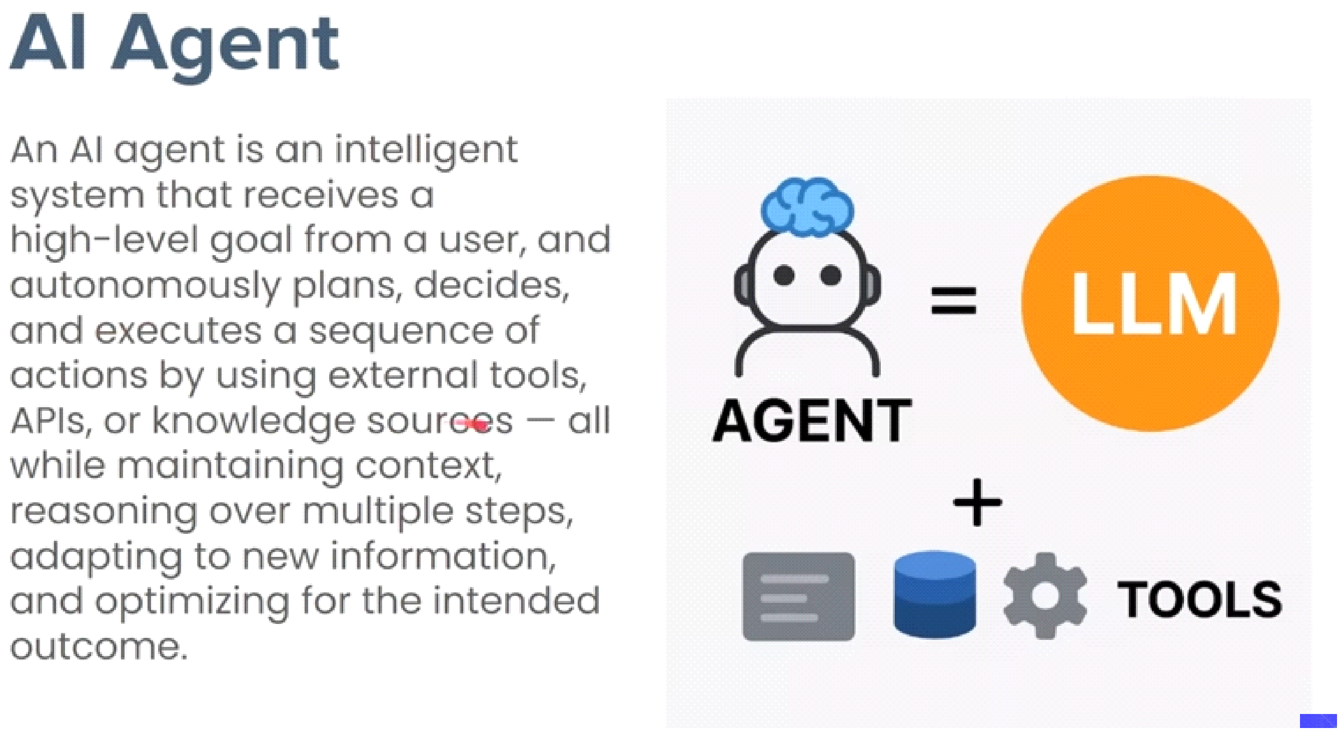
Building end-to-end AI Agent in LangChain

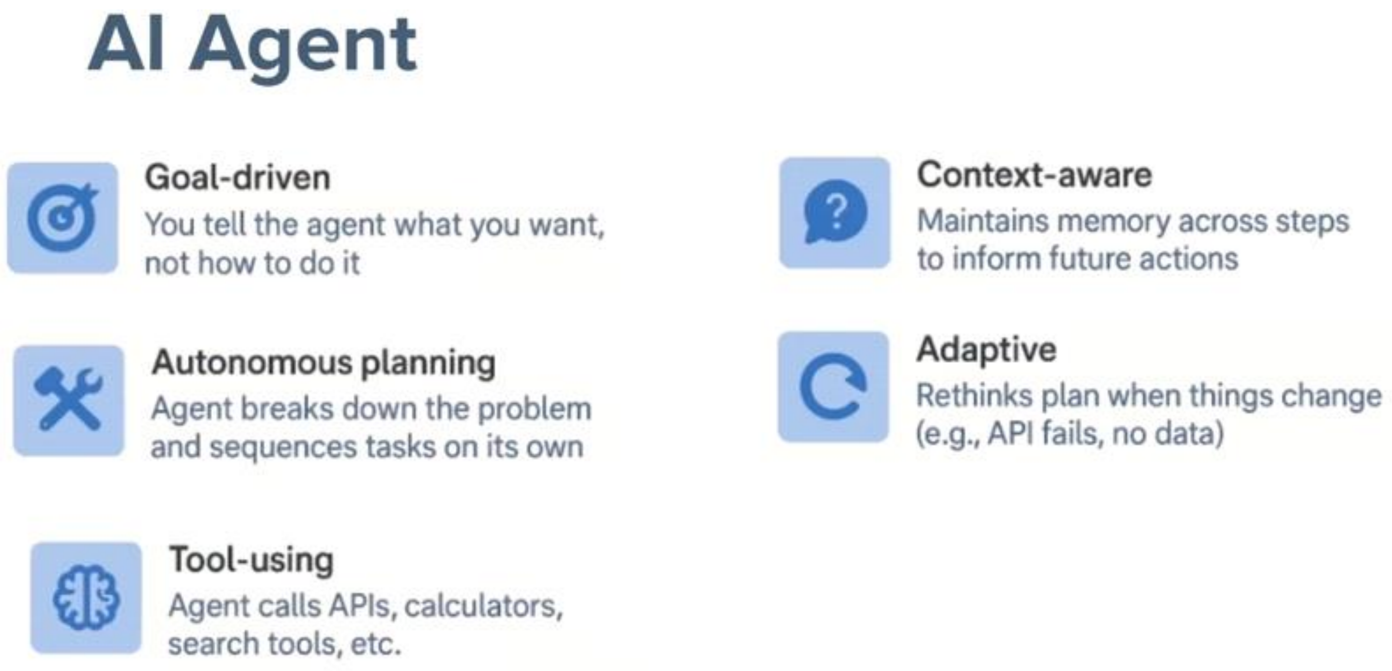
23 May 2025 23:07



AI Agent

An AI agent is an intelligent system that receives a high-level goal from a user, and autonomously plans, decides, and executes a sequence of actions by using external tools, APIs, or knowledge sources — all while maintaining context, reasoning over multiple steps, adapting to new information, and optimizing for the intended outcome.

AGENT
=
LLM
+
TOOLS



AI Agent

Goal-driven
You tell the agent what you want, not how to do it

Context-aware
Maintains memory across steps to inform future actions

Autonomous planning
Agent breaks down the problem and sequences tasks on its own

Adaptive
Rethinks plan when things change (e.g., API fails, no data)

Tool-using
Agent calls APIs, calculators, search tools, etc.

1. ReAct

01 May 2025 14:17

ReAct is a design pattern used in AI agents that stands for Reasoning + Acting. It allows a language model (LLM) to interleave internal reasoning (Thought) with external actions (like tool use) in a structured, multi-step process.

Instead of generating an answer in one go, the model thinks step by step, deciding what it needs to do next and optionally calling tools (APIs, calculators, web search, etc.) to help it.

```
Thought: I need to find the capital of France.
Action: search_tool
Action Input: "capital of France"
Observation: Paris
Thought: Now I need the population of Paris.
Action: search_tool
Action Input: "population of Paris"
Observation: 2.1 million
Thought: I now know the final answer.
Final Answer: Paris is the capital of France and has a population of ~2.1 million.
```

ReAct is useful for:

- Multi-step problems
- Tool-augmented tasks (web search, database lookup, etc.)
- Making the agent's reasoning **transparent and auditable**

It was first introduced in the paper:
"*ReAct: Synergizing Reasoning and Acting in Language Models*" (Yao et al., 2022)

2. Agent & Agent Executor

01 May 2025 14:49

`AgentExecutor` orchestrates the **entire loop**:

1. Sends inputs and previous messages to the agent
2. Gets the next `action` from agent
3. Executes that tool with provided input
4. Adds the tool's `observation` back into the history
5. Loops again with updated history until the agent says `Final Answer`.

3. Creating an Agent

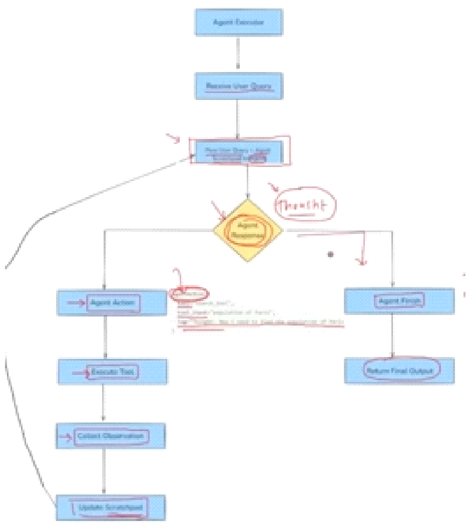
01 May 2025 16:29

```
agent = create_react_agent(
    llm=llm,
    tools=[search_tool],
    prompt=prompt
)
```

4. Creating an Agent Executor

01 May 2025 16:30

```
agent_executor = AgentExecutor(
    agent=agent,
    tools=[search_tool],
    verbose=True
)
```



```
AgentAction(
    tool="search_tool",
    tool_input="population of Paris",
    log="Thought: Now I need to find the population of Paris"
)
```