FREE GUIDE · BULDRR AI · 2026

How Claude Killed Prompt Engineering

Three moves Anthropic made in one week — Record a Skill, Opus 5, and Context Engineering — and what they mean for how you use AI from here.

Record a Skill | Claude Opus 5 | Context Engineering

3
MOVES THAT CHANGED EVERYTHING

80%
OF SYSTEM PROMPT STRIPPED

0
PROMPTS NEEDED TO TEACH CLAUDE A SKILL

CREATED BY

Vikash Kumar

linkedin.com/in/vikashyavansh

WHAT HAPPENED

Three moves. One week. One direction.

For two years, writing the perfect prompt was the skill everyone told you to learn. Longer prompts. More context. Better instructions. This week, Anthropic shipped three things that say the opposite direction is right.

THE IRONY

Everyone spent two years learning how to write longer, more detailed prompts. Anthropic just shipped three things that say the opposite direction is right. The skill is not dead yet — but it is becoming a transitional one, not a permanent one.

1 Record a Skill

Claude learns tasks by watching you — no prompting required

Instead of writing instructions for Claude, you now show it. Record your screen, explain each step out loud, and Claude turns the recording into a reusable skill it can execute on its own later.

How to use it

1. **Open Claude Desktop**
   Go to the Claude desktop app. This feature is in the Cowork tab, not the standard chat interface.
2. **Click the + button and select Record a Skill**
   This starts the screen recording session. Claude begins watching.
3. **Do the task normally and explain as you go**
   Talk through what you are doing and why. Claude uses your narration to understand the intent behind each action, not just the steps themselves.
4. **Stop recording**
   Claude processes the recording and creates a reusable skill. It names the skill, writes the steps, and stores it for future use.
5. **Run the skill later**
   The next time you need the same task done, trigger the skill and Claude executes it. No re-explaining. No new prompt.

WHAT TO RECORD FIRST

Start with a repetitive task you do at least twice a week — formatting a report, processing a client email, updating a spreadsheet. The more repetitive the task, the more value the skill saves.

2 Claude Opus 5

Performs better with less instruction — not more

Claude Opus 5 came within half a percentage point of Claude Fable 5 on coding benchmarks at roughly half the cost. But the more interesting finding is how it handles prompts. Detailed, heavily structured prompts did not outperform minimal ones. In many tests, they performed worse.

Opus 5 Prompting Guide — what works now

OLD APPROACH (LESS EFFECTIVE)

- → Long system prompts with detailed rules
- → Numbered instruction lists with conditions
- → Telling Claude exactly how to think
- → Prescribing every output format
- → Hedging and disclaimers in your prompt

NEW APPROACH (MORE EFFECTIVE)

- → Clear goal, minimal constraints
- → Trust the model to handle the steps
- → Give intent, not procedure
- → Ask for what you want, not how to do it
- → Iterate on output, not on instruction length

OLD PROMPT — LESS EFFECTIVE WITH OPUS 5

```
# You are a professional email writer. Your task is to write a follow-up email.
# Follow these rules exactly:
# 1. Keep the email under 150 words
# 2. Use a professional tone but not overly formal
# 3. Reference the previous meeting
# 4. Include a clear call to action
# 5. Do not use phrases like "I hope this finds you well"
# 6. End with "Best regards"

Write a follow-up email to a client after a product demo.
```

NEW PROMPT — MORE EFFECTIVE WITH OPUS 5

```
Write a short follow-up email to a client after a product demo.
Goal: get them to book a next step meeting.
Tone: warm but direct.
```

> **The principle:** Opus 5 was trained to understand intent. The more you constrain how it thinks, the less it can apply that training. Give it the destination, not the route.

3 Context Engineering

Anthropic stripped 80% of Claude Code's system prompt — it ran better

Anthropic published a post on context engineering for the Claude 5 generation. The core finding: instead of building large, detailed system prompts, the newer models respond better to minimal, well-structured context. They removed over 80% of Claude Code's original system prompt and saw faster, smarter performance.

The 6 principles of context engineering

PRINCIPLE 1

Give goals, not procedures

Tell Claude what success looks like. Let it figure out how to get there. Prescribing every step reduces its ability to adapt.

PRINCIPLE 2

Remove rules that hedge

If a rule in your system prompt starts with "try to" or "if possible," remove it. These weaken the model's confidence without adding value.

PRINCIPLE 3

Use examples, not explanations

One example of the output you want is worth ten sentences explaining what you want. Show the model, do not describe the model.

PRINCIPLE 4

Keep context relevant

Every sentence in your context window that is not directly relevant to the current task reduces the model's focus on what matters.

PRINCIPLE 5

Let the model self-check

Opus 5 and Claude 5 generation models can verify their own work. Ask them to check the output before returning it instead of writing validation rules yourself.

PRINCIPLE 6

Start minimal, add only what fails

Begin with the shortest possible system prompt. Only add a rule when you observe a specific failure. This keeps your context lean and effective.

BEFORE CONTEXT ENGINEERING — CLAUDE CODE'S ORIGINAL SYSTEM PROMPT (SIMPLIFIED)

```
# You are Claude Code, an AI coding assistant. You help users write, debug,
# and review code. Always follow these guidelines:
# - Write clean, readable code
# - Add comments to complex logic
# - Explain your changes
# - Ask for clarification when requirements are unclear
# - Follow the user's existing code style
# - Suggest tests when relevant
# - Handle errors gracefully
# [... 80% more rules like this ...]
```

AFTER CONTEXT ENGINEERING — WHAT REMAINED (SIMPLIFIED)

```
You are Claude Code. Help the user build working software.

When you make changes, explain what you changed and why.
When you are unsure, ask one clarifying question before proceeding.
```

WHAT IT MEANS FOR YOU

What to change in how you use Claude

1. **Audit your system prompts**

   Go through every system prompt or CLAUDE.md you have. Remove every rule that starts with "try to," "if possible," "generally," or "where relevant." These are hedges. They reduce clarity without adding control.

2. **Replace rules with examples**

   For every formatting or style rule you have written, replace it with one example of the output you want. Claude will infer the rule from the example better than from the description.

3. **Record your first skill this week**

   Pick one task you do more than twice a week. Record it in Claude Desktop. The time investment is under 10 minutes and the skill runs automatically from that point forward.

4. **Switch to Opus 5 for complex tasks**

   If you are currently using Claude Fable 5 on the Max plan for complex coding, analysis, or multi-step tasks, test Opus 5. It is within half a percentage point on performance at roughly half the API cost.

5. **Stop writing longer prompts**

   The instinct when Claude underperforms is to add more rules. With the Claude 5 generation, the better instinct is to remove rules and give a clearer goal. Try the minimal version first.

The skill is changing. Get ahead of it.

Prompt engineering is not dead yet. But context engineering, skill recording, and minimal instruction design are where the best Claude users are already moving.

1. Audit your system prompts — remove every hedge and "try to"
2. Record your first repetitive task as a Claude skill this week
3. Switch to goal-first prompts — give destination, not route
4. Test Opus 5 on your complex tasks and compare the cost

Vikash Kumar

Founder, BULDRR AI · Claude Code and n8n Automation Expert · $1.44M saved for clients

Connect on LinkedIn

$1.44M	20K+	28mo
SAVED FOR CLIENTS	LINKEDIN FOLLOWERS	BUILDING DAILY