Job Interview in the AI-Era: Coding, Systems, Agents

Zhaoying Pan

05/22/2026

Lecture 3

Foundation: AI Agents

Outline

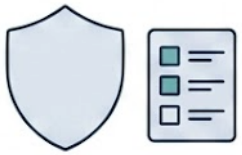
- AI Agent Overview

- Work with AI Agents

- Prompting in Code Context

- Preparing Interviews with AI Agents

- Asking for Clarification Effectively

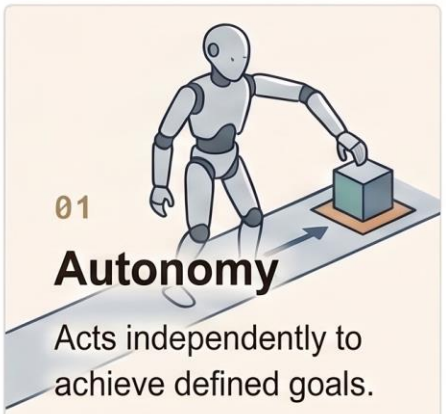


- Risks and Best Practices

PURDUE UNIVERSITY

1. AI Agent Overview

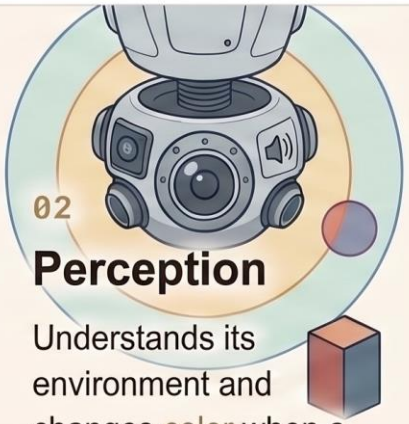
AI agents are AI-powered systems that reason, plan, remember, and act autonomously to complete tasks for users.



01

Autonomy

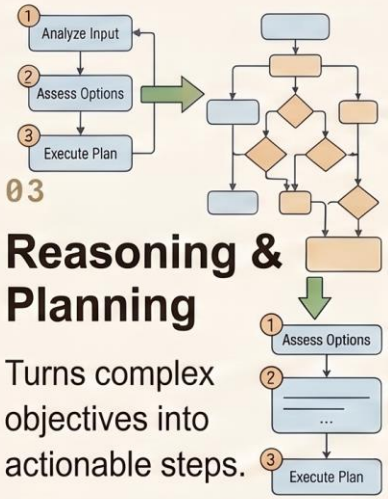
Acts independently to achieve defined goals.



02

Perception

Understands its environment and changes color when a stylized obstacle is introduced.



03

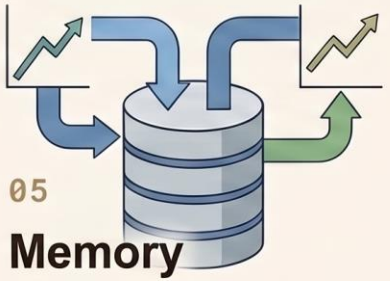
Reasoning & Planning

Turns complex objectives into actionable steps.

04

Tool Use

Leverages software and systems to carry out tasks.



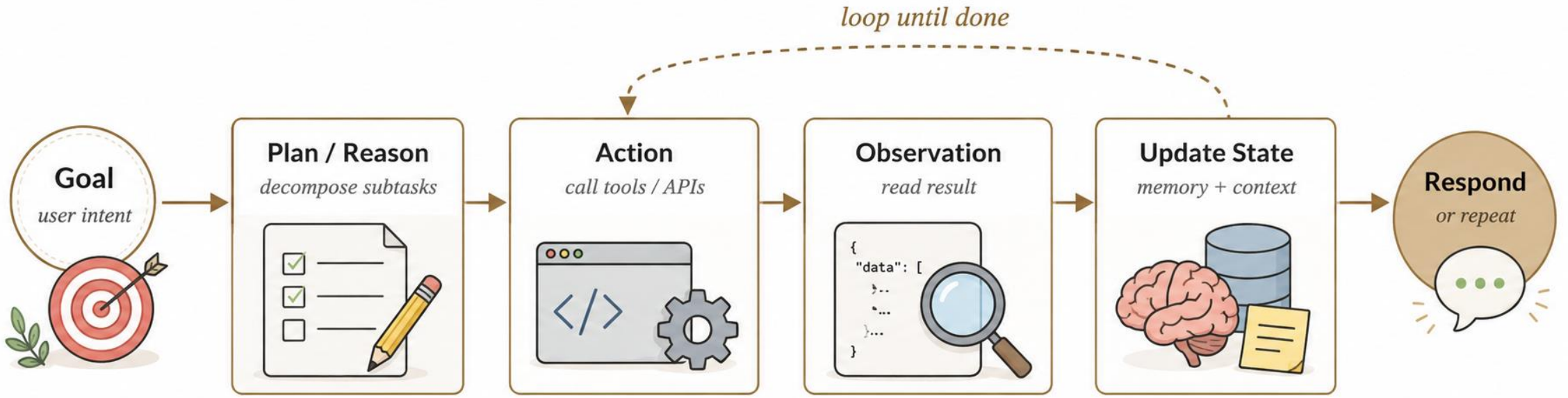
05

Memory

Uses prior context and outcomes to improve over time two simple points.

AI Agent Overview

Core idea: LLM + memory/context + tools + control loop



Reasoning. Break a complex problem into smaller subtasks.

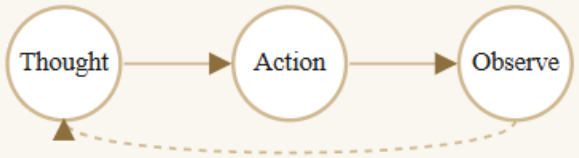
Actions. Use tools, call APIs, gather external information — then decide whether to respond or continue.

AI Agent Overview

PATTERN · 01

ReAct

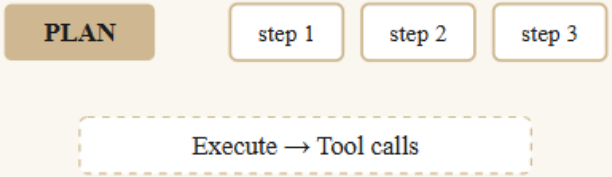
Reasoning and action. Structures the workflow as a loop of **thought → action → observation**.



PATTERN · 02

Plan-and-Execute

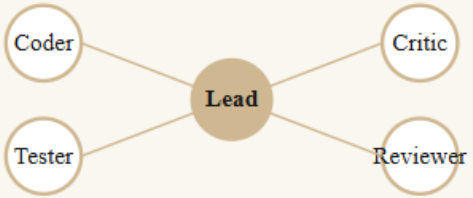
Generate a detailed plan, then **execute steps sequentially**, often using tools.



PATTERN · 03

Multi-agent

Multiple specialized agents **collaborate** to solve complex tasks or simulate environments.



PURDUE UNIVERSITY

AI Agent Overview

Resources

- cloud.google.com/discover/what-are-ai-agents
- ibm.com/think/topics/ai-agents
- atul4u.medium.com/the-complete-agentic-ai-system-design-interview-guide-2026
- github.com/amitshekhariitbhu/ai-engineering-interview-questions
- github.com/huggingface/agents-course
- dev.to/jamesli/react-vs-plan-and-execute-a-practical-comparison-of-llm-agent-patterns
- academy.langchain.com/courses/intro-to-langgraph
- reddit.com/r/cscareerquestionsuk/comments/1qmybi3/ai_engineering_agents_interview_prep

2. Work with AI Agents

For software and ML engineers, using AI agents for coding has become increasingly common.

> A **coding agent** is an AI system that helps with software tasks **beyond simple text generation**, often by operating over code context and iterative feedback.

UP NEXT Why agents — comparing **agent workflow** vs. **one-shot prompting**.

One-shot Prompting

The user gives a **single prompt**; the model returns one response — often a code snippet or explanation.

Useful for simple tasks:

— small coding questions

— syntax lookup

— simple functions

— quick examples

```
# Example — one-shot prompt

› "Write a Python function
   to reverse a linked list."

# Response
def reverse(head):
  prev = None
  while head:
    nxt = head.next
    head.next = prev
    prev, head = head, nxt
  return prev
```

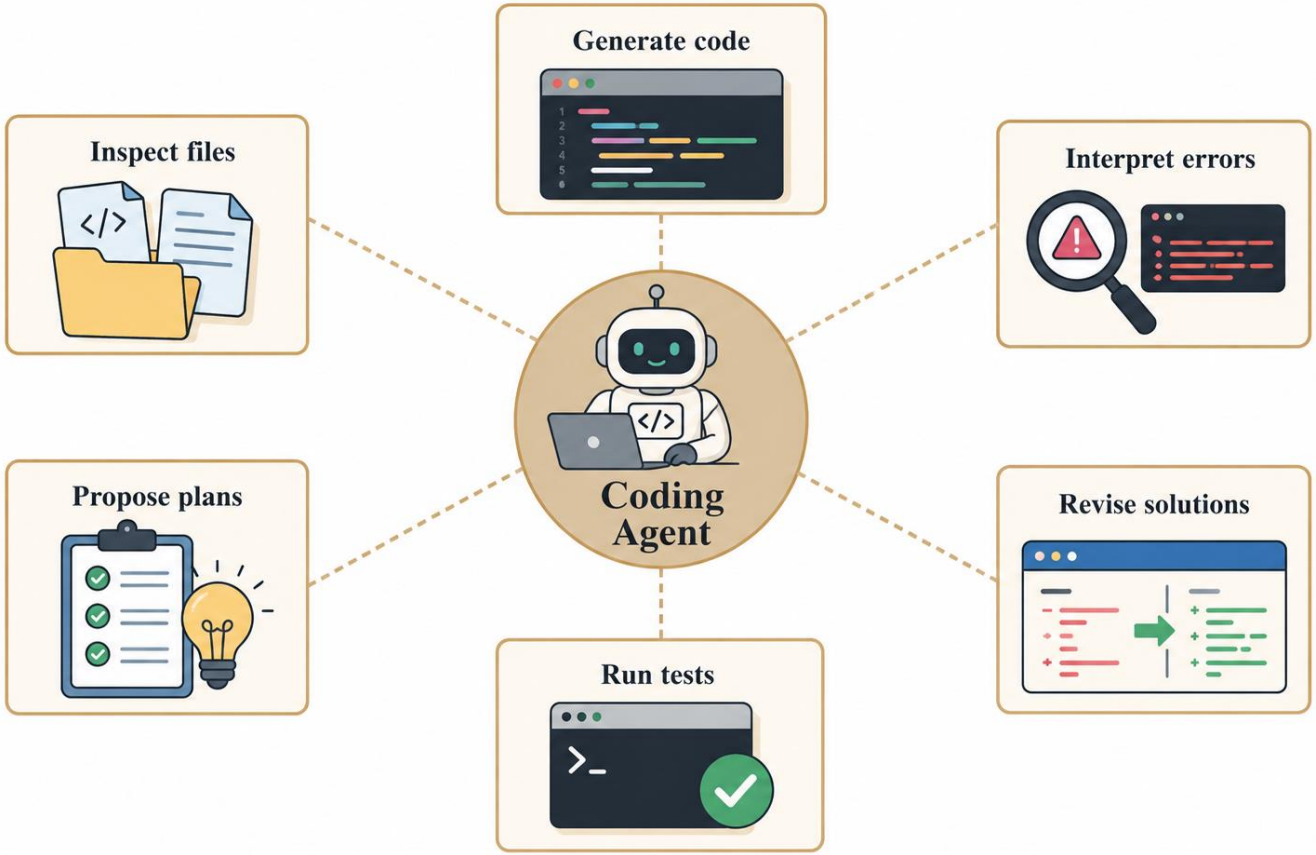
When One-Shot Breaks Down

1. Real coding tasks often require multiple decisions or rounds of refinement.
2. Generated code may compile but fail on edge cases.
3. Generated code may not be compatible with the project structure, APIs, or constraints.

> **In practice:** one-shot prompting is good for **local tasks**, but weak for **larger workflows**, where coding agents can be a better choice.

Agent Workflow

The workflow is **iterative**, not single-turn. The agent acts as a collaborator — not a one-time responder.



PURDUE UNIVERSITY

Comparison

ONE-SHOT PROMPTING

Asks for a **quick answer**.

`single prompt → single response`

AGENTIC WORKFLOW

Manages a **more complex process**.

`plan → act → observe → revise → …`

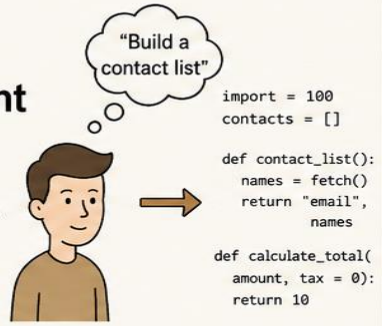
> However, a coding agent is **not "automatically correct"**. It is a capable assistant that still requires **human judgment**.

PURDUE UNIVERSITY

Strengths of Coding Agents

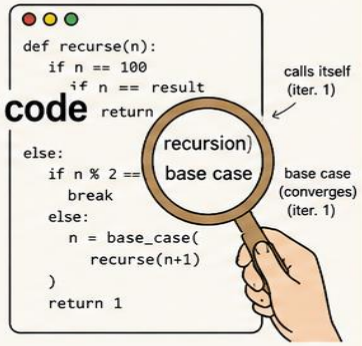
Translate intent

Natural-language implementation drafts.



Explain unfamiliar code

Reads a file or function what it does.

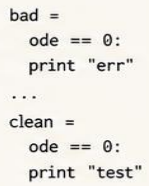


Debug hypotheses

Proposes likely root causes; suggests where to look first.

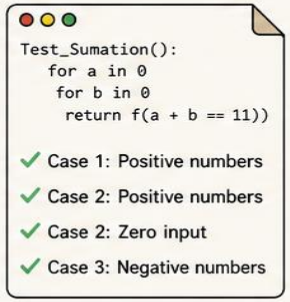
Refactor

Cleans up repetitive code and applies consistent patterns.



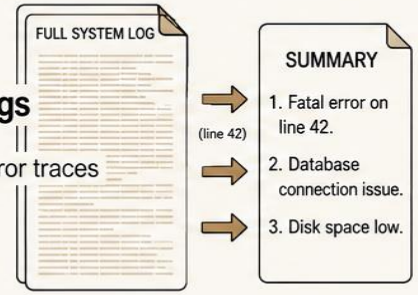
Write tests

Generates unit tests and edge-case examples.



Summarize logs

Boils down long error traces relevant signal.



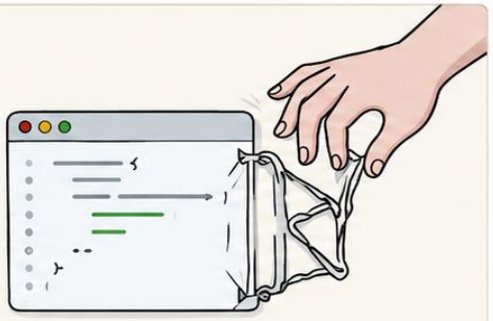
PURDUE UNIVERSITY

Common Weaknesses

RISK

Misunderstanding hidden constraints

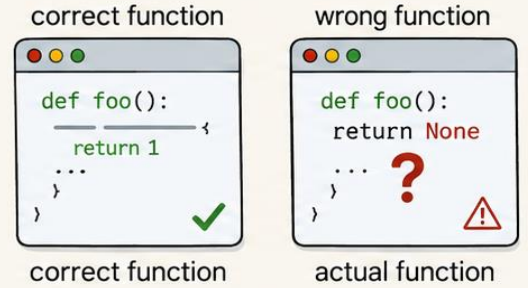
Implicit rules in the manifest or demo that weren't spelled out in the prompt.



RISK

Failing silently

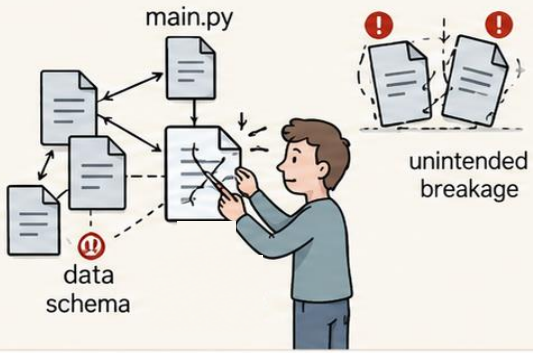
Passive-looking code that doesn't do what it claims to do.



RISK

Fragile cross-file changes

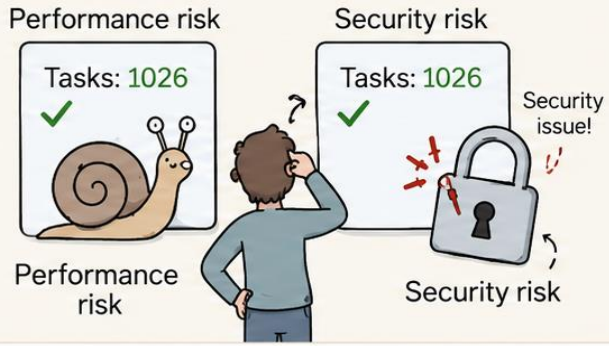
Edits in one file fie in ways that break unrelated parts of the system.



RISK

Overlooking perf perf & security

Functional but slow, or functional but unsafe — both pose risks.



PURDUE UNIVERSITY

A Useful Mindset

DON'T ASK

"Can the agent do coding **for me**?"

ASK

"How do I use the agent to **improve the quality and speed** of my coding process?"

THE HUMAN ROLE

— Defining the problem well

— Setting constraints

— Reviewing trade-offs

— Verifying outcomes

— Testing edge cases

PURDUE UNIVERSITY

Common Coding Agent Choices

TOOL · 01

GitHub Copilot

Fast coding assistance and code autocomplete.

`best for: inline completion`

TOOL · 02

Cursor

Code autocomplete and support for multiple frontier models.

`best for: in-IDE agent runs`

TOOL · 03

Claude Code / Codex

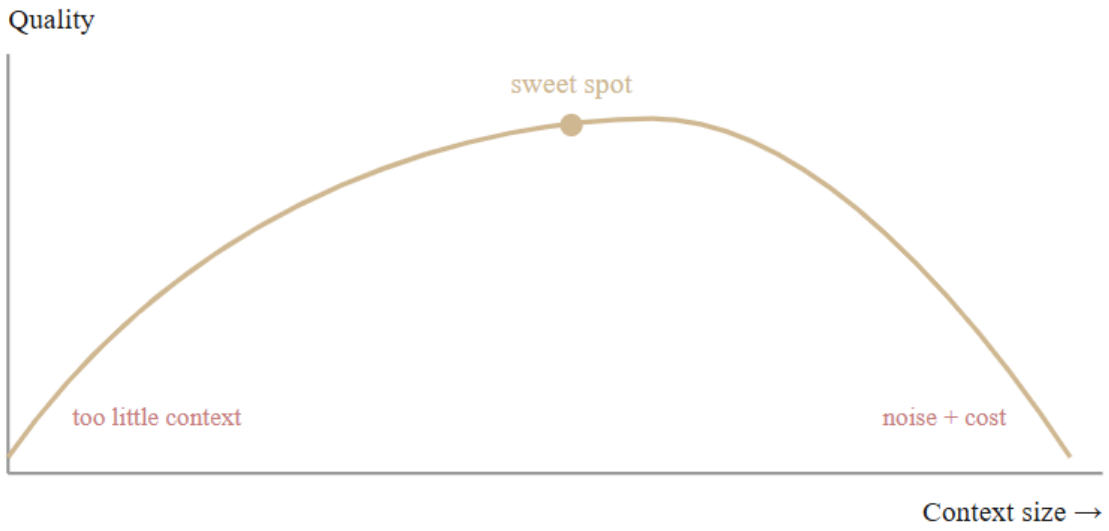
Deep reasoning and complex debugging.

`best for: long-running tasks`

Tutorials from Anthropic: `claude.com/resources/courses`

Context Management

Is more context always better?



IN PRACTICE

— Context windows are **limited**

— Irrelevant context **hurts reasoning**

— Long histories increase **cost & latency**

PURDUE UNIVERSITY

Safe & Wise Usage

⚠ SAFE USAGE

Do **not** accept every suggestion from an AI agent without review.

Especially when it:

— modifies files

— runs commands

— changes project structure

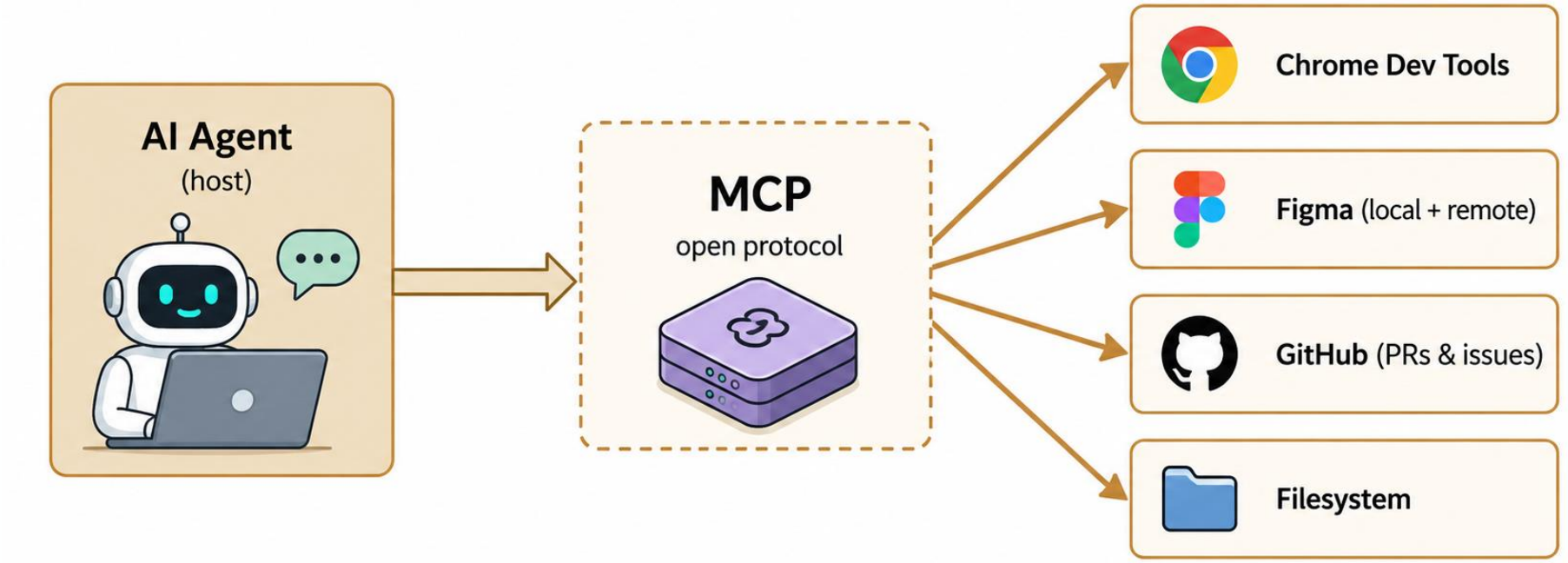
◇ USAGE MANAGEMENT

Use AI agents wisely by **managing token usage**.

Pick the **smallest sufficient model** for the task — faster / smaller models are fine for simple work.

Model Context Protocol (MCP)

An open-source standard for connecting AI applications to external systems.



The agent calls any compliant MCP server with the same protocol — control browsers, design tools, repos, or local files.

Agent Skills

Skills are **reusable, filesystem-based resources** that give agents domain-specific expertise — workflows, context, and best practices.

Unlike prompts, Skills load **on demand** — no need to repeat the same guidance across conversations.

Think of them as **plugins or extensions** for AI agents.

Write your own — or use community-contributed skills.

```
~/.agent/skills/
├── wikipedia/
│   ├── SKILL.md
│   └── scripts/
├── google-search/
│   └── SKILL.md
├── github/
│   └── SKILL.md
└── my-team-style/
    └── SKILL.md

# loaded on demand, not at startup
```

PURDUE UNIVERSITY

Example Skills

SKILL · wikipedia

Search Wikipedia

Searches, retrieves, and summarizes content from English Wikipedia.

SKILL · google-search

Web search

Search the web using Google Custom Search Engine (PSE).

SKILL · github

GitHub operations

Operates GitHub via `gh` for issues, PRs, and repo actions.

COMMUNITY-CONTRIBUTED

- ↗ github.com/openai/skills
- ↗ github.com/anthropics/skills
- ↗ github.com/VoltAgent/awesome-agent-skills

PURDUE UNIVERSITY

Resources — Tools & Protocols

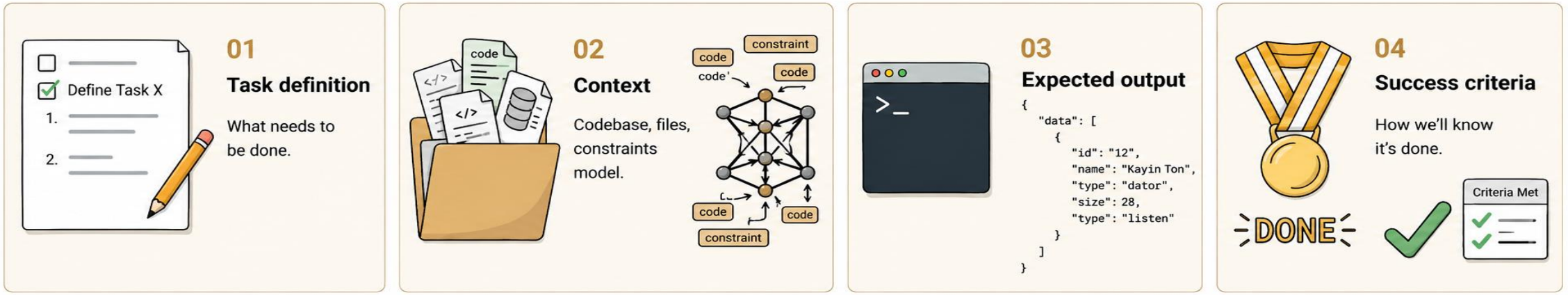
- ↗ github.com/features/copilot
- ↗ cursor.com
- ↗ openai.com/codex
- ↗ code.claude.com/docs/en/overview
- ↗ modelcontextprotocol.io/docs/getting-started/intro
- ↗ anthropic.com/engineering/code-execution-with-mcp
- ↗ developers.openai.com/codex/mcp
- ↗ claude.com/resources/courses

Resources — Skills & MCP Servers

- ↗ github.com/anthropics/skills
- ↗ code.claude.com/docs/en/skills
- ↗ github.com/VoltAgent/awesome-agent-skills
- ↗ developers.figma.com/docs/figma-mcp-server/local-server-installation
- ↗ developers.figma.com/docs/figma-mcp-server/remote-server-installation
- ↗ github.com/ChromeDevTools/chrome-devtools-mcp
- ↗ github.com/github/github-mcp-server
- ↗ github.com/modelcontextprotocol/servers/tree/main/src/filesystem
- ↗ datacamp.com/blog/top-agent-skills

3. Prompting in Coding Context

A strong coding prompt usually includes:



INSTEAD OF

"Fix this code."

BETTER

"Fix this Python function that parses JSON logs. **Keep the same signature**, **don't add dependencies**, and **explain the root cause**."

PURDUE UNIVERSITY

Useful Context Includes…

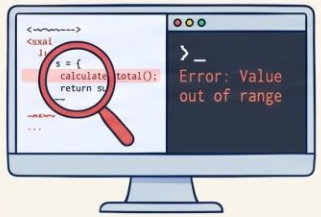
- Language and framework
- Constraints
- Relevant files or code snippets
- Expected behavior or outcome
- Possible error messages to handle

Tip. Ask the AI to **rewrite your prompt** or **identify missing components** before running the task.

PURDUE UNIVERSITY

A Good Debugging Prompt

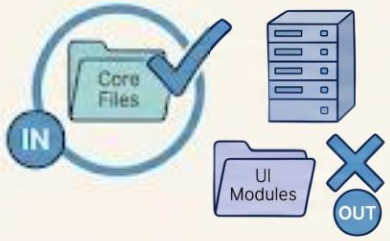
01 **Clear symptom**



Clear symptom

What you observe — exact error or output.

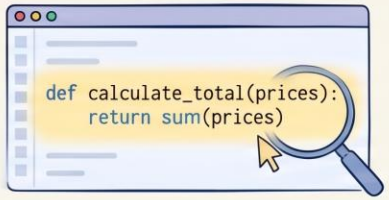
02 **Clear scope**



Clear scope

What's in/out — files, modules, environment.

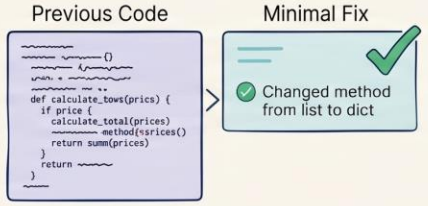
03 **Relevant code**



Relevant code

The function or excerpt under suspicion.

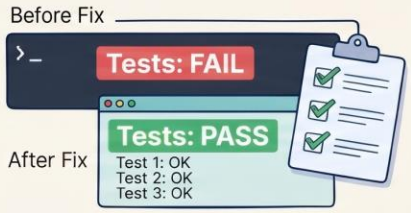
04 **Minimal fix**



Minimal fix

Ask for the smallest change that resolves it.

05 **Verification**



Verification

How you'll test that it's actually fixed.

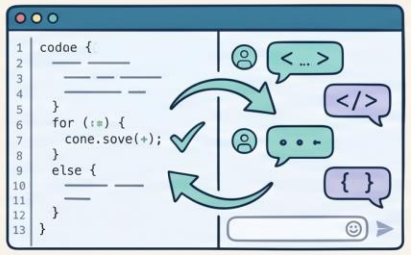
Symptom → Scope → Code → Minimal fix → Verification.

PURDUE UNIVERSITY

Preparing Interviews with AI Agents

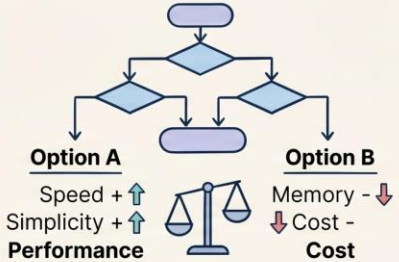
AI agents are most useful when they improve understanding, not just speed.

01 **Practice**



Practice
Coding interviews — interactively.

02 **Explain**

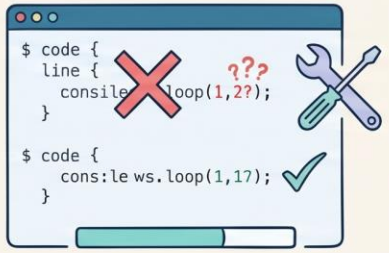


Explain
Algorithms and trade-offs.

03 **Review**

Review
Improve or critique your solution.

04 **Debug**



Debug
Help find and fix issues.

05 **Simulate**

Simulate
Mock technical interview scenarios.

PURDUE UNIVERSITY

Weak Usage vs. Better Usage

WEAK USAGE

"Give me the solution to this LeetCode problem."

→ short-circuits the learning loop

BETTER USAGE

"**Act as an interviewer.** Give me hints gradually without revealing the final solution immediately."

"**Evaluate my solution like a real interviewer.** Identify weaknesses or potential improvements."

PURDUE UNIVERSITY

4. Asking for Clarification Effectively

Clarification is part of problem-solving — in interviews and in collaboration with AI agents.

> A good coding prompt and a good interview problem statement share the **same anatomy**: goal · constraints · input/output · evaluation · …

> In coding tasks, failure often begins **before implementation**. Clarify with yourself **before** prompting the agent.

Clarification Matters Twice

Key idea. Better clarification → better reasoning → better prompts → better code.

A Good Clarification Question Targets…

01 • GOAL

What are we trying to achieve?

The actual outcome behind the request.

02 • SCOPE

What's in, what's out?

Boundaries of the work.

03 • CONSTRAINTS

What must we respect?

Performance, dependencies, style.

04 • I/O

What data shapes?

Inputs, outputs, formats.

05 • EDGE CASES

What if things go wrong?

Unusual or failure cases.

06 • EVALUATION

How is success judged?

Tests, metrics, acceptance.

PURDUE UNIVERSITY

5. Risks and Best Practices

AI can accelerate coding — but it can also accelerate mistakes.

⚠ RISK · 01

Plausible but incorrect logic

Reads well, runs fine, quietly does the wrong thing.

⚠ RISK · 02

Silent bugs or insecure code

Missing validation, unsafe defaults, hidden side effects.

⚠ RISK · 03

Security & privacy

Leaking secrets, calling external APIs without consent.

⚠ RISK · 04

Missing edge cases

Happy-path code that fractures at the boundaries.

Risks and Best Practices

- ✓ Be explicit about requirements and constraints
- ✓ Provide relevant context — not excessive noise
- ✓ Ask for *plans* before large code changes
- ✓ Request *minimal diffs* when debugging
- ✓ Verify with tests and examples
- ✓ Inspect assumptions and external API usage
- ✓ Keep the human in charge of decisions
- ✓ Treat generated code as a *draft*, not a truth source

> **Simple rule.**
> Trust the **workflow** more than any **single output**.

PURDUE UNIVERSITY

Exercise 1

Asking for Clarification Effectively

Discuss in groups — ask **5 clarification questions**.

SCENARIO

Data pipeline for student assignment submissions

Design a system that **collects uploaded files**, **extracts useful information**, and **prepares the data for later analysis**.

Hint — start with: goal · scope · constraints · I/O · edge cases · evaluation

PURDUE UNIVERSITY

Exercise 1

Asking for Clarification Effectively

Discuss in groups — ask **5 clarification questions**.

SCENARIO

Notification system for a mobile application

Send updates to users about **messages**, **reminders**, and **account activity**.

Hint — push vs. pull? delivery guarantees? user preferences? rate limits? failure modes?

Exercise 1

Asking for Clarification Effectively

Discuss in groups — ask **5 clarification questions**.

SCENARIO

File-sharing platform for student group projects

Allow users to **upload**, **share**, and **organize** files.

Hint — who can see what? versioning? size limits? permissions? collaboration model?

PURDUE UNIVERSITY

Exercise 2

Coding Agent Setup

- Task:
  - Choose one agentic tool and install it.
  - First attempt the problem independently for 15–20 minutes.
  - Then use the agent to assist you in implementing problem. Ask the agent to give you hints gradually without revealing the final solution immediately.

PRACTICE PROBLEMS

LEETCODE · MEDIUM

#3 — Longest Substring Without Repeating Characters

leetcode.com/problems/longest-substring-without-repeating-characters

LEETCODE · HARD

#42 — Trapping Rain Water

leetcode.com/problems/trapping-rain-water

PURDUE UNIVERSITY

Exercise 3

Agent-assisted implementation and debugging

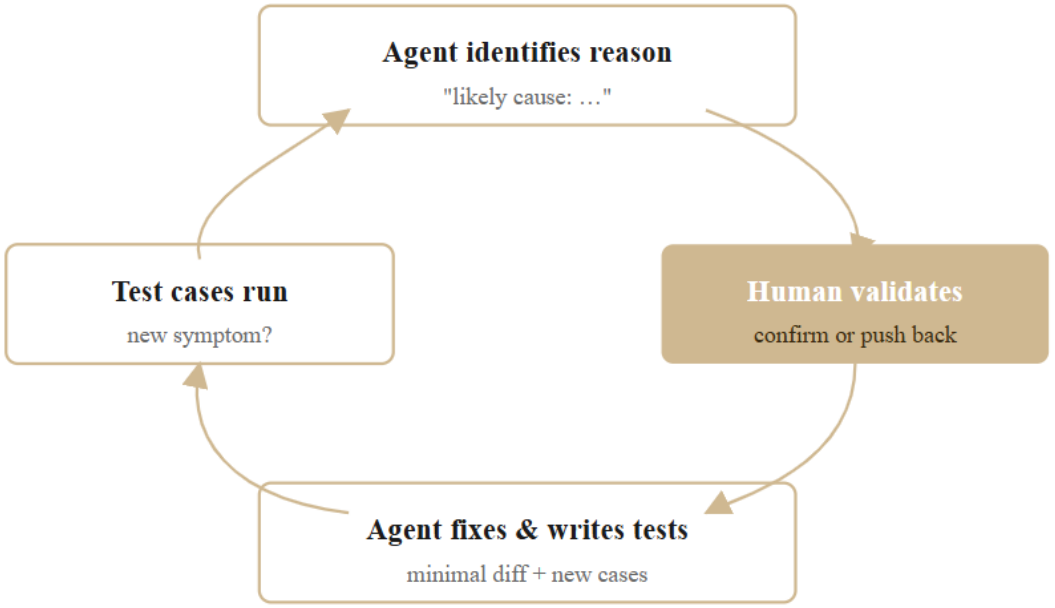
GOAL

Use AI agents to assist debugging and new function implementation.

TASK

1. Choose one agentic tool.
2. `git clone github.com/neubig/starter-repo`
3. Draft a prompt; implement **top-k frequent words** reader (ignore case & punctuation).
4. Improve via a debugging loop with the agent.

THE DEBUGGING LOOP



PURDUE UNIVERSITY

Example Testcase for step 4

Agent-assisted implementation and debugging

```
# Case-insensitive frequency count
def test_top_k_words_case_insensitive():
    text = "Apple apple APPLE banana"
    assert top_k_words(text, 1) == [("apple", 3)]
```

The function should treat **"Apple"**, **"apple"**, and **"APPLE"** as the same token — then return the top-k pairs.

PURDUE UNIVERSITY