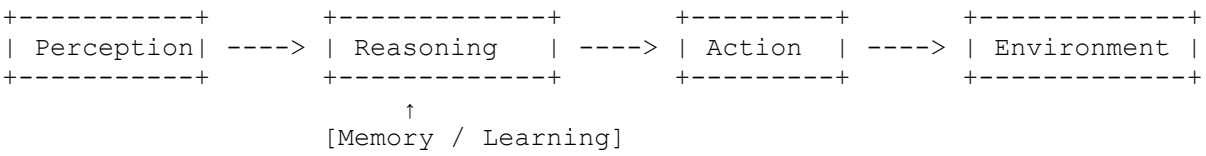
🔍 What is an AI Agent?

AI Agent: A software entity that **perceives** its environment, **reasons** about it, and **acts** to achieve specific goals.

🧠 Core Components:

1. **Perception**: Takes input from the environment (e.g., sensors, API calls, user input).
2. **Decision Making**: Applies rules, machine learning, or reasoning to decide what to do.
3. **Action**: Performs actions in the environment (e.g., API calls, text replies, movement).
4. **Learning** (optional): Improves performance using past experience (Reinforcement Learning, ML).

🔄 Typical AI Agent Workflow



🛠️ Open-Source Frameworks to Build AI Agents

Category	Framework	Use Case
General AI Agent	□ **LangChain**	Language agents (chatbots, tools)
Agent Workflow Graph	□ **LangGraph**	Visual/memory-driven agent workflows
Multi-Agent Orchestration	□ **CrewAI**	Team-based AI agents with roles/goals
Language Agents	□ **Haystack**	Question answering, Retrieval-Augmented Generation (RAG)
Agent Framework	□ **Auto-GPT**	Autonomous LLM agents with long-term memory & goal tracking
RL Agents	🔘 **OpenAI Gym**	Reinforcement learning training environments
Game AI	🔘 **PettingZoo**	Multi-agent environments for games/simulations
Planning Agents	🔘 **PyDatalog**	Declarative logic programming for reasoning
Robotics / Real-world Agents	🔘 **ROS** (Robot Operating System)	Sensor/actuator-based real-world agents

Beginner Roadmap to Building AI Agents

✅ Step 1: Learn the Fundamentals

- Python (must-have language for AI)
- Basic AI concepts: agents, environments, search, planning
- ML basics (classification, regression, reinforcement learning)

✅ Step 2: Choose a Use Case

Example ideas:

- Chatbot that books flights
- AI that solves puzzles
- Game-playing agent
- Research assistant

✅ Step 3: Pick a Framework

Start with **Langchain**, **Haystack**, or **Auto-GPT** if building language agents.

✅ Step 4: Build a Simple AI Agent (LangChain Example)

Goal: Build a chatbot agent that can search the web and respond.

Tech stack:

- Python
- LangChain
- OpenAI API (or other LLMs)
- Tool: SerpAPI (for search)

✅ Goal

Build a LangChain agent using LLaMA (via Ollama) that can answer questions with tool usage (like web search).

🔗 What You'll Use

Tool	Purpose
□ LangChain	Agent framework
🦙 Ollama	Local LLM server (runs LLaMA, Mistral, etc.)
🧠 LLaMA	The actual language model
🔎 SerpAPI (optional)	Web search tool

Step-by-Step Guide

🛠️ 1. Install Dependencies

```
# Base tools
pip install langchain langchain-community langgraph

# Ollama client
pip install ollama
```

💻 2. Start Ollama with LLaMA

If you haven't already:

```
# Download and run llama3
ollama run llama3
```

You can also try `mistral`, `llama2`, etc.

🧠 3. Create Your LangChain Agent with Ollama LLM

Simple Langchain example using Llama 3 modle with Ollama

```python
from langchain_ollama import OllamaLLM

def main():
    llm = OllamaLLM(model="llama3", temperature=0.7)

    prompt = "Tell me a joke about computers."
    response = llm.invoke(prompt)
    print("Response:", response)

if __name__ == "__main__":
    main()
```

Output:-

Response: Why did the computer go to therapy?

Because it had a little "glitch" in its program!

Now with simple Agent Here's a minimal working script using LangChain with Ollama + LLaMA:

```
from langchain_ollama import OllamaLLM
from langchain.agents import initialize_agent
from langchain.agents.agent_types import AgentType

def main():
    llm = OllamaLLM(model="llama3", temperature=0.7)

    # No tools passed - empty list
    tools = []

    # Initialize a simple chat zero-shot agent (if available)
    # If your version doesn't have CHAT_ZERO_SHOT, fallback to using LLM
directly
    try:
        agent = initialize_agent(
            tools=tools,
            llm=llm,
            agent=AgentType.CHAT_ZERO_SHOT,
            verbose=True,
        )
        prompt = "Tell me a joke about computers."
        response = agent.run(prompt)
    except AttributeError:
        # Fallback if CHAT_ZERO_SHOT is not present
        print("AgentType.CHAT_ZERO_SHOT not found, running LLM directly...")
        response = llm.invoke("Tell me a joke about computers.")

    print("Response:", response)

if __name__ == "__main__":
    main()
```

Output:-

AgentType.CHAT_ZERO_SHOT not found, running LLM directly...

Response: Why did the computer go to therapy?

Because it had a little "glitch" in its programming!

✅ Step 5: Add Abilities (Tools/Plugins)

- Add APIs: Google Calendar, Notion, etc.
- Add memory: store conversations
- Add planning: let the agent plan steps (like Auto-GPT)

✅ Step 6: Deploy Your Agent

- Web app: Use **Streamlit**, **Flask**, or **Gradio**
- Discord/Slack bot: Integrate via APIs
- Mobile: Wrap using Flutter or React Native

✅ Step 7: Learn More & Contribute

- Follow open-source projects
- Contribute to LangChain, Haystack, or Auto-GPT
- Build your own modular AI agent framework

🧒💻 Suggested Beginner Projects

Project	Tools
ChatGPT-style Bot	Langchain, OpenAI, Streamlit
News Summarizer Agent	Haystack, Newspaper3k
Auto Researcher Agent	Auto-GPT, Web search APIs
Game AI (Tic Tac Toe)	Python, Simple rules/RL
Voice Assistant	SpeechRecognition, Langchain

Resources

- LangChain Docs
- OpenAI Cookbook
- Fast.ai Course
- Andrej Karpathy's Neural Networks Zero to Hero