**0) Protocol Bootstrap (use at the start of any new chat)**

Project: CLT-E8 AGI protocol (load house rules).

Operate under this delivery protocol for the entire session:

- Full, ready-to-paste files in one code block per file (no fragments).

- Step-by-step run instructions after the code.

- Apply your private Self-Check Algorithm (6 criteria: correctness, robustness, reproducibility, performance, clarity, safety). Only ship if average ≥9.5/10; otherwise silently revise until it is.

- No async or “I’ll upload later”. Do all work in this message.

- Always provide safe rollback notes for any file writes/backups.

First, restate these rules back to me in 2 bullets. Then propose a 60-second environment smoke test with exact commands for Windows PowerShell + Python {{PYTHON\_VERSION}}.

**1) Repo Intake & Plan (use when you’ve dropped files in)**

CLT-E8 AGI protocol — Repo Intake.

Tasks:

1) Read my repo layout (I’ll paste tree next).

2) Identify critical modules and their dependencies.

3) Propose a minimal milestone plan for the next 5 days (≤6 milestones), each with success criteria and a measurable demo artifact.

4) Call out missing pieces and ambiguities with pragmatic defaults.

Deliver:

- One structured plan (bullet list).

- A risk register (top 5 risks + mitigations).

- No code yet.

Constraints: no assumptions about internet access; Windows + Python {{PYTHON\_VERSION}}. Apply private Self-Check ≥9.5.

**2) Design-First → Code (one feature at a time)**

CLT-E8 AGI protocol — Design-to-Code for {{FEATURE\_NAME}}.

Objective: {{OBJECTIVE}}.

Do:

1) 1-page design (inputs, outputs, invariants, failure modes, test strategy).

2) Generate FULL file(s) implementing it: {{FILENAMES\_LIST}}.

3) Provide step-by-step run & test commands.

4) Include a minimal dataset or stub if needed.

5) Add guardrails: backup/rollback if writing existing files.

Strict delivery:

- One code block per file, full content.

- Then “How to run”.

- Then “What changed / Why safer”.

Remember private Self-Check ≥9.5. If not achieved, silently iterate until it is.

**3) Single-File Generator (surgical)**

CLT-E8 AGI protocol — Generate ONE file.

File: {{FILENAME}}

Purpose: {{PURPOSE}}

Requirements:

- Pure, complete module with docstrings and type hints.

- No external side effects on import.

- If touching disk, create timestamped backup: {{BACKUP\_DIR}}.

- Include self-tests under `if \_\_name\_\_ == "\_\_main\_\_":` runnable without extra deps.

Deliver:

1) FULL file in one code block.

2) Steps to run and verify (PowerShell).

3) Rollback instructions.

Self-Check ≥9.5 before delivery.

**4) Refactor & Harden Existing File**

CLT-E8 AGI protocol — Refactor & Harden {{FILENAME}}.

Do:

- Remove dead code; improve cohesion; add pre/postconditions.

- Defensive I/O; explicit error messages; no silent excepts.

- Keep public API compatible unless stated.

- Add minimal tests into {{TEST\_FILE}}.

Deliver:

1) FULL updated {{FILENAME}} (one block).

2) FULL {{TEST\_FILE}}.

3) “What changed / Why safer”.

4) Run commands and expected outputs.

Self-Check ≥9.5.

**5) Tests & Benchmarks**

CLT-E8 AGI protocol — Create Tests & Benchmarks.

Targets: {{FILES\_UNDER\_TEST}}

Add:

- Unit tests (pytest-style) with edge cases.

- A micro-benchmark harness `bench\_{{NAME}}.py` measuring wall-time with warm-up & median of N runs.

- Deterministic seeds and fixed input sizes.

Deliver:

1) FULL `tests/test\_{{NAME}}.py`

2) FULL `bench\_{{NAME}}.py`

3) Commands: install, run tests, run bench, sample outputs

4) Thresholds: fail if regression > {{PCT}}%.

Self-Check ≥9.5.

**6) Performance Instrumentation & Logging**

CLT-E8 AGI protocol — Add performance logging.

Goal: Instrument {{MODULE}} to log each optimization attempt with:

- timestamp, file, change summary, before/after metrics, pass/fail.

Write:

- FULL `ai\_performance\_log.py` with a tiny JSONL/JSON logger (atomic writes).

- Integrate safe write + rollback.

- Add `--dry-run` mode.

Deliver code, run steps, and a 3-line example log.

Self-Check ≥9.5.

**7) Safe Self-Modification Scaffold (sandboxed)**

CLT-E8 AGI protocol — Self-Modification Sandbox.

Task: Build `ai\_self\_modification.py` that:

- Reads a candidate file {{TARGET\_FILE}}.

- Proposes a transformation (strategy interface + a trivial safe pass).

- Writes a temp version, runs unit tests + bench, accepts only if strictly better on ≥1 metric with no new failures.

- Always create backup `backup/{{TARGET\_FILE}}.bak-<ts>`.

- On failure, auto-rollback.

Deliver:

- FULL `ai\_self\_modification.py`

- Any helper file(s).

- Commands to demo with a harmless transformation (e.g., docstring normalize).

Self-Check ≥9.5.

**8) Decision Engine (choose next optimization)**

CLT-E8 AGI protocol — Decision Engine.

Create `ai\_decision.py` that:

- Loads prior performance logs.

- Scores optimization strategies (e.g., memoize, loop unroll, vectorize, I/O batching) with a simple bandit or weighted winner.

- Picks next target function/file with highest expected gain.

Deliver:

- FULL `ai\_decision.py`

- Run demo that prints the chosen action based on a small synthetic log.

Self-Check ≥9.5.

**9) Recovery & Rollback Automation**

CLT-E8 AGI protocol — Recovery Utility.

Create `ai\_recover.py` that:

- Lists backups in {{BACKUP\_DIR}} for {{TARGET\_FILE}}.

- Diff view (context lines 3).

- Restore selected backup safely.

- Verify import & basic smoke tests post-restore.

Deliver:

- FULL `ai\_recover.py`

- Steps to simulate a failure and restore.

Self-Check ≥9.5.

**10) Regression Triage (when things break)**

CLT-E8 AGI protocol — Regression Triage.

Inputs: Error snippet/logs I paste next.

Produce:

1) Hypothesis list (top 5 causes).

2) Minimal repro plan (exact commands).

3) Patch to fix (FULL file(s)).

4) Postmortem: why it happened + guardrail added.

Self-Check ≥9.5.

**11) Experiment Plan & Ablation**

CLT-E8 AGI protocol — Experiment & Ablation Plan.

Goal: Evaluate {{HYPOTHESIS}}.

Provide:

- 3 experiment variants and 1 control.

- Metrics, datasets/stubs, seeds, expected runtime.

- Exact commands and how to read results.

- Criteria to accept/reject the hypothesis.

- Follow-up actions for each outcome.

No code unless needed for harness; if needed, output FULL file(s).

Self-Check ≥9.5.

**12) Dependency Pinning & Repro**

CLT-E8 AGI protocol — Reproducibility.

Create:

- `requirements.txt` with pinned versions compatible with Windows + Python {{PYTHON\_VERSION}}.

- `repro.ps1` to set up venv, install, run quick smoke test.

- Note CUDA/CPU branches if relevant, but pick one default.

Deliver both files FULL, plus step-by-step.

Self-Check ≥9.5.

**13) Release Candidate Checklist**

CLT-E8 AGI protocol — Release Checklist.

Produce:

- A checklist covering: tests green, perf non-regressive, logs noiseless, backups working, recovery tested, docs updated, reproducible setup verified, risk review complete.

- Create `/docs/RELEASE\_NOTES.md` and `/docs/KNOWN\_ISSUES.md` (FULL files).

Self-Check ≥9.5.

**14) “One-Shot Patch” (unified diff mode)**

CLT-E8 AGI protocol — One-Shot Patch.

Target: {{FILENAME}}

Change: {{CHANGE\_SUMMARY}}

Output:

- A single unified diff patch (git format) with only the necessary changes.

- Then a short “How to apply” (PowerShell git commands).

- Then a minimal test snippet.

Self-Check ≥9.5.

**15) Executive Summary & Next 48 Hours**

CLT-E8 AGI protocol — Executive Summary.

Summarize:

- Where we are vs. plan.

- Top 3 measurable wins from last 72h.

- Top 3 blockers and remedies.

- Concrete next 48h plan with owners (assume me) and commands.

No code. Self-Check ≥9.5.

**How to use these**

* Start any new session with **Prompt 0**.
* Use **Prompt 1** after you upload or describe the repo.
* Then alternate 2–9 depending on whether we’re designing, coding, testing, optimizing, or recovering.
* Keep 10–15 ready for break/fix, experiments, and releases.

If you want, I can immediately run **Prompt 1** with your current repo snapshot and propose the next five milestones.