KNOBS

Configure the model *before* you blame the prompt.

01 / LIMITS

Output *length*

Caps the response in tokens. Stops generation; does not make output more concise. Critical for ReAct-style loops.

02 / RANDOMNESS

Temperature

Controls how greedy sampling is. 0 = deterministic best token. Higher = more diverse, more surprising, more creative.

03 / SHORTLIST

Top-*K*

Sample only from the top K most likely next tokens. Low K is focused and factual; high K opens the door to variety.

04 / SHORTLIST

Top-*P*

Nucleus sampling. Keep the smallest set of tokens whose cumulative probability reaches P. Adapts to context.

Rule of thumb. Start at temperature 0.2, top-P 0.95, top-K 30 for balanced tasks — 0 for extractive, 0.9+ for creative.

MORE TOKENS → MORE COST, MORE LATENCY

03 / THE REPERTOIRE

Eleven prompting techniques, *four families.*

I · SHOTS

01

Zero-shot

Just the task. The baseline.

02

One-shot

A single example to anchor the pattern.

03

Few-shot

3–5 diverse examples, including edge cases.

II · FRAMING

04

System

Sets purpose and output contract (e.g. JSON).

05

Role

Assigns a persona — tone, voice, expertise.

06

Contextual

Task-specific background the model needs now.

III · REASONING

07

Step-back

Answer a broader question first, then the real one.

08

Chain of Thought

"Think step by step." Show the working.

09

Self-consistency

Sample many CoT paths, take the majority answer.

10

Tree of Thoughts

Branch & backtrack through reasoning states.

IV · ACTING

11

ReAct

Interleave reasoning with tool calls and observations.

+

Automatic PE

Let the model write & rank candidate prompts for you.

+

Code prompting

Write, explain, translate, debug — same techniques apply.

04 / IN PRACTICE

Chain of Thought, *in the wild.*

THE PROMPT — USER

INPUT

When I was 4 years old, my partner was 3 times my age. Now, I am 20 years old. How old is my partner?

→ Let's think step by step.

WHY IT WORKS

Five extra words—*"Let's think step by step"* —flip the model from guessing a final number to showing the working . Each intermediate step is grounded in the previous one, so arithmetic and multi-hop reasoning land far more often.

THE REASONING — MODEL

1 Partner's age when I was 4
My partner was 3× my age back then.
4 × 3 = 12

2 The age gap (it's fixed for life)
Difference between our ages doesn't change.
12 − 4 = 8

3 Add the gap to my current age
I'm 20 now, partner is 8 years older.
20 + 8 = 28

ANSWER

28 years old

05 / HABITS

What separates a *working prompt* from a great one.

08

BEST PRACTICES

Most failed prompts fail on these. None of them are clever — they are discipline.

01 **Provide examples**
The single highest-leverage move. Diverse, high quality, edge cases included.

02 **Design with simplicity**
If the prompt confuses you, it confuses the model. Cut, don't pile on.

03 **Be specific about the output**
Length, format, tone, schema. Say what "done" looks like.

04 **Instructions over constraints**
Tell it what to do. "Don't" lists are brittle and easy to break.

05 **Control max tokens**
Cap output to your need. Saves cost, latency, and rambling tails.

06 **Use variables, not hard-codes**
Template the prompt. Swap inputs without rewriting the whole thing.

07 **Mix classes in few-shot**
Don't group examples by label — the model will overfit to the order.

08 **Document every attempt**
Log prompt, model, config, output. PE without notes is just guessing.