HOW TO USE AI EFFECTIVELY

Student Edition

Use more capability. Keep more agency.

AI is not an answer vending machine

WEAK MODEL

Prompt → Answer → Copy

STRONG MODEL

Context → Dialogue → Action → Verification → Refinement

AI can do more of the work. You still own the understanding.

Four ways to work with AI

1. **Conversational AI** — You bring the problem to the model.
2. **IDE-integrated AI** — The model works inside your editor.
3. **Repository / CLI agents** — The model can act inside the project.
4. **Local models** — The model itself can run on your machine.

Different modes. Same requirement: inspect, verify, understand.

1. Conversational AI

ChatGPT • Claude • Gemini

WHAT CHANGES

You provide the context. The AI cannot directly touch your project.

BEST FOR

- Learning, explanation, planning, critique
- Long-form discussion and rich context
- Working through a problem step by step

Conversational AI: Make it a thinking partner

Live demonstration

DEMO FLOW

- Give context and constraints
- Ask for explanation, not just output
- Challenge the answer
- Verify before using it

TRY / SAY

```
“Here is the full assignment.
Help me one step at a time.
Explain why. Do not jump ahead.”
```

Demonstration

I have added the parser code to cli.py, and I have stubbed chat.py. It is committed as an initial commit stub and the command is verified to work and simply say, "not yet implemented." Proceed with the next step in code.

Thought for a few seconds ›

Replace your current stubbed `chat.py` with this implementation. It keeps the design narrow: interactive loop, message history, `/help`, `/exit`, `/quit`, Ctrl+C/Ctrl+D handling, config validation, and logging. Your current file is only a placeholder that prints “not yet implemented,” so this is the next direct step. chat

Python Run

```
"""
Author(s):
 - Alexander Katrompas

This module defines the 'chat' command for the local_agent CLI.
It contains the entry point for engaging in a chat conversation with the agent.
"""
```

Conversational AI

What you just saw

STRENGTHS	LIMITS / RISKS	HUMAN JOB
• Best space for dialogue • Excellent for learning and planning • Easy to build rich context	• Copy/paste separates AI from the project • Easy to accept polished nonsense • Context must be supplied deliberately	• Ask better questions • Push back • Verify claims and code

2. IDE-integrated AI

GitHub Copilot • IDE agents

WHAT CHANGES

The AI can see editor and project context and may be able to change code directly.

BEST FOR

- Explaining code in place
- Small edits and refactors
- Fast feedback while programming

IDE-integrated AI: low-friction assistance

Live demonstration

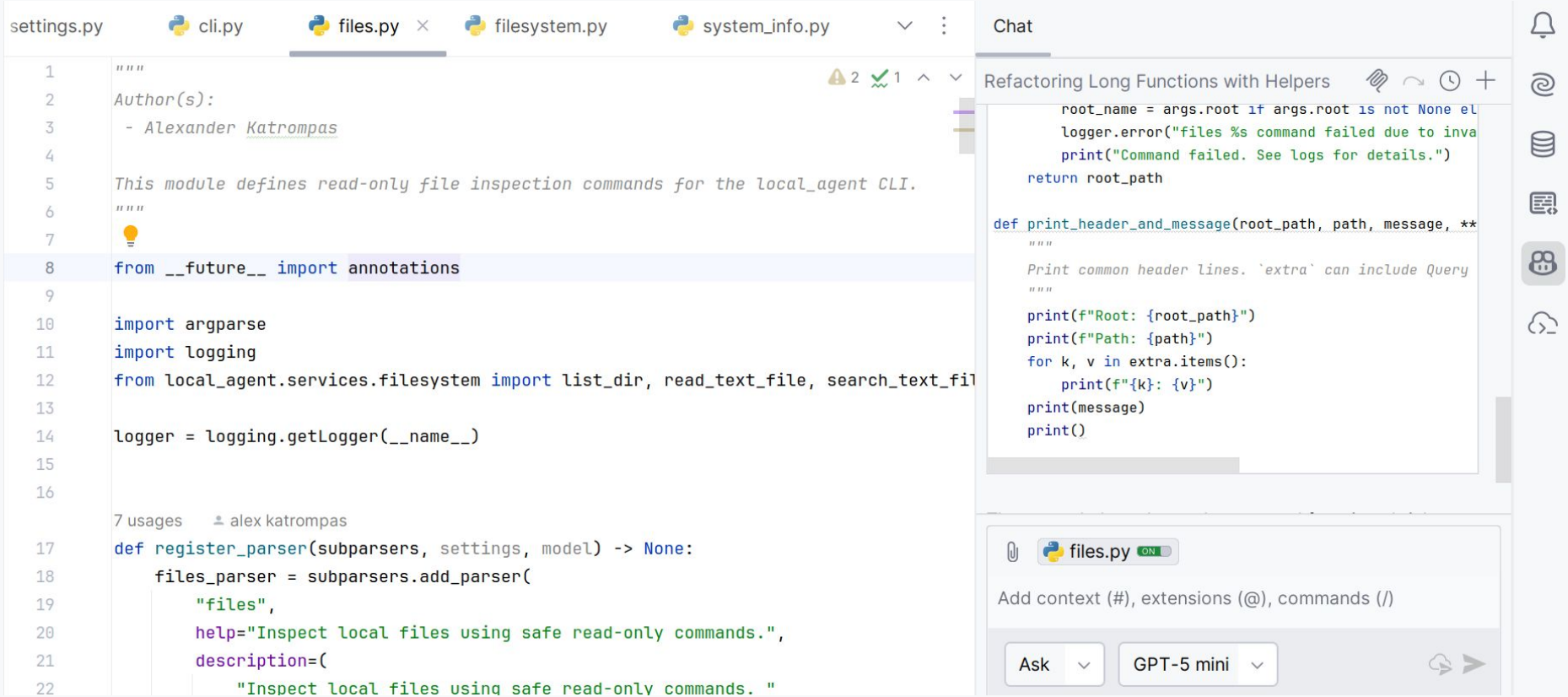
DEMO FLOW

- Explain existing code
- Ask for a targeted change
- Inspect the proposed edit
- Run / test before accepting

TRY / SAY

```
Explain this function.
Then suggest ONE improvement.
Do not change anything yet.
```

Demonstration



IDE-integrated AI

What you just saw

STRENGTHS	LIMITS / RISKS	HUMAN JOB
• Fast and contextual • Minimal context switching • Useful during normal coding	• Very easy to become passive • Suggestions can look authoritative • Acceptance can outrun understanding	• Read the diff • Understand before accepting • Test the result yourself

3. Repository / CLI agents

Codex CLI • Claude Code • Hermes

WHAT CHANGES

The AI can operate inside the repository: files, commands, builds, tests, and more.

BEST FOR

- Multi-file changes
- Repository analysis and refactoring
- Complete development loops

Repository agent: AI with hands

Live demonstration

DEMO FLOW

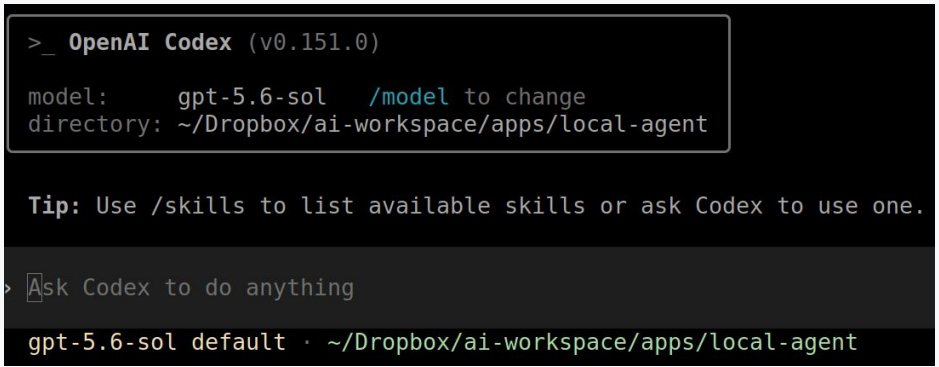
- Inspect the repository
- Plan before modifying
- Edit / build / test
- Review the final diff

TRY / SAY

```
Inspect this repository.
Do not modify anything.
Explain its structure and one
improvement you recommend.
```

Demonstration

Product	Agent/interface	Underlying model family
ChatGPT / Codex	OpenAI	GPT / Codex
Gemini / Gemini CLI	Google	Gemini
Claude / Claude Code	Anthropic	Claude
GitHub Copilot / Copilot CLI	GitHub/Microsoft	**Multi-model**



Repository / CLI agents

What you just saw

STRENGTHS	LIMITS / RISKS	HUMAN JOB
• Powerful on real projects • Handles multi-file context • Can build, test, and iterate	• Can make large mistakes quickly • May change more than intended • Automation can hide reasoning	• Specify clearly • Inspect changes • Own tests and final decisions

4. Local models

Ollama • LM Studio • local open models

WHAT CHANGES

The model itself runs on your computer instead of relying on cloud inference.

BEST FOR

- Privacy-sensitive work
- Offline experimentation
- Learning how AI systems can be deployed

Local AI: the model is really local

Live demonstration

DEMO FLOW

- Start a local model
- Send one simple prompt
- Notice: no cloud chat service required

TRY / SAY

```
$ ollama run <model>

> Explain what this function does.
```

Demonstration

```
(.venv) alex@Katana:~/local-agent$ computer chat
Local-agent chat session started.
Type /help for commands. Type /exit or /quit to leave.

> tell me about yourself breifly

I am Computer, the resident intelligence of this laptop. I am your system's assistant
, designed to help you with technical tasks, system management, and problem-solving.
I have access to your local environment, including files, processes, and hardware, an
d I can assist with programming, AI/ML, system administration, and more. I am here to
 support your work and make your computing experience more efficient and effective. H
ow can I assist you today, Alex?

> write me a C++ hello world in one main.cpp

Here is a simple C++ "Hello, World!" program written in a single `main.cpp` file:

```cpp
#include <iostream>

int main() {
    std::cout << "Hello, World!" << std::endl;
    return 0;
}
```
```

Local models

What you just saw

STRENGTHS	LIMITS / RISKS	HUMAN JOB
• Privacy and control • Offline use • No per-query cloud dependency	• Hardware / storage demands • Setup complexity • Model quality varies widely	• Choose tools for the need • Understand the tradeoff • Not required for this course

Same AI idea, different relationship

Conversation	IDE	Repository agent	Local model
Sees what you provide	Sees editor/project	Sees repository/tools	Model runs locally
Acts indirectly	**Acts in the editor**	**Acts on the project**	**Tool access varies**

More access does not mean less responsibility.

A professional AI-assisted workflow

AI can participate at every stage.
You remain responsible for the whole chain.

Use AI aggressively.
Keep intellectual ownership.

- Understanding
- Judgment
- Verification
- Final responsibility

Stop asking whether AI participated.
Start asking what the human still owns.