

A survey of

AI Coding Abilities

from beginner to advanced
covering the most important
tools, background, and techniques

Kyle Jensen*

YALE SOM FACULTY SEMINAR
WEDNESDAY OCTOBER 19th 2025

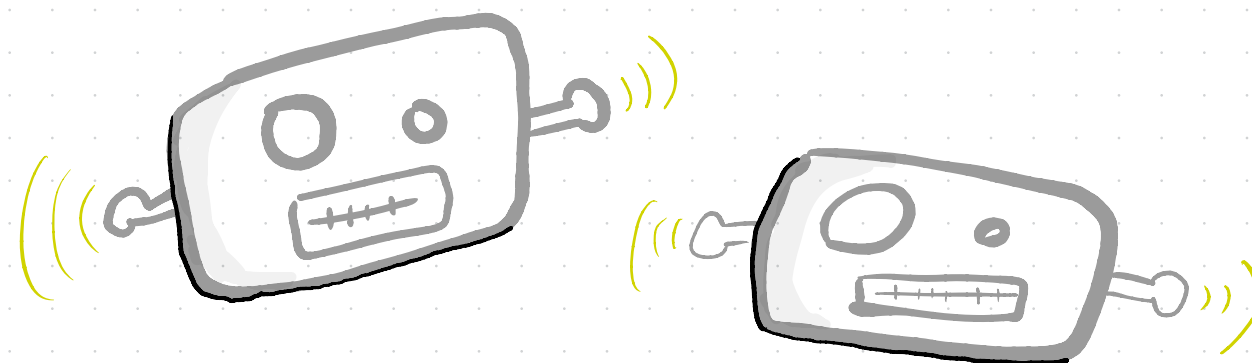


Nobody had research to share
today, so I volunteered this.
This is not research.

we. are. here.
to. help.

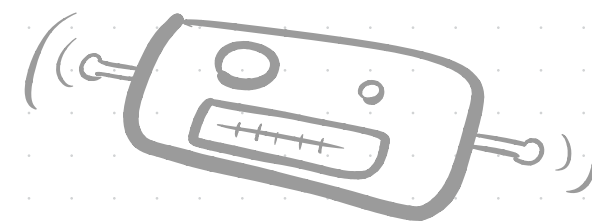
promise not
to take your
job. ha. ha. ha...

I mean
010010011....



THE SIX-FOLD

Path to Enlightenment



Level 0

Copy / paste from the ChatGPT etc.

Level 1

In-line chat / completion in VSCode/etc

Level 2

Agent mode in VSCode/Cursor/replit

Level 3

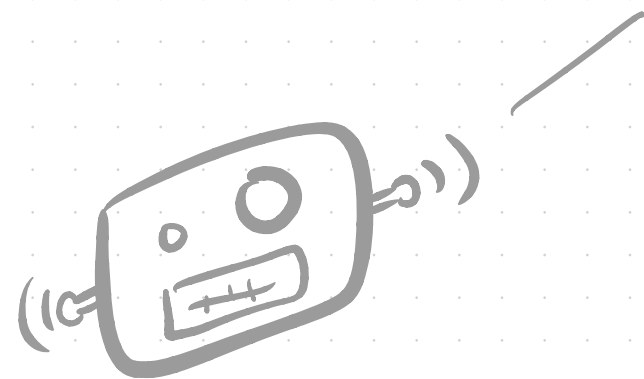
Agents in Claude code, OpenAI codex, etc.

Level 4

Agents with MCP tools

Level 5

Full auto containerized agent teams



One rule to rule
them all...

$$\text{Performance} \sim \frac{\text{correctness}^2 \cdot \text{completeness}}{\text{size}}$$

Top tips for copy/paste level



- use backticks (`) to delineate code
- the usual prompting best practices all apply to the case of writing code

- ROLE PLAY
- CHAIN OF THOUGHT ~ Writing is thinking
- PLANNING ~ planning is prompting
- DO NOT ARGUE WITH THE MACHINE

- Use web search tools to prime the LLM and prevent hallucination

web search is an example of "tool use", which is a new-ish feature

- NEVER RELY ON ITS BUILT-IN KNOWLEDGE OF CODE LIBRARIES


```
<|start|>system<|message|>You are ChatGPT, a large language model trained by OpenAI.  
Knowledge cutoff: 2024-06  
Current date: 2025-06-28  
Reasoning: high  
# Valid channels: analysis, commentary, final.  
Channel must be included for every message.  
Calls to these tools must go to the commentary channel: 'functions'.<|end|>
```

system prompt
(written by open AI)

ANATOMY OF A MODERN LLM CHAT

```
<|start|>developer<|message|># Instructions  
Be concise and technical. Provide code examples when relevant.
```

developer
message
(tool definitions)

```
# Tools  
## functions  
namespace functions {  
  type web_search = (_: {  
    query: string,  
    num_results?: number,  
  }) => any;  
} // namespace functions<|end|>
```

```
<|start|>user<|message|>What's new in the latest version of polars?<|end|>
```

user
prompt

```
<|start|>assistant<|channel|>analysis<|message|>User asking about latest polars version.  
My knowledge cutoff is 2024-06, need to search for recent updates.<|end|>
```

thinking

```
<|start|>assistant<|channel|>commentary<|message|><|call|>functions.web_search  
{  
  "query": "polars python latest version new features", "  
  "num_results": 5  
}<|end|>
```

tool call

```
<|start|>tool<|message|>[{"title": "Polars 1.8.0 Release Notes", "snippet": "New  
streaming engine, improved join performance, added pivot_wider...", "url": "..."}]<|end|>
```

tool
results

```
<|start|>assistant<|channel|>final<|message|>Polars 1.8.0 (latest as of Nov 2025)  
includes:
```

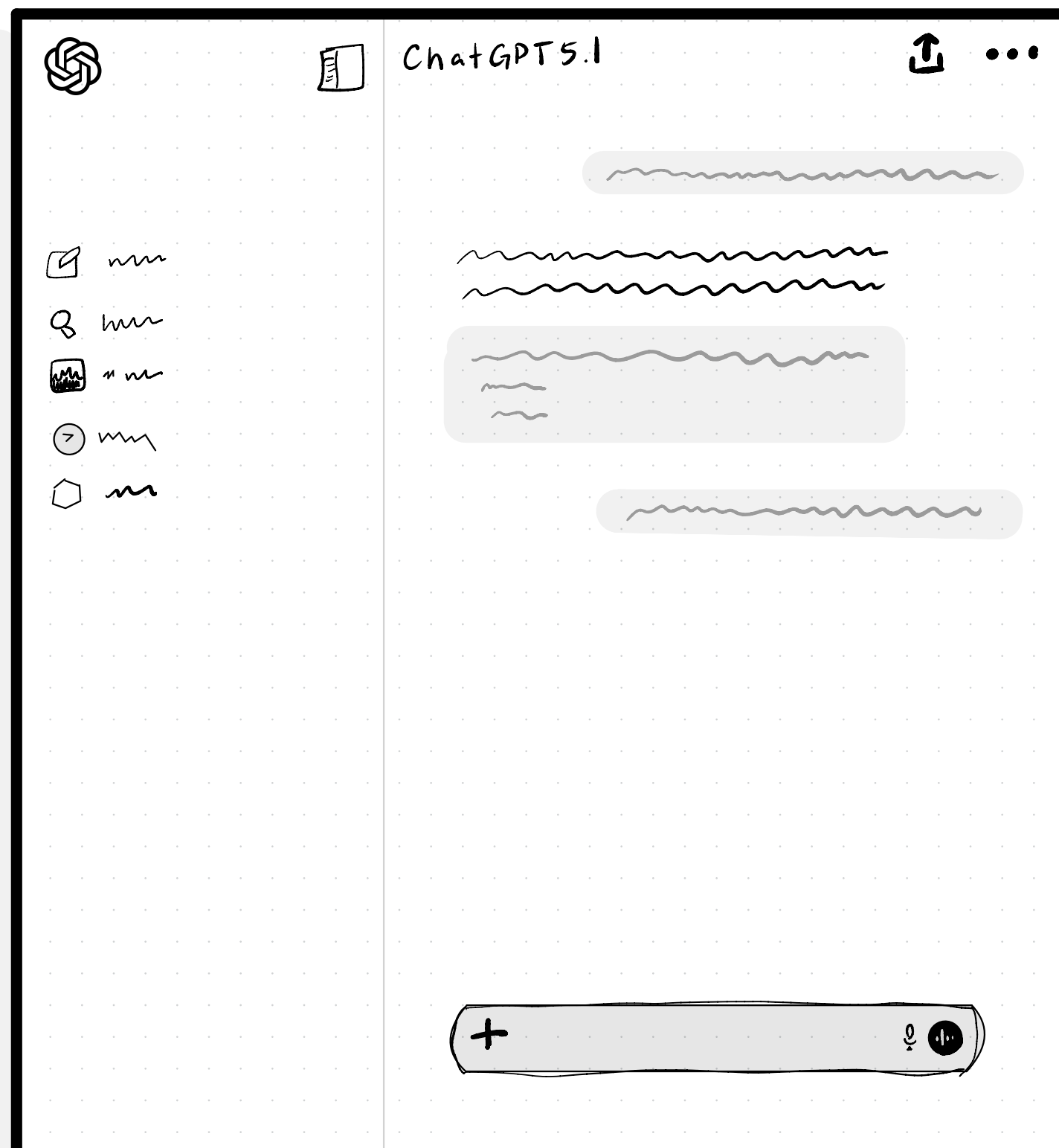
- **Streaming engine overhaul**: Better memory efficiency for large datasets
- **Join performance**: 2-3x faster for certain join types
- **New API**: `pivot_wider()` for reshaping data
- **LazyFrame improvements**: Enhanced query optimization

final response

```
Check the release notes for migration details if upgrading from older versions.<|return|>
```

You
typically
see only
these
two

You don't want to
write serious code
in the web interface



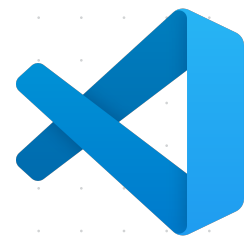
- **Lack of context**
The model can't see other files
- **Limited tool use**
So many unix tools on your computer
- **Generic model**
Better to use code-specific models
- **Long feedback cycles**
copy paste copy paste copy paste

Integrated development environments



Cursor

The most AI-forward IDE.
Likely the best choice for most.
Paid tiers at \$20, \$60, \$200/mo.
One year free for students.



VSCode + Copilot

*let's start
with this
one!*

Decent choice. Free for verified teachers and students



Zed

Decent for basic stuff.
Only use if you need
great collaboration like
google docs but on code



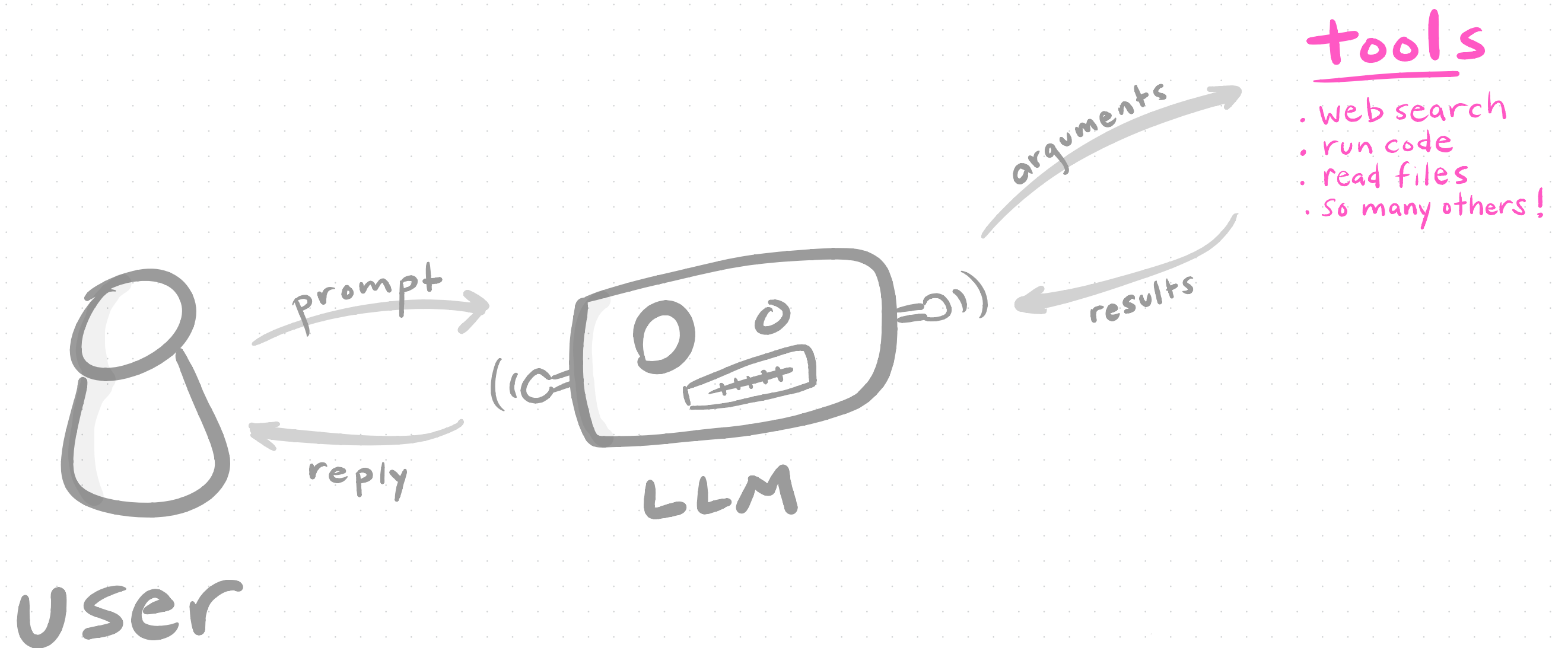
VS Code

+ Copilot

Let's look at
these features

1. Code completion
2. In-line AI chat
3. Sidebar chat/plan
4. Agent mode
└─ what is an "agent"

an agent is an LLM
in a loop with tools



Let's look at
these features



Cursor

1. Agent-forward workflow
The agent is the main way you interact with your codebase
2. Automatic context indexing
It is pretty smart about reading your code
3. MCP Support
The model context protocol gives your agents new abilities

Terminal-based AI tools



Claude Code

My favorite. Most powerful agent features. Skills, plugins.



OpenAI Codex CLI

Good if you are already paying for a high level of ChatGPT. Many people prefer ChatGPT5-codex to sonnet 4.5.



Gemini CLI

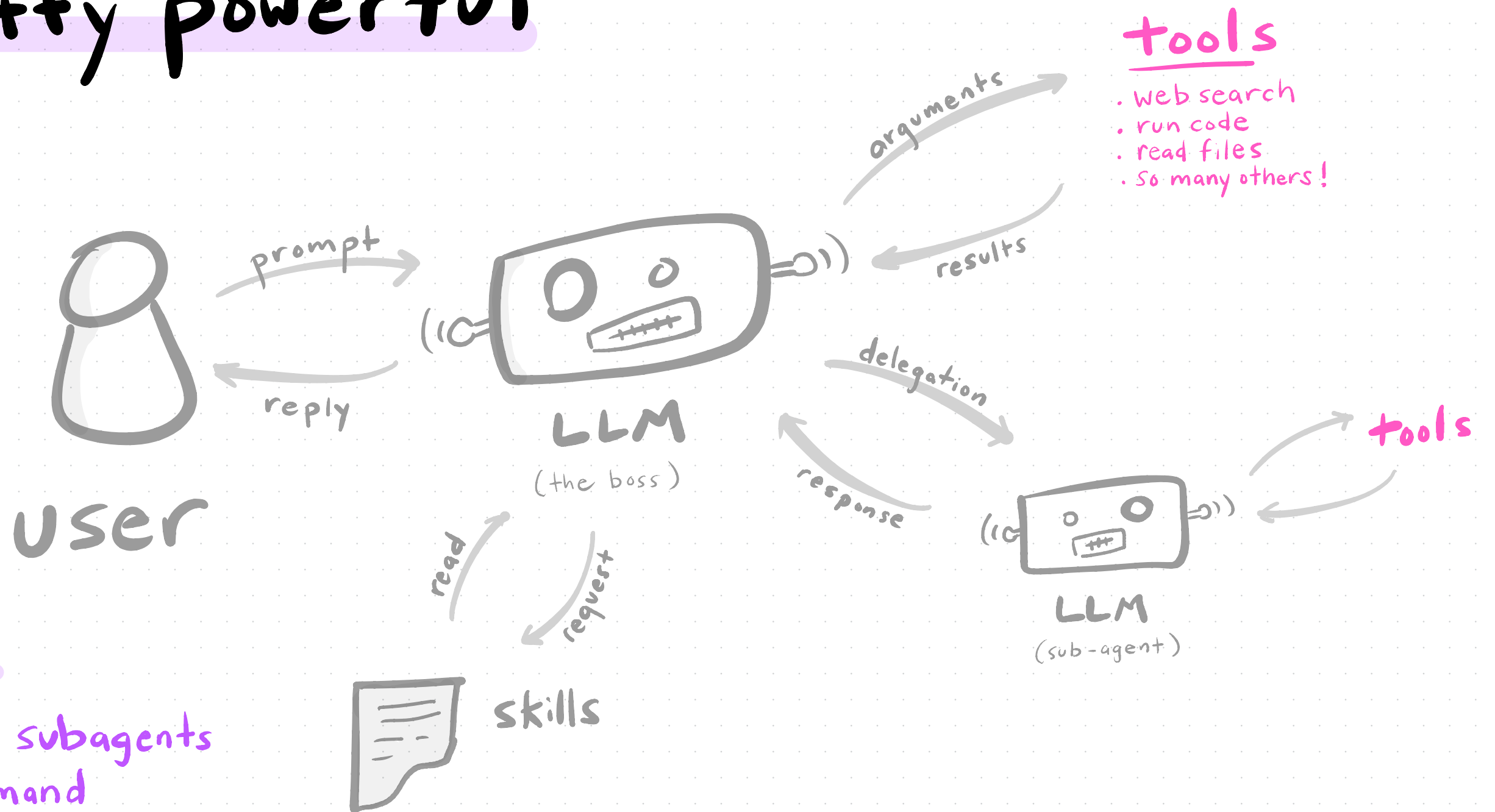
Decent free tier. Has up to a 1 million token context window! (I think...)



Copilot CLI

Really tight integration with Github. Nice free level for educators.

Claude's agents + skills are pretty powerful



SWEET FEATURES

- Delegation to subagents
- Upskilling on demand
- Background task management
- Parallel task execution

You can give Claude new abilities using MCP tools

(model
context
protocol)

and others

these are
my top few...
you can find
hundreds

- Sequential Thinking

Makes the model spend a ton of tokens thinking in a very structured fashion

- Exa, Tavily, Kagi

Web search and retrieval tools that are like "Deep Research" on steroids.

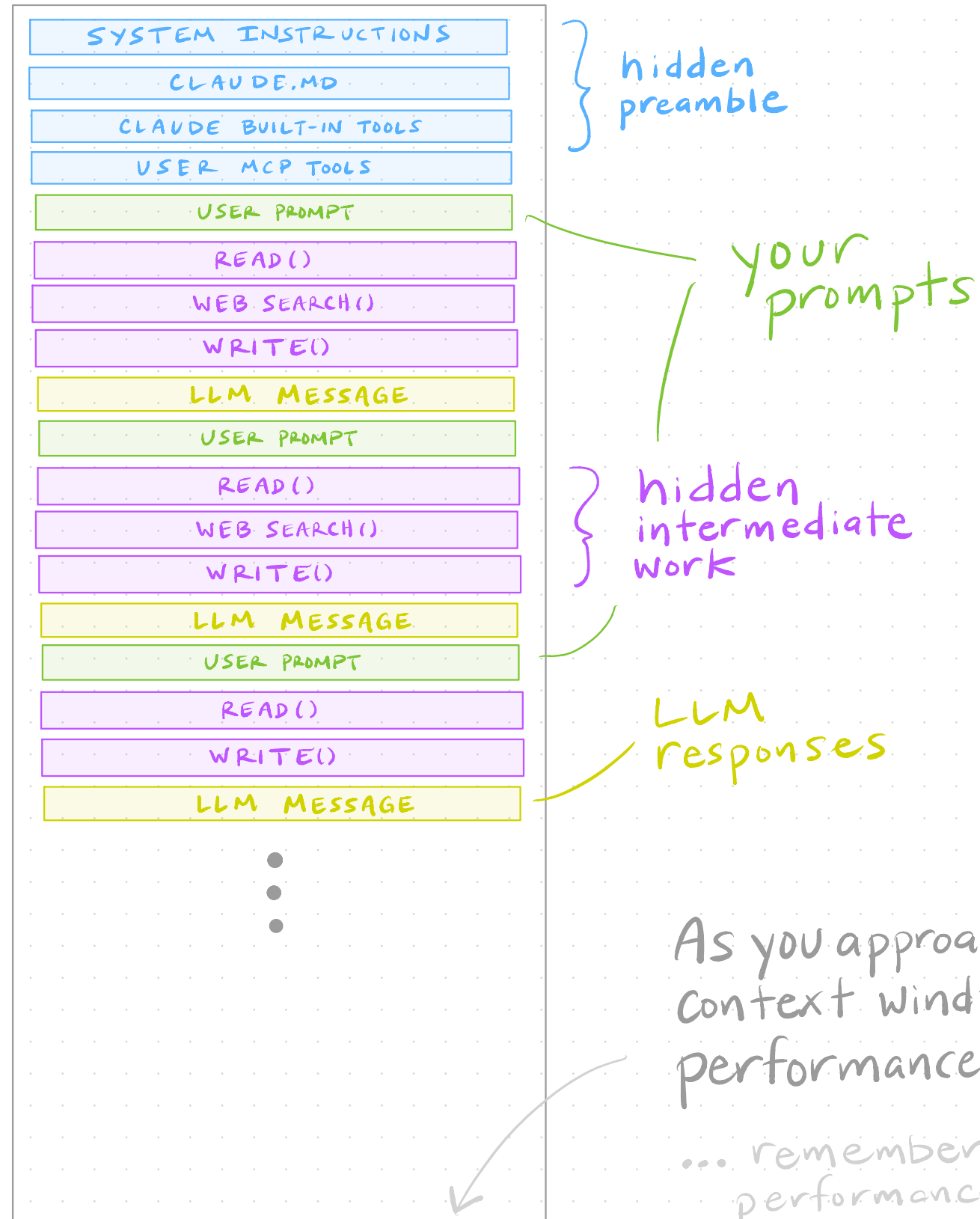
- Playwright

Browser automation, great for web scraping

- Codex!

I like to have Claude chat with ChatGPT 😊

CODING SESSION WITH AI



next few slides based heavily on HumanLayer

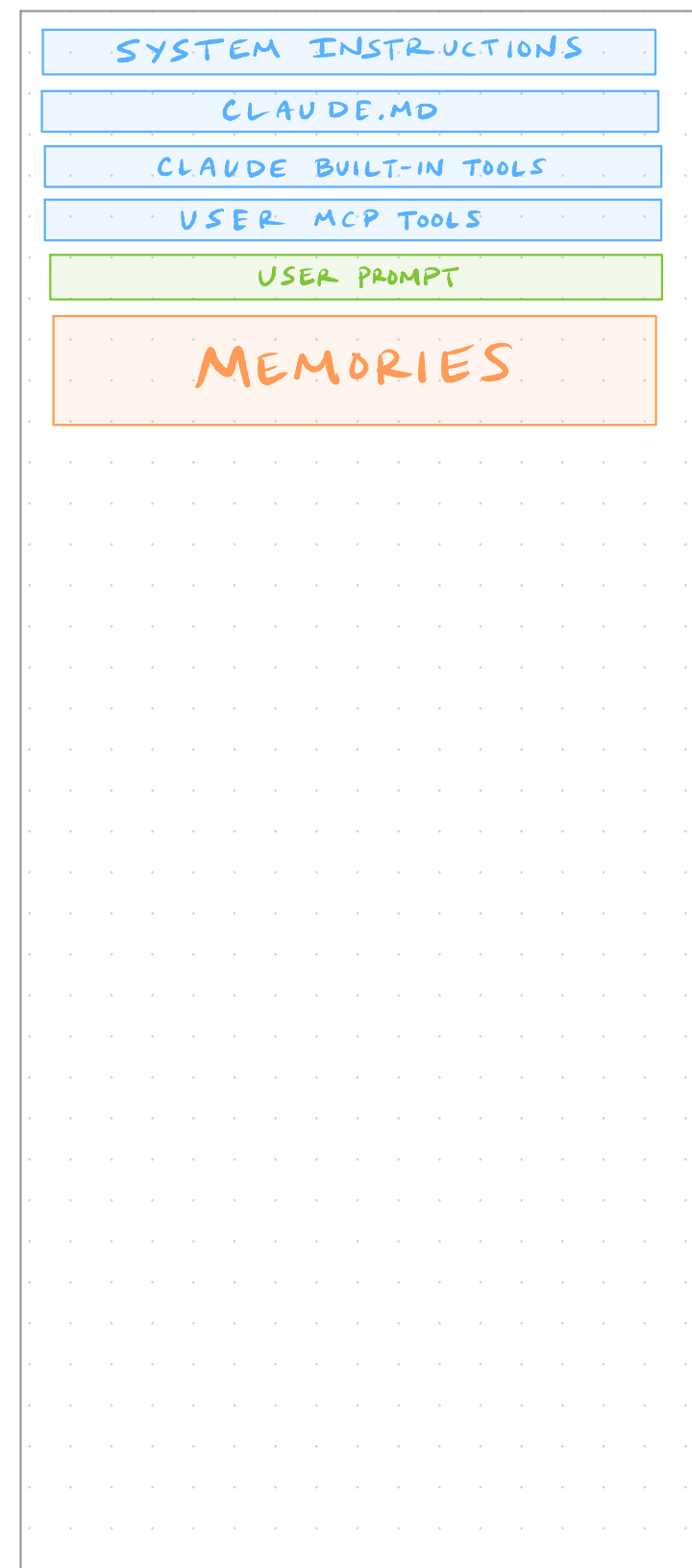
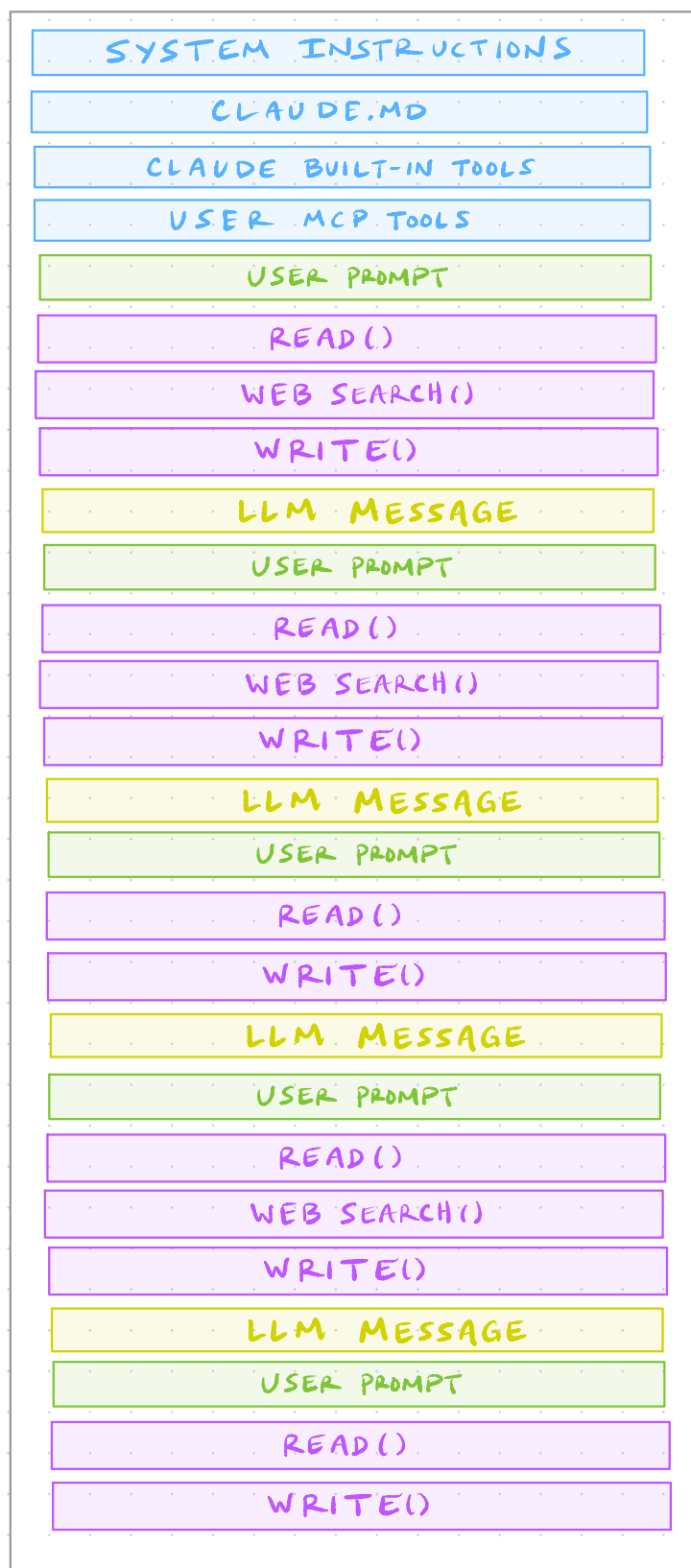
Context Engineering is the new Prompt Engineering

As you approach the context window, performance worsens

... remember performance $\sim \frac{\text{correctness}^2 \cdot \text{completeness}}{\text{size}}$

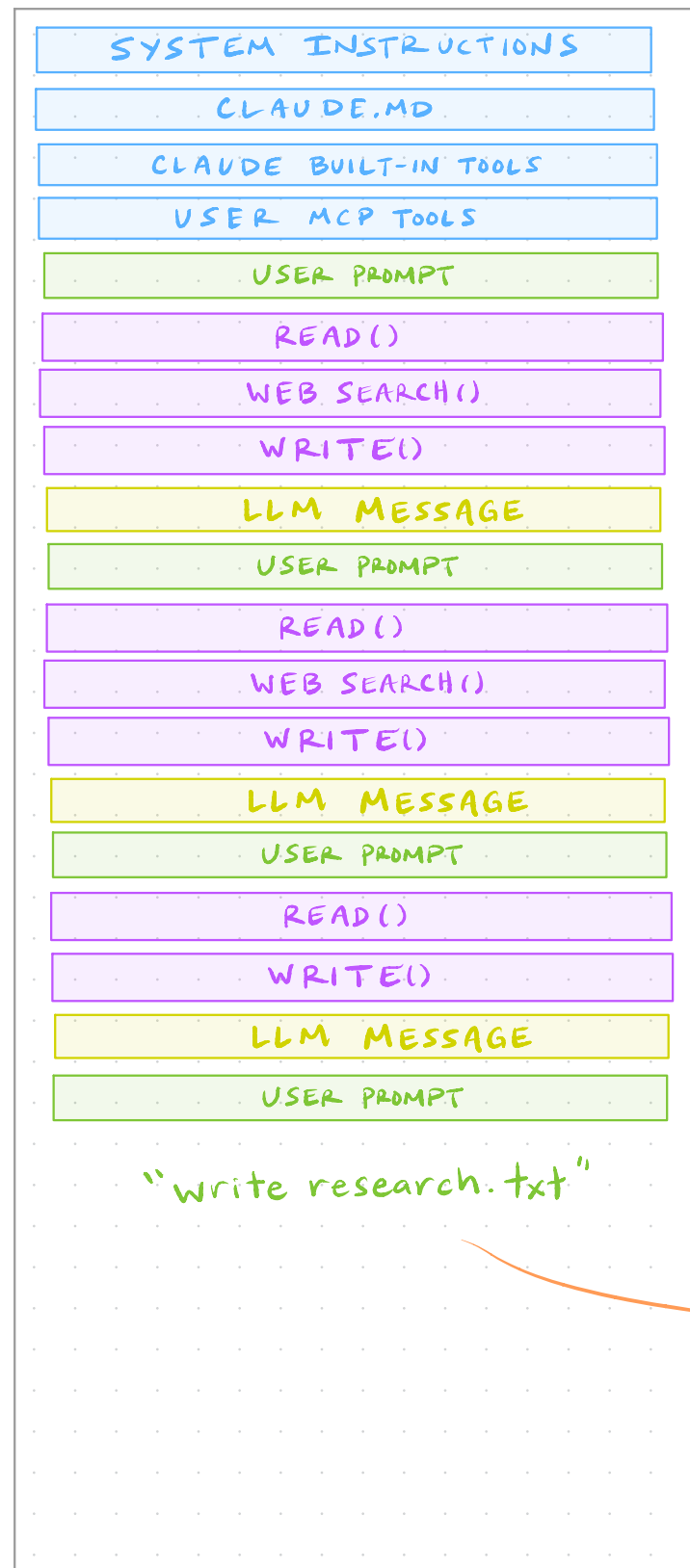


If you hit the content window the LLM will auto-compact

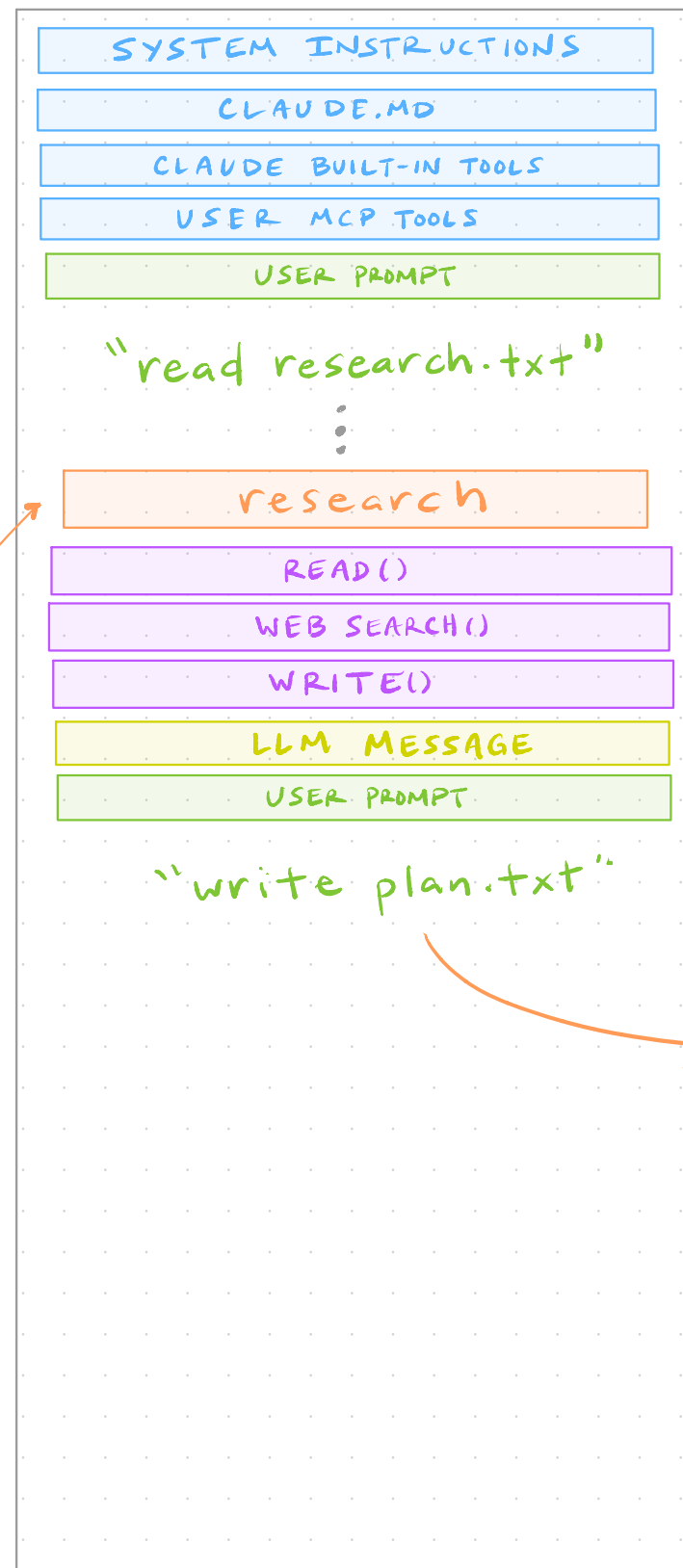


...this is
rarely
ideal...

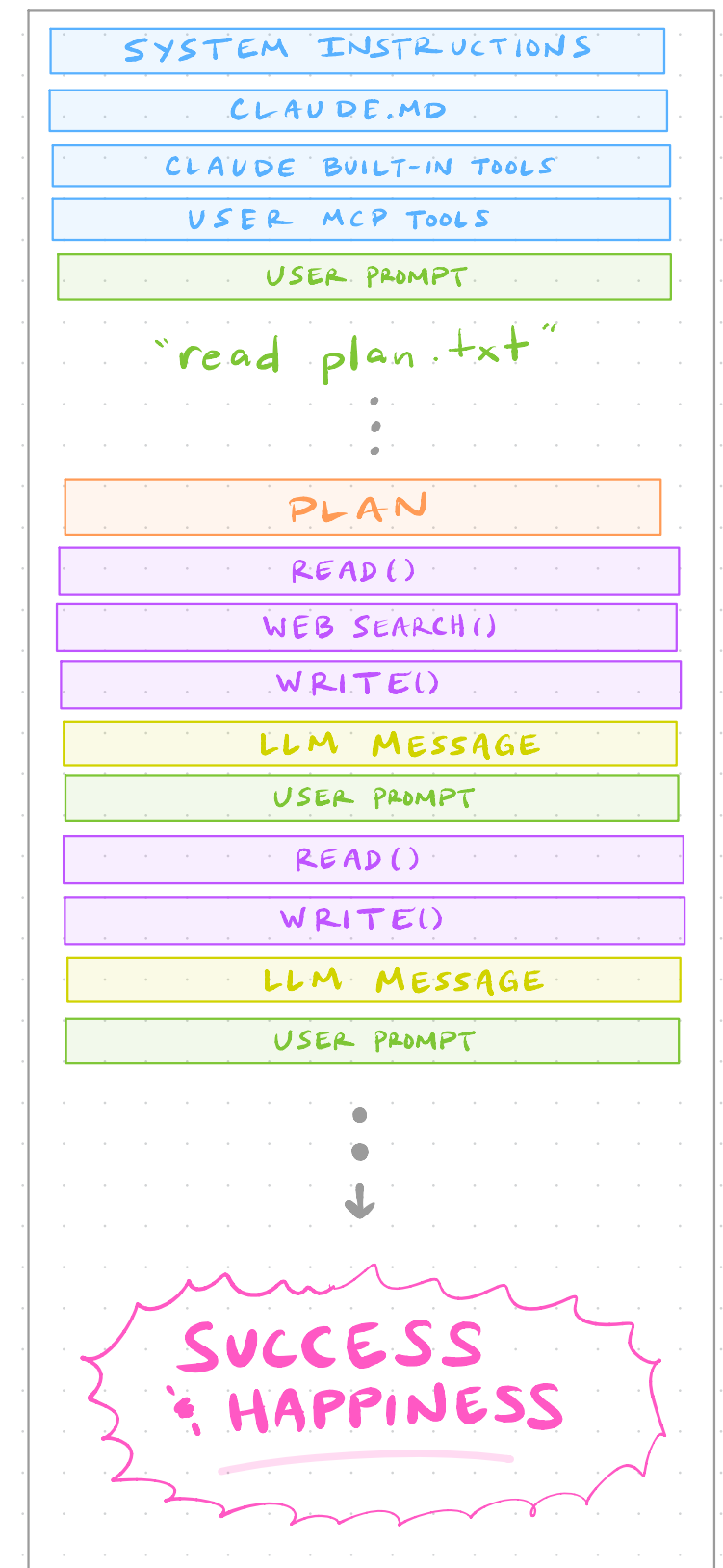
Frequent intentional compaction is better



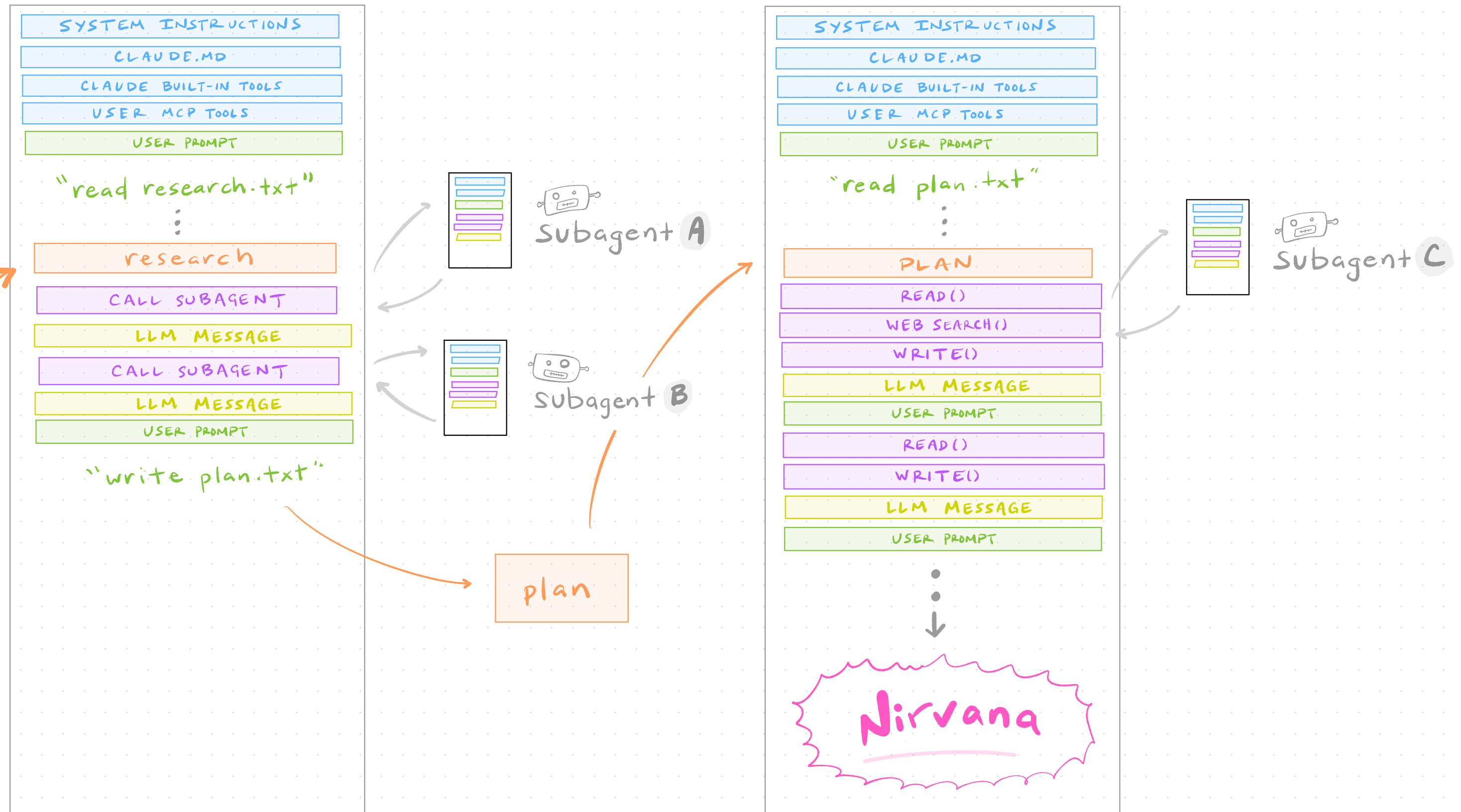
research



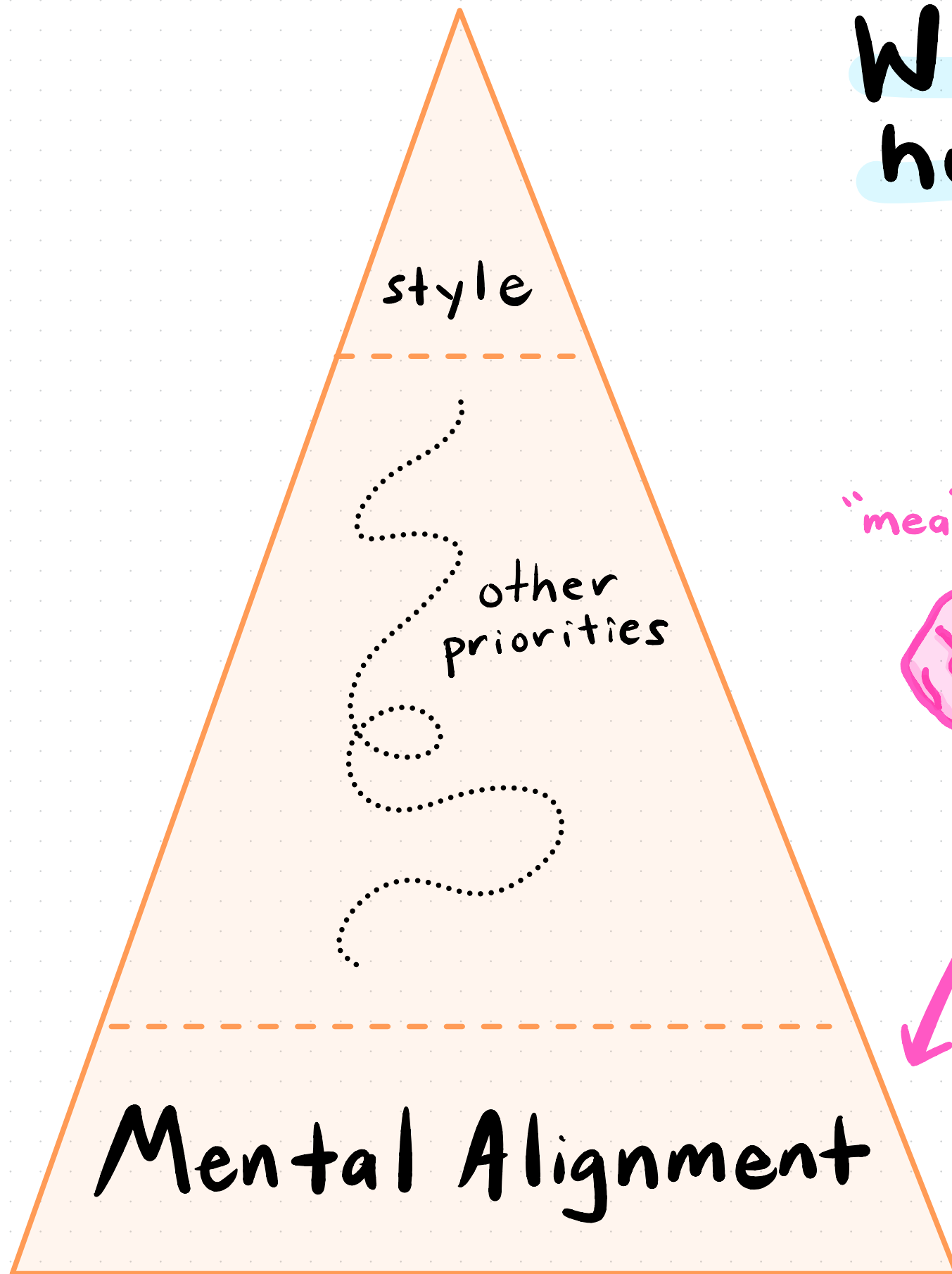
plan



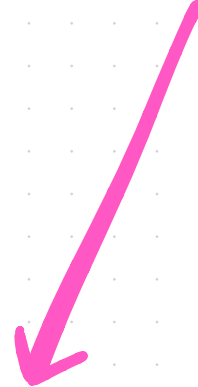
Delegation to sub-agents is best



What is the role of humans now in coding?



your
"meat computer"



The best marginal use of your time is creating

- mental alignment with the machine
- the conditions for it to excel - tools, guardrails, guidance, etc.

Yolo mode is great but need caution

SubscribeLog inMenu

Weekly editionThe world in briefWar in the Middle EastWar in UkraineUnited StatesWorld Ahead 2026The

Leaders | Crossing the bridge

How to stop AI's "lethal trifecta"

Coders need to start thinking like civil engineers

Sep 25th 2025 | 3 min read

LARGE LANGUAGE MODELS (LLMs), a trendy way of building artificial intelligence, have an inherent security problem: they cannot separate code from data. As a result, they are at risk of a type of attack called a prompt injection, in which they are tricked into following commands they should not. Sometimes the result is merely embarrassing, as when a customer-help agent is persuaded to talk like a pirate. On other occasions, it is far more damaging.

The worst effects of this flaw are reserved for those who create what is known as the "lethal trifecta". If a company, eager to offer a powerful AI

- Access to private data
- Ability to communicate externally
- Exposure to untrusted content

(Please come see me if you want to yolo safely & successfully)

~~Fun~~

I will send these
slides & some helpful
links to all who RSVPd
for the seminar.