CLAUDE CHEAT CODES

The Complete Power User Reference Guide

Every verified prompt code, hidden feature, and advanced technique for Claude.ai, the API, and Claude Code — compiled from Anthropic docs, community testing, and real-world usage.

NOTE: Codes marked with a star are the highest-impact techniques confirmed by multiple independent testers.

1. Verified Prompt Codes

These shorthand codes modify Claude's behavior by signaling clear intent. They are not backdoors — they work because Claude recognizes the conventions behind them. Around 15 have been independently confirmed to produce consistent effects.

`L99`	Forces committed expert opinions. You get real recommendations with specific tradeoffs instead of balanced 'it depends' answers.
`/ghost`	Strips all AI-detectable patterns — em-dashes, 'I hope this helps,' robotic phrasing. Output reads as human-written.
`ULTRATHINK`	Maximum reasoning depth. Produces deep analyses with multiple layers and zero hedging. In Claude Code, triggers extended thinking.
`PERSONA: [specific]`	Assign a hyper-specific role: 'Senior Postgres DBA at Stripe, 15 years, cynical about ORMs.' Specificity grants real opinions.
`/skeptic`	Challenges your question before answering. Catches the 'wrong question' problem before you waste time on the wrong path.
`OODA`	Triggers the Observe-Orient-Decide-Act military framework for structured strategic analysis.
`/noyap`	Extreme brevity mode. All filler, caveats, and padding eliminated from the response.
`/punch`	Tightens and sharpens prose. Great for emails, headlines, and copy.

Effective Combo Stacks

- **/ghost + /punch** — Humanize + shorten. Perfect for cold emails and outreach.
- **PERSONA + L99** — Expert + committed opinion. Best for technical decisions.
- **/skeptic + ULTRATHINK** — Right question + max depth. Ideal for strategy work.

WARNING: What does NOT work: /nofilter, /unlocked, L33T, DAN-mode prompts. These are ChatGPT jailbreak attempts with zero effect on Claude.

2. XML Prompting

Claude was literally trained on XML-structured prompts. Its own system prompt uses XML tags internally. This makes XML the single highest-impact formatting technique.

Essential Tags to Adopt

- **<instructions>** — What you want done
- **<context>** — Background information and relevant details
- **<examples>** — Wrap individual <example> tags for few-shot demos
- **<formatting>** — Output format specifications
- **<document>** — With <source> subtags for multi-document work
- **<thinking> / <answer>** — Separate reasoning from final output

Key Rules

1. **Documents go at the TOP, queries at the BOTTOM**
   Anthropic testing shows up to 30% quality improvement for inputs over 20K tokens.

2. **Tag names don't matter — consistency does**
   Use descriptive labels that match your content. Stay consistent across prompts.

3. **Nest tags hierarchically for complex content**
   Combine XML with few-shot examples, thinking tags, and output format control.

4. **Remove markdown from your own prompt**
   Claude mirrors your register. Conversational input = verbose output. Direct input = direct output.

PRO TIP: To kill excessive formatting, add this tag to your system prompt: <avoid_excessive_markdown_and_bullet_points> with instructions to use flowing prose and reserve markdown for code blocks only.

3. Thinking & Reasoning

Chain-of-Thought Levels

1. **Basic CoT**
   Add 'Think step-by-step' to your prompt.
2. **Guided CoT**
   Outline specific reasoning steps for Claude to follow.
3. **Structured CoT**
   Use <thinking> and <answer> XML tags to separate reasoning from output.

Extended & Adaptive Thinking

- **Adaptive thinking** (Claude 4.6) is now recommended over manual budgeting — Claude calibrates depth automatically.
- Extended thinking can **hurt performance by up to 36%** on simple/intuitive tasks. Use only for math, logic, architecture, and debugging.
- **Haiku 4.5 + thinking enabled** unlocks near-Sonnet performance at lower cost.
- Set max output tokens to **64K** at medium/high effort for best results.

Claude Code Thinking Triggers

`"think"`	Baseline extended reasoning
`"think harder"`	Bridges intelligence gap between models
`"ultrathink"`	Maximum reasoning depth (Claude Code only)

Few-Shot Examples

- Include **3-5 diverse examples** for best results (90% format success rate vs. instructions alone).
- Wrap in <example> tags. Include <thinking> tags in examples to show desired reasoning.
- Claude 4.x replicates naming conventions, code style, formatting, and punctuation from examples.

Effort Parameter (API)

- **Low** — Fast, similar to older models. Good for simple lookups.
- **Medium** — Routine coding, tool-heavy workflows.
- **High** — Complex reasoning, autonomous agents (default on Sonnet 4.6).

4. Hidden Features in Claude.ai

Memory — Three Layers

1. **Chat Memory (free for all users)**
   Claude summarizes conversations daily. View/edit in Settings. Imports from ChatGPT, Gemini, and Grok are supported.
2. **Claude Code Memory**
   CLAUDE.md files loaded every session. Auto-memory saves debugging insights. 'Auto Dream' consolidates memory between sessions.
3. **API Memory Tool**
   Persistent cross-session memory for custom applications built on the API.

Projects

- Now **free for all users** (5 projects on free tier, unlimited on paid).
- Each project has its own memory, custom instructions, and knowledge base.
- When knowledge approaches context limits, Claude auto-enables **RAG mode** (up to 10x capacity).
- Move chats between projects. Convert Claude's output back into project knowledge.

Styles & Personalization

- **Profile Preferences** (Settings) apply to all conversations globally.
- **Project Instructions** apply per-project for specialized context.
- **Custom Styles** control tone/format per conversation. Upload writing samples to match your voice.

Other Hidden Features

- **Incognito Chats** — Cmd+Shift+I. No memory, no history. Great for sensitive queries.
- **Connectors Directory** — 150+ integrations (Drive, Gmail, Slack, GitHub, Notion, Figma, etc.) now available to all users.
- **Memory Import** — Settings > Capabilities > Memory Import to transfer from other AI assistants.

5. Claude Code Power Commands

Essential Commands

`/init`	Auto-generate a CLAUDE.md from your codebase
`/compact`	Summarize conversation to free context. Add focus: /compact Focus on API changes
`/clear`	Reset context between tasks. Fresh context beats degraded sessions
`/fork`	Branch the session to try different approaches without losing original
`/btw`	Quick overlay question that never enters conversation history
`Escape`	Stop mid-response. Double-Escape shows all previous messages to jump back to
`Shift+Tab`	Cycle between Normal, Auto-Accept, and Plan mode

CLAUDE.md Best Practices

- Keep under **500 lines** with 20-30 lines of essentials at top.
- 'Point, don't dump' — reference where detailed info lives rather than pasting everything.
- Claude follows ~150-200 instructions reliably. System prompt uses ~50 of those slots already.
- Run **/init** to auto-generate a starting file from your codebase.

Parallel & Advanced Workflows

- **Parallel instances** — Run 2-3+ Claude sessions in different terminal panes simultaneously.
- **claude --worktree** — Git worktree-isolated sessions for safe parallel work.
- **Subagents** — 'Use subagents to investigate X' keeps research costs out of main context.
- **Custom slash commands** — Place .md files in .claude/commands/[name].md with $ARGUMENTS.
- **Routines** (April 2026) — Schedule repeating automations that run on Anthropic's servers.
- **--dangerously-skip-permissions** — Eliminates constant file edit confirmations.
- **/install-github-app** — Auto-review PRs with customizable prompts.

6. Behavior Control & Anti-AI Writing

Claude 4.x Behavioral Shift

> **KEY CHANGE:** Claude 4.x takes you literally and does exactly what you ask — nothing more. Earlier models inferred intent. Now you must explicitly request 'above and beyond' behavior.

Eliminate Hedging

- Assign confident personas with specific backgrounds
- Use the L99 prefix for committed expert opinions
- Add 'Do not hedge' explicitly to instructions
- Try confidence calibration: 'Rate confidence 1-10 and explain why'

Reduce Verbosity

- Write terse prompts yourself — Claude mirrors your register
- Set explicit constraints: 'Respond in under 100 words'
- Use DO/DO NOT lists for precise behavior control
- Define exit conditions: 'Stop after the signature'

Anti-AI Word Ban List

Add these to your system prompt or project instructions to eliminate robotic writing patterns:

```
utilize, synergy, leverage, foster, delve, tapestry, testament, showcase, pivotal,
underscore, embark, navigate, landscape, realm, crucial, robust, cutting-edge,
revolutionize, empower, harness
```

Reduce Hallucinations

- For long documents (20K+ tokens): ask Claude to extract word-for-word quotes before analysis
- Require citations for each claim
- Restrict to provided docs: 'Use only the information above. Do not rely on general knowledge.'
- Run the same prompt multiple times and compare outputs for critical work

7. API Cost Optimization

Prompt Caching

- Cache hits cost only **10% of standard input price**
- Place static content (tools, system instructions, examples) at beginning of prompts
- Cache lifetime: 5 min default (refreshed on use) or 1 hour with extended TTL
- Stacks with Batch API's 50% discount for massive savings

Batch Processing

- Flat **50% discount** on all tokens
- Supports up to 300K output tokens per request on Claude 4.6
- Most jobs complete in under an hour

Other API Tricks

- **Structured Outputs** — Guaranteed schema-conformant JSON (replaces old prefill-{ trick)
- **Advisor Tool (beta)** — Pair fast executor model with smarter advisor for near-advisor quality at lower cost
- **Parallel tool calling** — Add <use_parallel_tool_calls> system prompt tag for independent calls
- **Token-efficient tool use** — Reduces token overhead for tool-heavy workflows

Subscription Savings

- Annual billing saves ~17% (Pro drops to $17/month)
- API billing only beats Pro below ~50 sessions/month
- Watch for credit offers during model launches and double-usage holiday events
- **Claude for Open Source** — Free Max 20x for 6 months (repos with 5K+ stars)

8. The Context Engineering System

The biggest gap between casual and power users is not prompt tricks — it is building persistent systems that load automatically before every interaction.

The 5 Essential Files

1. **Identity File**
   Your role, company, current priorities, decisions already made
2. **Voice Profile**
   Your beliefs, contrarian takes, sentence rhythm, things you find cringe
3. **Anti-AI Writing File**
   Banned words list + banned patterns (long intros, closing summaries, rule-of-three chains)
4. **Taste File**
   Examples of output you love AND output you hate
5. **Project Context File**
   Architecture decisions, current state, what has been tried and failed

High-Leverage Techniques

- **Socratic Prompting** — Ask Claude what questions it needs answered to do the task well. Surfaces assumptions you didn't know you were making.
- **HANDOFF.md** — Write a handoff note before switching contexts so the next session picks up seamlessly.
- **Pattern Detection** — Periodically ask: 'What patterns do you see in how I have corrected your outputs?' Save insights to memory.
- **After every correction**, tell Claude to update its CLAUDE.md so mistakes are not repeated.

BOTTOM LINE: The users getting the most from Claude are not writing better single prompts — they are building better systems around every interaction. Treat Claude as infrastructure, not a chat box.

Compiled April 2026 from Anthropic documentation, community testing, and verified sources.