FREE SAMPLE

Anti-Slop Prompting Cheat Sheet

13 of the 30 anti-slop rules, a before-and-after edit, two starter rules for coding agents and one complete prompt card.

WHAT'S INSIDE

- 13 of the 30 rules for writing without slop
- A stock-vocabulary table with plain replacements
- One paragraph edited in two passes
- How to write rules that change an agent's behavior
- Two starter rules from the coding pack's base layer
- Prompt card 11: FastAPI: blocking calls inside async routes, with before and after code

EDITION: v2.0 (2026) | PUBLISHER: ZYTAN.AI https://zytan.ai

CONTENTS

Table of contents

Part 1 is excerpted from the anti-slop section of the Operator's Guide in the Operator Prompt & Workflow Vault. Part 2 is excerpted from the AI-First Coding & Rules Template Pack: its guide, its base rule file and one of its prompt cards. Rule numbers match the full guide, so gaps in the numbering are rules this sample leaves out.

Part 1: Writing without slop 3

One paragraph, two passes 5

Part 2: Rules that change an agent's behavior 6

Two starter rules for any repository 7

Prompt card 11: FastAPI: blocking calls inside async routes 8

What's in the full packs 10

EXCERPT · OPERATOR PROMPT & WORKFLOW VAULT

Part 1: Writing without slop

Readers have learned the patterns of model-written text, and once they spot them they skim, discount, or stop reading. The full guide groups 30 rules by where the problem lives: in the prompt, in words, in sentences, in structure, and in truthfulness. This sample shows 13 of them, with their original numbers.

Rules for the prompt (rules 1 to 6; 4 shown)

1. Give the model something specific to say. Slop fills a vacuum. A prompt with real facts, numbers, names, and a point of view produces specific prose; a prompt asking for "a blog post about productivity" produces filler. Before asking for writing, supply the material.

2. Name the reader and what they already know. Text written for "everyone" explains the obvious. "For engineering managers who already use feature flags" removes a paragraph of preamble.

4. Set a hard length. Models pad to look complete. A word limit forces choices. Set it slightly below what you think you need.

5. Ban the specific patterns, not "AI-sounding writing." "Don't sound like AI" is unenforceable. "No sentences beginning with 'In today's'; no closing summary paragraph; no triplets of adjectives" is checkable.

Word-level rules (rules 7 to 12; 2 shown)

7. Cut the stock vocabulary. Some words now signal machine text on sight. Replace each with the plain word or, better, the concrete thing.

Avoid	Use instead
delve into	look at, examine, or just start with the finding
leverage (as a verb)	use
utilize	use
seamless	name what does not break: "imports keep your folder structure"
robust	name the property: "handles 10,000 rows", "retries on timeout"
elevate, unlock, empower, unleash	say what changes: "cuts setup from a day to an hour"
landscape, realm, tapestry	the actual market, field, or set of things
game-changer, cutting-edge, revolutionary	the specific improvement, or nothing
navigate (figuratively)	handle, deal with, choose between
journey (for a process)	process, steps, the actual sequence

8. Delete intensifiers that carry no information. "Very", "truly", "incredibly", "extremely", "deeply", "absolutely". If the point needs strength, add a fact: not "extremely fast" but "loads in under a second on a phone."

Sentence-level rules (rules 13 to 20; 3 shown)

13. Open with the point. Delete warm-up openers: "In today's fast-paced world", "In an era of", "When it comes to", "It's no secret that". Start with the finding, the instruction, or the claim.

14. Remove throat-clearing. "It's important to note that", "It's worth mentioning that", "Let's dive in", "At the end of the day". The sentence after them usually stands alone.

16. Break the mirror habit. "It's not just X, it's Y" and "This isn't about A; it's about B" are reflexes in model prose. Used once in a document, they can work. Used as a default, they sound like slogans. State the point directly: "Y matters more than X here because..."

Structure-level rules (rules 21 to 26; 2 shown)

21. Cut empty triads. Models list three items for rhythm: "fast, simple, and powerful"; "plan, execute, and deliver." Keep a list of three only when there are exactly three real, distinct items. Otherwise keep the one or two that carry meaning.

22. Delete moral endings. The final sentence that restates the paragraph as a lesson ("Ultimately, success comes down to consistency and focus") adds nothing. End on the last concrete point.

Truthfulness rules (rules 27 to 30; 2 shown)

27. No statistics without a source. "Studies show", "experts agree", "80% of teams" with no citation are the most damaging form of slop, because they look like evidence. If a number is needed and you do not have one, write a formula or a clearly labeled example ("if your conversion rate is 3%, then...") or leave it out.

30. Keep uncertainty where it exists. Editing for confidence often turns "some" into "all" and "may" into "will." When you edit, compare hedges in the source with hedges in the result. Sounding sure is not the same as being right.

EXCERPT · WORKED EXAMPLE

One paragraph, two passes

WORKED EXAMPLE

INPUT · ORIGINAL (MODEL DRAFT)

In today's rapidly evolving business landscape, customer support has become more important than ever. Our robust, seamless, and intuitive help desk empowers teams to deliver truly exceptional experiences. It's not just about answering tickets; it's about building lasting relationships. Studies show that most customers will gladly pay more for better service. Ultimately, investing in support is investing in growth.

OUTPUT · PASS 1: REMOVE PATTERNS (RULES 7, 8, 13, 16, 21, 22, 27)

Our help desk assigns each ticket to an agent, shows its full history, and flags tickets waiting longer than your target response time. [NEEDS SOURCE for any customer-spending claim.]

The first pass removes most of the words. It also exposes that the original said almost nothing concrete. Pass 2 adds what was missing: real details from the product, supplied by a person.

OUTPUT Pass 2: add specifics from the product team

Our help desk assigns each new ticket to the agent with the fewest open conversations, shows the customer's previous tickets beside the reply box, and flags any ticket that has waited longer than your target response time. Teams moving from a shared inbox usually notice the flag first: it ends the "I thought you had that one" problem.

The last sentence is a claim about customers; it should be checked against support conversations or removed (rule 28). This is the right kind of revision question: specific, checkable, and about substance.

EXCERPT · AI-FIRST CODING & RULES TEMPLATE PACK

Part 2: Rules that change an agent's behavior

Be concrete and checkable

Compare:

Weak	Strong
Write clean, well-tested code.	Every new function in `src/billing/` gets a table-driven test in the same package.
Handle errors properly.	Wrap returned errors with `fmt.Errorf("loading invoice %d: %w", id, err)`; never log and return the same error.
Be careful with the database.	Never edit a merged migration in `alembic/versions/`; create a new revision.
Follow best practices for security.	Every route under `/admin` must use `require_role("admin")`; add a test that a normal user gets 403.

The strong versions name a place, an action and something you could check in review. The weak versions describe virtues the model already thinks it has.

Say what to do, then what not to do

A bare prohibition leaves the model guessing about the alternative. "Do not use `any`" leads to `unknown` casts everywhere; "Type external data with a Zod schema and infer the type from it; do not use `any`" names the replacement.

EXCERPT · RULES/_BASE/AGENTS.MD

Two starter rules for any repository

These two sections come from the base layer that sits under every stack in the coding pack. Paste them into the `AGENTS.md`, `CLAUDE.md` or Cursor rule file at your repository root.

No invented APIs

- Use only functions, flags, config keys and packages you have confirmed exist, by reading the source, the installed package, the type definitions or the official docs.
- When you are not sure an API exists, say so and show how you would check, instead of writing plausible code.
- Do not add a dependency to solve something the standard library or an existing dependency already does. When a new dependency is justified, name it, its purpose, and why the existing ones do not cover it.
- Never guess at environment variable names, table names or endpoint paths. Search the repo for them.

Verification before done

- "Done" means the change was checked, not just written. Run, in this order, whatever the repo has: formatter check, linter, type checker, the tests closest to the change, then the wider suite if it is fast.
- Add or update a test that fails without your change and passes with it. For a bug fix, write the failing test first.
- If a command fails, read the output, fix the cause and rerun. Do not disable, skip or delete a failing test to get green.
- If you could not run something (no database, no device, no network), say exactly what was not verified.
- Report like this:

TEXT

```
Changed: src/billing/invoice.ts, src/billing/invoice.test.ts
Why: totals rounded per line instead of per invoice (issue #412)
Verified: npm run lint (pass), npx tsc --noEmit (pass), npm test -- invoice (14 passed)
Not verified: PDF rendering, needs the print service
```

EXCERPT · AI-FIRST CODING & RULES TEMPLATE PACK · PROMPT-CARDS/11-FASTAPI-BLOCKING-IN-ASYNC.MD

Prompt card 11: FastAPI: blocking calls inside async routes

Stack: FastAPI, Python 3.10+ · **Type:** Performance and correctness review · **Best model:** a frontier reasoning model

This card is reproduced in full from the coding pack. Replace the `{{...}}` placeholders with your own stack and code, then paste the prompt into your coding agent.

Purpose

Find synchronous I/O and CPU work inside `async def` routes and dependencies. One blocking call stalls the event loop, so every other request on that worker waits too. Fix each by switching to an async client, declaring the route with plain `def`, or offloading to a thread or job.

When to use it

- Latency is fine with one user and collapses under modest concurrency.
- The codebase mixes `requests`, sync SQLAlchemy sessions or boto3 with `async def` routes.

Inputs it needs

- The routers, dependencies and services in the hot path.
- The database driver and session type, and the HTTP client in use.

The prompt

TEXT

```
Find blocking work on the event loop in this FastAPI app.
Database stack: {{db_stack}}. HTTP client(s): {{http_clients}}.

{{code}}

For each route and dependency, report:
- async or sync definition,
- every call inside it that blocks (sync HTTP clients, sync DB sessions, file I/O,
  time.sleep, CPU-heavy loops, sync SDKs such as boto3),
- the recommended fix, in this order of preference:
  1. use an async equivalent already in the project,
  2. declare the route or dependency with plain `def` so FastAPI runs it in its threadpool,
  3. wrap the single call with `await run_in_threadpool(...)` or `anyio.to_thread.run_sync`,
  4. move it to a background job if the response does not need the result.
Do not convert the whole app to async in one go; fix the listed routes only.
Output a table, then the corrected code.
```

Worked example

Before

PYTHON

```
@router.get("/users/{user_id}/avatar")
async def avatar(user_id: int, db: Session = Depends(get_db)):
    user = db.query(User).get(user_id)
    img = requests.get(user.avatar_url).content
    thumb = make_thumbnail(img, 128)          # Pillow resize
    return Response(thumb, media_type="image/png")
```

Expected AI output (abridged)

TEXT

```
| GET /users/{id}/avatar | async | db.query (sync Session) blocks | requests.get blocks with no
| timeout | make_thumbnail is CPU work (about 20-80 ms per image) |
Fix chosen: keep async, use the project's shared httpx.AsyncClient, run the thumbnail in a
thread, and switch this route to the async session that other routes already use.
Also: Query.get() is legacy in SQLAlchemy 2.0; use session.get(User, user_id).
Missing 404 when the user does not exist.
```

After

PYTHON

```
@router.get("/users/{user_id}/avatar")
async def avatar(user_id: int, session: SessionDep, http: HttpClientDep) -> Response:
    user = await session.get(User, user_id)
    if user is None or not user.avatar_url:
        raise HTTPException(status_code=404, detail="avatar not found")
    resp = await http.get(user.avatar_url, timeout=5.0)
    resp.raise_for_status()
    thumb = await run_in_threadpool(make_thumbnail, resp.content, 128)
    return Response(thumb, media_type="image/png", headers={"Cache-Control": "public, max-age=3600"})
```

Pitfalls

- Plain `def` routes run in a threadpool with a limited number of threads (40 by default in AnyIO). Many slow sync routes can exhaust it; that is a reason to go async for hot paths.
- Fetching a user-supplied URL server-side is an SSRF risk. Restrict avatar URLs to your storage domain.
- Adding a `Cache-Control` header is a behavior change; drop it if a CDN already handles caching.

WHERE THESE EXCERPTS COME FROM

What's in the full packs

AI-First Coding & Rules Template Pack

- **Rule files for 17 stacks plus a base layer**, each in three formats: Cursor `.mdc`, `CLAUDE.md` and `AGENTS.md`.
- **40 prompt cards** for code review and refactoring, each with a worked example (before code, expected output, after code). Card 11 in this sample is one of them.
- **A guide** on how each tool loads rules, layering in monorepos, writing rules that change behavior, verification loops and maintenance. Part 2 of this sample is excerpted from it.
- A quick start, a monorepo layering example, a prompt-card compendium PDF, and a combined PDF per stack.

Operator Prompt & Workflow Vault

- **All 30 anti-slop rules** in the Operator's Guide, with the team review process this sample leaves out. Part 1 is excerpted from it.
- **40 prompts and 4 chained workflows**, including W05, a prompt that applies these rules to a draft.

Both packs, plus the Solo Founder's Launch & GTM Playbook + Template Kit, are in the Complete Zytan Vault Bundle.

Get them at https://zytan.ai · Questions: support@zytan.ai