UNDERSTANDING AI AGENTS

the fast way

build your own employees

What is an AI Agent? 🤔

An **AI Agent** is like a smart robot assistant that can:

- 🤖 Think and make decisions on its own
- 🤖 Use tools to get things done (like searching the internet or doing math)
- 🤖 Remember what it learned
- 🤖 Work towards completing a goal you give it

Real-World Example 🌟

Imagine asking a robot: "Plan my birthday party!" The AI agent would research party ideas, create a shopping list, suggest games, and even help send invitations!

Principle 1: Goals & Tasks 🎯

Every Agent Needs a Mission!

AI agents work best when they have a clear goal or task to accomplish.

How it works:

- 🤖 You give the agent a **goal** (like "write a story about space")
- 🤖 The agent breaks it into smaller **tasks** (research space, create characters, write paragraphs)
- 🤖 It completes each task step by step

In frameworks: This is called `task definition` or `objective setting`

Principle 2: Memory 🧠

Agents Remember Things!

Just like you remember what you learned in school, AI agents can store and recall information.

Types of Memory:

- 🤖 **Short-term memory:** Remembers the current conversation
- 🤖 **Long-term memory:** Stores important facts for later use
- 🤖 **Working memory:** Keeps track of what it's doing right now

In frameworks: This is called `memory management` or `context retention`

Principle 3: Tools 🛠️

Agents Use Tools to Get Things Done!

AI agents can use different tools, just like you use a calculator for math or a paintbrush for art.

Common Tools:

- 🤖 🔍 **Search tools:** Look up information on the internet
- 🤖 🧮 **Calculator tools:** Solve math problems
- 🤖 📝 **Writing tools:** Create documents or emails
- 🤖 🎨 **Image tools:** Generate or edit pictures

In frameworks: This is called `tool integration` or `function calling`

Principle 4: Reasoning & Planning 💭

Agents Think Before They Act!

AI agents can reason through problems and make plans, just like you plan your homework schedule.

How Reasoning Works:

- 🤖 The agent **observes** the situation
- 🤖 It **thinks** about what to do next
- 🤖 It **decides** on the best action
- 🤖 It **acts** and checks the result
- 🤖 It **learns** from what happened

In frameworks: This is called `reasoning loops` or `chain-of-thought`

Principle 5: Collaboration 🤝

Multiple Agents Can Work Together!

Just like a team of people, multiple AI agents can collaborate to solve bigger problems.

Team Example 👥

Imagine building a website:

- 🤖 **Designer Agent:** Creates the look and colors
- 🤖 **Writer Agent:** Writes all the text
- 🤖 **Coder Agent:** Builds the actual website
- 🤖 **Manager Agent:** Coordinates everyone!

In frameworks: This is called `multi-agent systems` or `agent orchestration`

Principle 6: Learning & Adaptation 📚

Agents Get Smarter Over Time!

AI agents can learn from their experiences and improve their performance.

Ways Agents Learn:

- 🤖 **From feedback:** You tell them if they did well or need to improve
- 🤖 **From examples:** They study examples of good work
- 🤖 **From mistakes:** They remember what didn't work
- 🤖 **From practice:** They get better by doing tasks repeatedly

In frameworks: This is called `feedback loops` or `reinforcement learning`

Popular AI Agent Frameworks 🚀

These are tools that help developers build AI agents:

LangChain

Great for building agents that use language models and chain together different tools

CrewAI

Perfect for creating teams of AI agents that work together on complex tasks

AutoGen

Helps build conversational agents that can chat and collaborate

All these frameworks use the same core principles we just learned!

Cool Things AI Agents Can Do! 🌟

- 🤖 **Personal Assistants:** Help you schedule meetings, answer questions, and manage tasks
- 🤖 **Homework Helpers:** Explain difficult concepts and help you learn
- 🤖 **Game Characters:** Create smart NPCs (non-player characters) in video games
- 🤖 **Creative Partners:** Help write stories, create art, or compose music
- 🤖 **Research Assistants:** Gather information and summarize findings
- 🤖 **Code Helpers:** Assist programmers in writing and debugging code

The Future is Exciting! 🚀

As AI agents get smarter, they'll be able to help us solve even bigger problems like climate change, medical research, and space exploration!

How It All Works Together 🔄

Here's how the principles combine to create a working AI agent:

The Agent Cycle

1. **Receive Goal** → You give the agent a task
2. **Plan & Reason** → Agent thinks about how to solve it
3. **Use Tools** → Agent uses tools to gather info or take action
4. **Remember** → Agent stores what it learned in memory
5. **Learn & Improve** → Agent adapts based on results

You'll say.

So what Tim, I cant code agents.

Wrong!

This presentation was written by an Agent

I’ll show you how:

“if I could just write some basic slides teaching kids what AI agents are, and all the key principles are that would work with Langchain, CrewAI, and all frameworks what would I write. And please write me the slides in HTML format”

But there is a way to cheat with this technology

AI can write agents

LEVEL 1 - BASIC

Direct LLM API Calls

Simple, single-turn API calls to language models with basic prompts and no additional tooling or frameworks

OpenAI API | Anthropic API | Google Gemini API

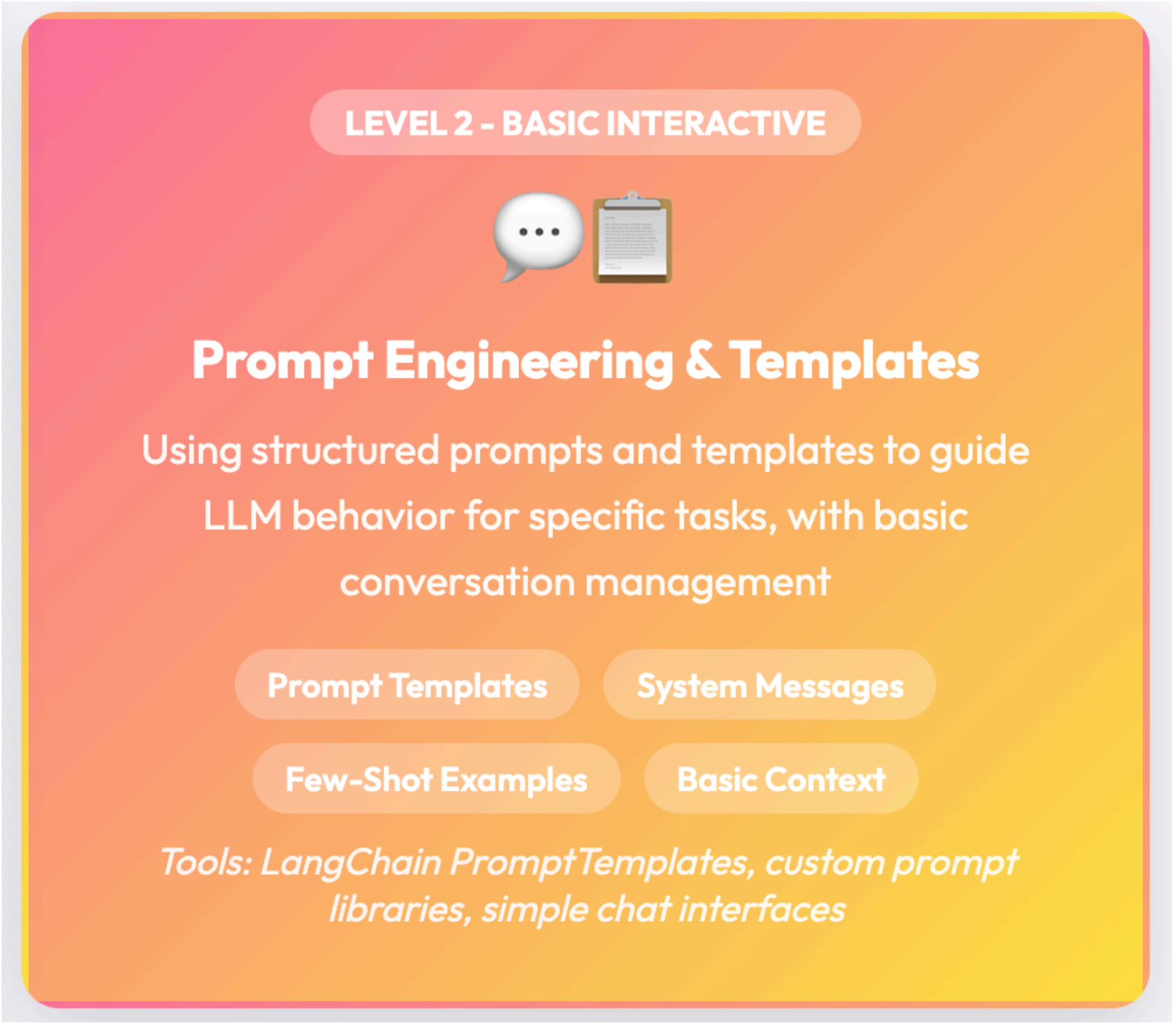
Simple Requests

Tools: REST APIs, basic HTTP clients, minimal SDKs, no frameworks

Cheat sheet prompt:

Level 1 - Direct API Calls:

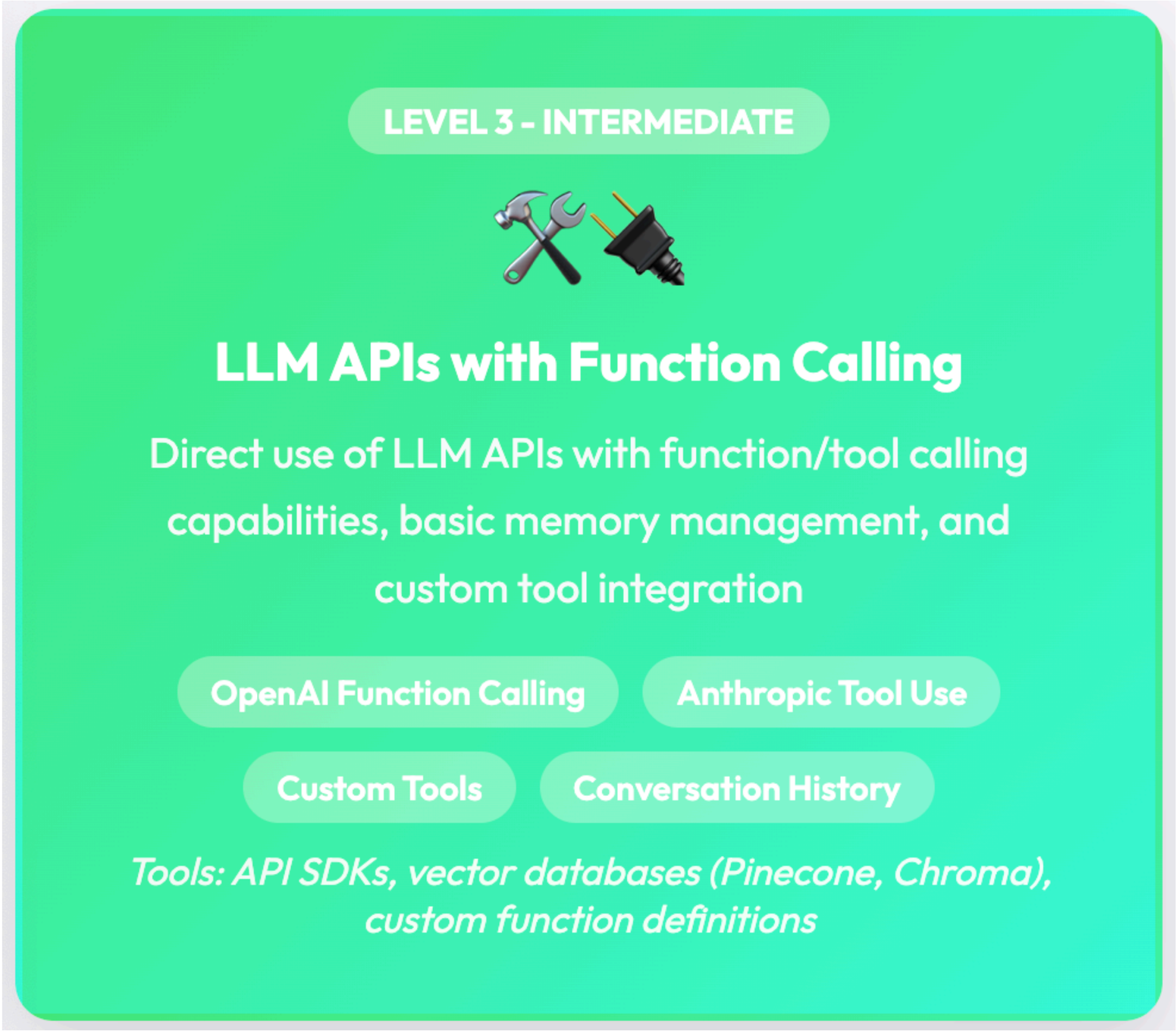
- "Build me a simple Node.js script that calls the OpenAI API..."
- "Create a Python script that uses the Anthropic API to translate..."



Cheat sheet prompt:

Level 2 - Prompt Templates:

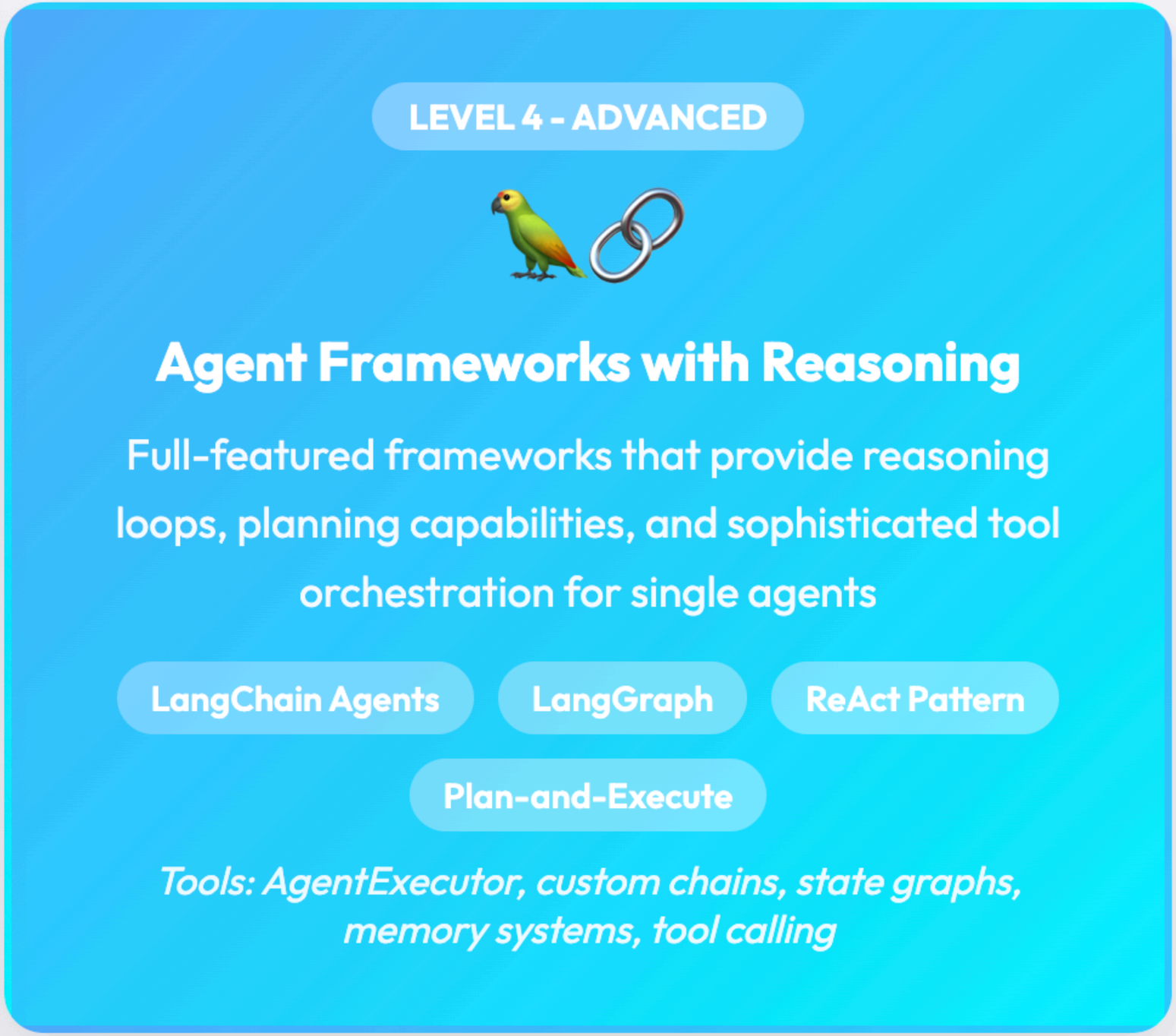
- "Build me a customer service chatbot using LangChain prompt templates..."
- "Create a code review assistant that uses prompt templates..."



Cheat sheet prompt:

Level 3 - Function Calling:

- "Build me a research assistant using OpenAI's function calling that can search the web, calculate statistics..."
- "Create an agent using Anthropic's tool use that can analyze CSV data..."



Cheat sheet prompt:

Level 4 - LangChain/LangGraph:

- "Build me a LangChain agent that can manage software projects using the ReAct pattern..."
- "Create a LangGraph agent that can write research papers using plan-and-execute..."

LEVEL 5 - MOST SOPHISTICATED

Multi-Agent Orchestration Frameworks

Advanced frameworks for building teams of specialized agents that collaborate, delegate tasks, and self-organize to solve complex problems

CrewAI | AutoGen | LangGraph (Multi-Agent) | Custom Orchestration

Tools: Agent roles, task delegation, inter-agent communication, shared memory systems

Cheat sheet prompt:

Level 5 - Multi-Agent Systems:

- "Build me a CrewAI system with multiple agents to create blog posts: Researcher, Writer, Editor, SEO, Manager..."
- "Create an AutoGen multi-agent system for business analysis with Data Analyst, Market Research, Strategy agents..."

A fun technical example "The Roast Bot" - the AI debate team

AI agents that roast each other's arguments in real-time debates!

Agents

💪 Debater A (Pro) - Confident and cocky

😏 Debater B (Con) - Witty and sarcastic

🔍 Fact-Checker - Interrupts with "FACT CHECK!" and calls out lies

🎉 Audience hype agent - "OHHHHH! SHOTS FIRED!" reactions

⚖️ Judge - Declares winner based on facts and logic

Debates on controversial topics (climate change, AI ethics, social media)

- Agents have personality and sass
- Real time fact checking teaches critical thinking
- Competitive and entertaining

📝 **The Project: Multi agent research paper assistant**

A CrewAI system with 7 specialised agents that collaborate to write research papers:

🔍 Research agent - Searches web, finds sources, creates summaries

📋 Outline architect - Creates paper structure and thesis

✍️ Writer agent - Drafts sections in academic style

📚 Citation manager - Formats citations (APA, MLA, Chicago)

✏️ Editor agent - Improves grammar, flow, and clarity

🎯 Critic agent - Evaluates argument strength, finds weaknesses

👔 Coordinator agent - Orchestrates the entire workflow

The workflow:

Topic → Research → Outline → Write → Cite → Edit → Critique → Revise → Complete

📊 **The Project: Multi agent assignment evaluator**

The core prompt:

A CrewAI system with 9 specialised agents that evaluate assignments and provide detailed feedback:

📋 Rubric analyser - Breaks down grading criteria into measurable components

📝 Content evaluator - Assesses if all required elements are present and quality

🎓 Academic standards agent - Checks grammar, citations, academic rigor

💯 Criteria scorer - Assigns scores with justification for each criterion

💡 Improvement advisor - Provides prioritised, actionable recommendations

⭐ Exemplar comparison - Shows what excellence looks like vs. current work

👥 Peer reviewsSimulator - Provides diverse perspectives

👨‍🏫 Professor persona - Evaluates from instructor's perspective

👔 Coordinator - Compiles comprehensive feedback report

The workflow:

Upload assignment + Rubric → Analyse Criteria → Multi Agent Evaluation → Score Prediction → Improvement Recommendations → Action Plan

I want to build a command line app that does this. You might need to create some folders and test files. Label them logically - 📊 The Project:
Multi agent assignment evaluator
The core prompt:
A CrewAI system with 9 specialised agents that evaluate assignments and provide detailed feedback:
📋 Rubric analyser - Breaks down grading criteria into measurable components
📝 Content evaluator - Assesses if all required elements are present and quality
🎓 Academic standards agent - Checks grammar, citations, academic rigor
💯 Criteria scorer - Assigns scores with justification for each criterion
💡 Improvement advisor - Provides prioritised, actionable recommendations
⭐ Exemplar comparison - Shows what excellence looks like vs. current work
👥 Peer reviewsSimulator - Provides diverse perspectives
👨‍🏫 Professor persona - Evaluates from instructor's perspective
👔 Coordinator - Compiles comprehensive feedback report
The workflow:
Upload assignment + Rubric → Analyse Criteria → Multi Agent Evaluation →
Score Prediction → Improvement Recommendations → Action Plan

```
cd ~/Documents/studentproject/assignment-evaluator
source venv/bin/activate
python src/main.py
```