datacamp

AI Agents Cheat Sheet

Learn about AI agents online at www.DataCamp.com

0 How to Use This Cheat Sheet

This cheat sheet is a companion to our ***Introduction to AI Agents*** course, but it also works on its own. It's organized into numbered sections, so you can go through them in order or jump straight to the part you're most interested in. While the concept of AI agents is broad, this cheat sheet focuses on systems built around language models.

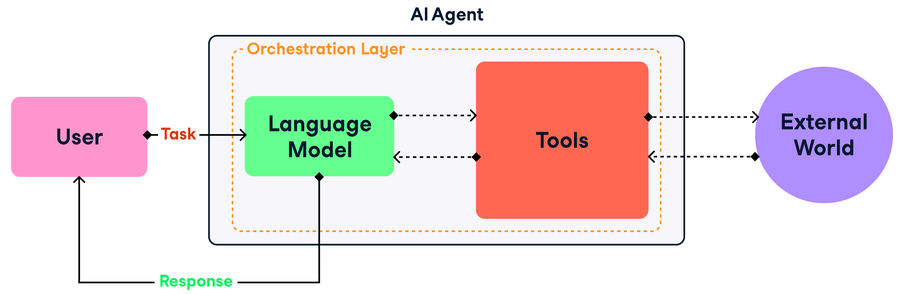
1 What Is an AI Agent?

An **AI Agent** is a system that uses a language model to achieve a user-defined goal. It interacts with its environment by reasoning, planning, and executing actions, often with the help of external tools.

For example, an AI agent could help you book a flight by searching for options and completing the reservation for you.

At its core, an AI agent consists of **three main components**:

- The **language model** powering its reasoning and decision-making;
- The **tools** it uses to interact with the external world and gather information;
- The **orchestration layer** that governs how the agent processes information, plans, and executes actions to achieve its goals.



2 Language Model

A **language model (LM)** is a type of artificial intelligence program designed to understand, generate, and process human language. The LM is the central decision-maker and reasoning engine in an AI agent.

A common misconception is that the underlying LM is the agent. LMs, such as GPT-4o, lack real-world access and can't "think" beyond their training data. They need to be connected to tools (e.g., a weather API) to act as AI agents.

Different types of language models can be used in AI agents—see the table on the right.

Type	Description	Examples	Suitable for
Large Language Models (LLMs)	General-purpose models	GPT-4o, Gemini 2.5 Flash, DeepSeek-V3	Tasks of medium complexity
Small Language Models (SLMs)	Efficient and cost-effective models	Gemma 3n, DeepSeek-R1-Distill-Qwen-1.5B	Simpler tasks
Reasoning Models	Powerful models that generate long chains of thought before generating an answer	OpenAI O3, DeepSeek-R1, Claude Opus 4	Complex problems in coding, math and science

3 Tools

Tools extend an AI agent's capabilities by allowing it to interact with external systems and data. They bridge the gap between a language model's internal capabilities and the outside world.

Tools can take various forms, often aligning with common web API methods like GET, POST, PATCH, and DELETE. Broadly, agents need three types of tools:

Extensions

Extensions bridge an agent and an external API (weather API, flights API, maps API, etc.). Each extension comes with example calls that teach the model which endpoint to invoke and which parameters are required, letting the agent decide at run time when (and how) to execute the API call.

Functions

Functions are custom chunks of code that live outside the model but can be called by it. The model decides which function to invoke and fills in the arguments, yet the code actually runs in your own app (client-side), not inside the agent. This lets developers keep tight control over execution, security, and post-processing.

Data stores

Data stores function as a "knowledge vault" from which agents can retrieve information embedded in existing files and databases. At runtime, the agent taps into these sources (spreadsheets, PDFs, databases, websites, etc.) pulling back just the passages it needs to stay accurate and current.

4 Orchestration Layer

The **orchestration layer** is the cyclical process governing how an AI agent processes information, reasons, plans, and executes actions to achieve its goals. It maintains the agent's memory, state, reasoning, and planning. This layer uses various cognitive architectures and prompt engineering frameworks to guide reasoning and planning:

- **Chain-of-Thought (CoT):** Enables reasoning through intermediate steps, breaking down complex problems into a series of simpler, sequential thoughts.
- **Tree-of-Thoughts (ToT):** Generalizes over CoT prompting by allowing the model to explore various "thought chains."
- **ReAct:** A framework that allows language models to Reason and take Action.

Orchestration patterns generally fall into two categories:

- **Single-agent systems:** A single LM, equipped with tools and instructions, executes workflows in a loop.
- **Multi-agent systems:** Workflow execution is distributed across multiple coordinated agents, often used when complex instructions or tool overload become an issue for a single agent. These can follow:
  - A **Manager** pattern (a central agent delegates tasks via tool calls); or a
  - A **Decentralized** pattern (agents hand off tasks to one another as peers).

5 Agentic Protocols

Agentic protocols are standardized frameworks that facilitate communication and interaction within and between AI agents.

Model Context Protocol (MCP)

Anthropic's model context protocol (MCP) is an open standard designed to standardize how applications provide context to LLMs and connect them to external tools and services.

Example: An AI assistant in Slack could use MCP to pull the latest project updates directly from a project management tool like Asana and display them in your channel.

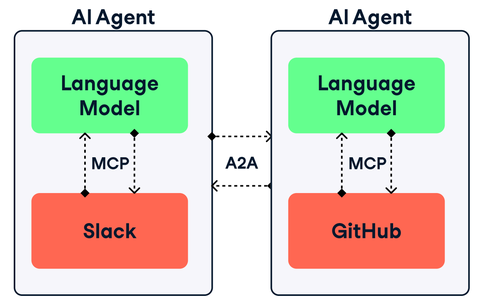
Agent2Agent (A2A) Protocol

Google's Agent2Agent (A2A) protocol enables communication and collaboration between AI agents.

Example: The same AI assistant in Slack, after fetching Asana updates, identifies a critical task that needs a specific report generated. Instead of doing it itself, it uses A2A to securely "talk" to a specialized "Reporting Agent" (which might live on a separate server).

Complementary Protocols

Our example shows that MCP and A2A are complementary protocols that aim to create a more interconnected and powerful AI agent ecosystem.



6 Building AI Agents (What to Choose Depending on Your Goals and Skills)

One-Prompt Agents

One-prompt agents primarily rely on a single, well-crafted prompt to guide the language model's behavior and output for straightforward tasks, minimizing complex multi-step reasoning or tool use.

Use cases: Generating reports, answering complex questions, or performing actions like booking tickets or buying groceries from an online store.

Ease of use: Easy, primarily requiring basic prompt engineering skills.

Manus · Project Mariner · Operator · Perplexity Labs

Coding Agents

Coding agents are AI agents specifically designed to assist with or perform coding tasks, such as generating code snippets, debugging, or refactoring.

Use cases: Automating development tasks, assisting software engineers with code generation or analysis, or enabling natural language programming.

Ease of use: Easy to hard, depending on the required programming knowledge and complexity of the task.

Codex · Google Jules · Devin · Replit · Lovable · Firebase Studio

Workflow-Based Agents

Workflow-based agents offer pre-built functionalities or visual interfaces for constructing agent workflows, requiring little or no coding. These are often geared towards automating specific business processes.

Use cases: Automating repetitive business processes, customer service triage, data entry, or other operational tasks that can be defined through a structured workflow.

Ease of use: Medium, as it requires understanding the workflow logic and configuration.

n8n · Make · Dify · FloWise · Langflow · AirOps

Agentic Frameworks

Agentic frameworks are software libraries and platforms that provide structures, tools, and best practices to help developers build, deploy, and manage AI agents, abstracting away some complexities.

Use cases: Developing custom, complex AI agent applications that require significant integration, specialized logic, or novel architectures.

Ease of use: Generally medium to hard, depending on framework complexity and desired customization.

OpenAI Agents SDK · Google's ADK · LangGraph · CrewAI · Smolagents · LlamaIndex