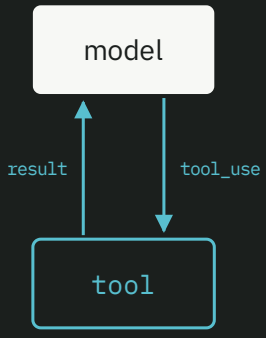
A FIFTEEN-MINUTE CONCEPTUAL GUIDE

Understanding AI Agents in 15 Minutes

What an agent really is, how one is built, and what happened when we ran three of them together.



BY

Ultimate Dimensions

Understanding AI Agents in 15 Minutes
Ultimate Dimensions

First edition, August 2026.

The program in this book was run for real on 25 August 2026 against Anthropic's `claude-sonnet-4-5` model. The output shown in Chapter 6 is unedited. Language models are not deterministic, so your own run will differ in wording and possibly in the number of steps taken.

The program, `agent_demo.py`, lives at github.com/juliusbsimon/understanding-ai-agents and is released under the MIT Licence: use it, change it, and ship it, with attribution. Product names belong to their owners; their features were accurate at the time of writing and will change.

Contents

Fifteen minutes, nine short chapters

CHAPTER 1

The consultant who could not leave the room

Imagine hiring the most widely read consultant in the world, on one condition: she may never leave the meeting room.

She can answer almost any question from memory. She writes beautifully. She can reason through a problem out loud and reach a sensible conclusion. But she cannot check your calendar, cannot look up today's exchange rate, cannot open the spreadsheet on your laptop, and cannot phone the supplier to confirm a price. Everything she tells you is drawn from what she already knew when she walked in, plus whatever you carry into the room and read to her.

That is a plain model chat, a chatbot with no tools. It is a remarkable thing, and for a great many tasks it is exactly what you want. But the moment a task depends on something outside the room, the consultant is stuck, and you become her hands: you fetch the number, paste it in, wait for her answer, then go and act on it yourself.

Now change one rule. Give her a telephone and a short, printed list of people she is allowed to call: the finance desk, the warehouse, a calculator service, a filing clerk who will save documents. Tell her the task, and tell her to keep working, making calls and reading the answers, until she is satisfied she is done. Then come back with the result.

That is an agent. Same consultant, same memory, same writing. The only things that changed are the telephone, the list, and the instruction to keep going. She still never leaves

the room. She never picks up a spreadsheet herself. She asks someone on the list to do it and reads what comes back.

Most modern products marketed as "AI agents," from the simplest chat-app assistant to the most elaborate enterprise automation, are versions of this arrangement. (People define the word differently; this book uses the narrowest definition that still covers what the products actually do.) The rest of this book makes the arrangement precise, shows you one built from scratch in a few dozen lines of code, and then shows what happened when three of them were asked to work together on a real question. If you never intend to write a line of code, read on anyway: Chapter 8 is for you, and it will make much more sense once you have seen what is under the floorboards.

CHAPTER 2

What an agent actually is

At the application boundary, a plain call to a language model is a request-response operation: context goes in, generated text comes out. An agent is that same model with two things added.

The first addition is a set of **tools**. A tool is a piece of ordinary software, a search, a calculator, a database query, a function that sends an email, together with a short description of what it does and what information it needs. The model never runs a tool itself. It only ever sees the descriptions, and when it wants one used it says so, in a structured way: "please call `search_kb` with the query *refund policy*."

The second addition is a **loop**. A small piece of code sits around the model and does three things over and over: ask the model what to do next; if the model asked for a tool, run it and hand back the result; if the model answered in plain words instead, stop and return that answer. That loop is the agent. The model is the mind, the tools are the hands, and the loop is the nervous system connecting them.

Notice what this definition leaves out. There is no requirement that the agent be autonomous for hours, or have a personality, or "learn." Those are features some products add on top. The irreducible core is a model, a tool list, and a loop, and once you have written that loop yourself, every agent framework on the market becomes easier to read, because you can see which part of the loop each feature is decorating.

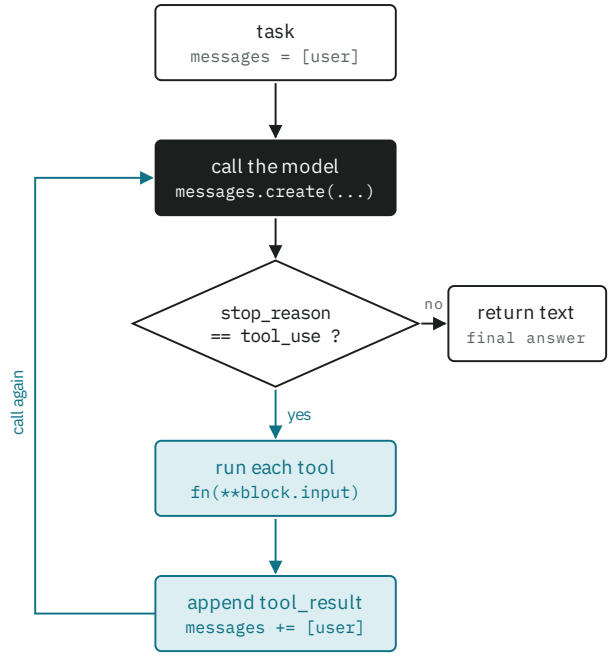
Seven words you will hear constantly

`system prompt`	Standing instructions given to the model before the conversation begins: who it is, what it may and may not do, how to format its answers. In the consultant analogy, the briefing you give her before the task.
`messages`	The running transcript of the conversation, alternating between what you (or your code) said and what the model replied. The model has no memory outside this list. It is the loop's job to keep adding to it.
`tool`	A function you own plus a description the model can read. The description and schema are the model's interface to the tool, so their quality strongly affects whether and how it is used.
`tool_use`	A block in the model's reply that says "call tool X with these arguments." It is a request, not an action. Nothing happens until your code acts on it.
`tool_result`	What your code sends back after running the tool, tagged with the id of the request it answers, so the model knows which question this is the answer to.
`stop_reason`	The one field that drives the loop. Usually the model stopped because it wants a tool run (`tool_use`) or because it has finished (`end_turn`). Other values exist, such as `max_tokens` (the reply was cut off) and `refusal`; the demo ignores them for brevity, and production code must not.
`workflow`	Code you write that decides which model calls or agents run, and in what order. The model decides *how* to do a step. The workflow decides *which* step comes next.

CHAPTER 3

The loop, drawn out

Here is the entire control flow of an agent. There is one decision in it. Everything else is bookkeeping.



One agent, one branch. Everything the model "remembers" is the growing list of messages; the loop exits the first time the model replies without asking for a tool.

Read it from the top. A task arrives and becomes the first message. The model is called with the whole transcript so far and replies. If the reply contains a tool request, the loop runs

the tool, appends the result to the transcript, and calls the model again, now with one more exchange in its memory. If the reply is plain text, the loop returns it and stops.

Three consequences follow directly from the picture, and they explain most of what people find surprising about agents in practice.

The transcript only grows. Every call to the model resends everything that came before. A task that takes twenty tool calls is sending a long and lengthening document twenty times. This is why agents cost more than chatbots, and why serious systems summarise or cache the transcript.

The model decides when it is done. Nothing in the loop says "stop after three searches." The model stops when it judges it has enough. This is the source of an agent's flexibility, and also of its unpredictability: the same task may take four calls one day and seven the next.

The safety valve is yours to add. The small print at the bottom of the figure, a cap on the number of turns, is not part of the model. If you do not write it, nothing prevents a confused agent from looping forever, and the bill from following it.