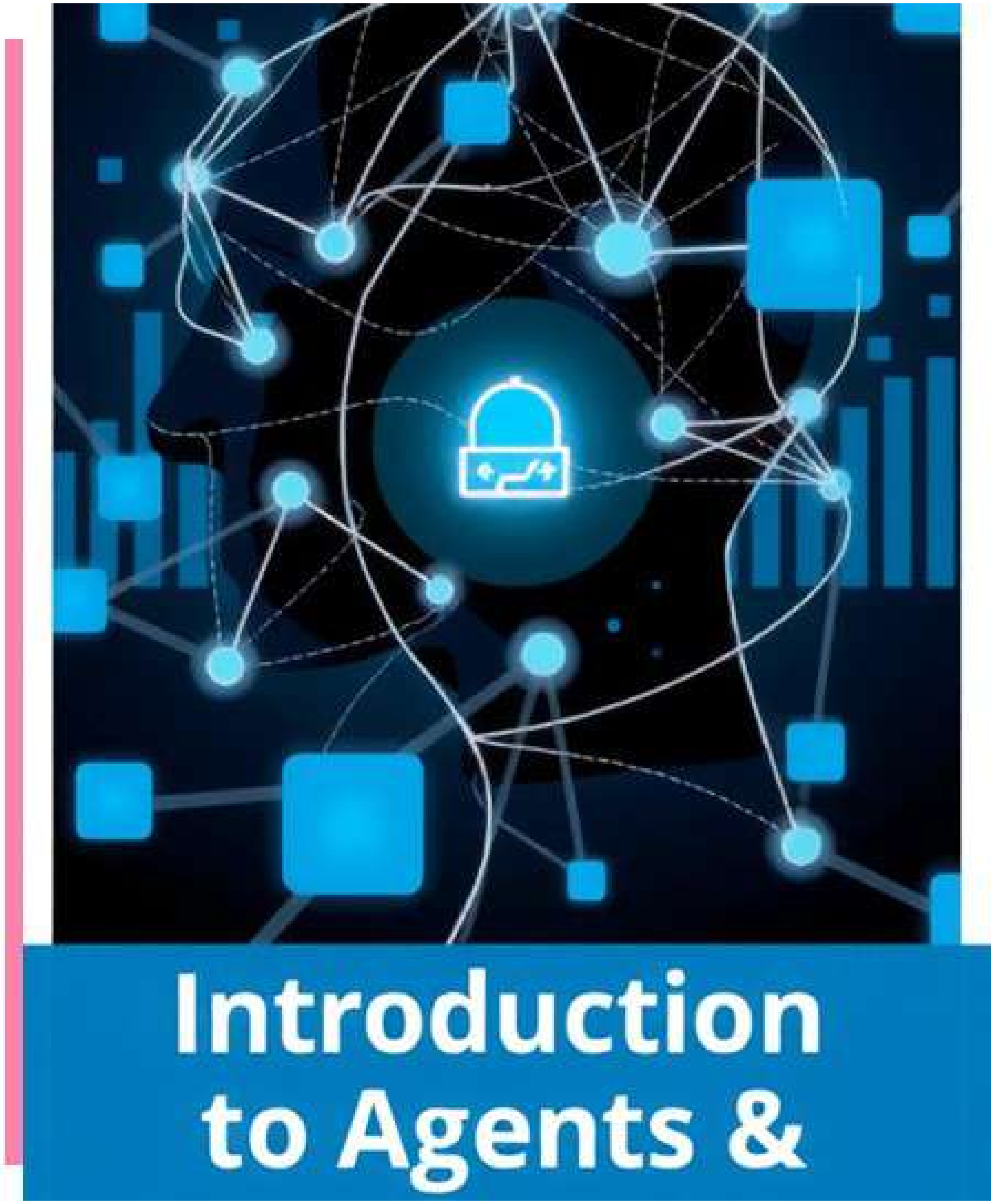
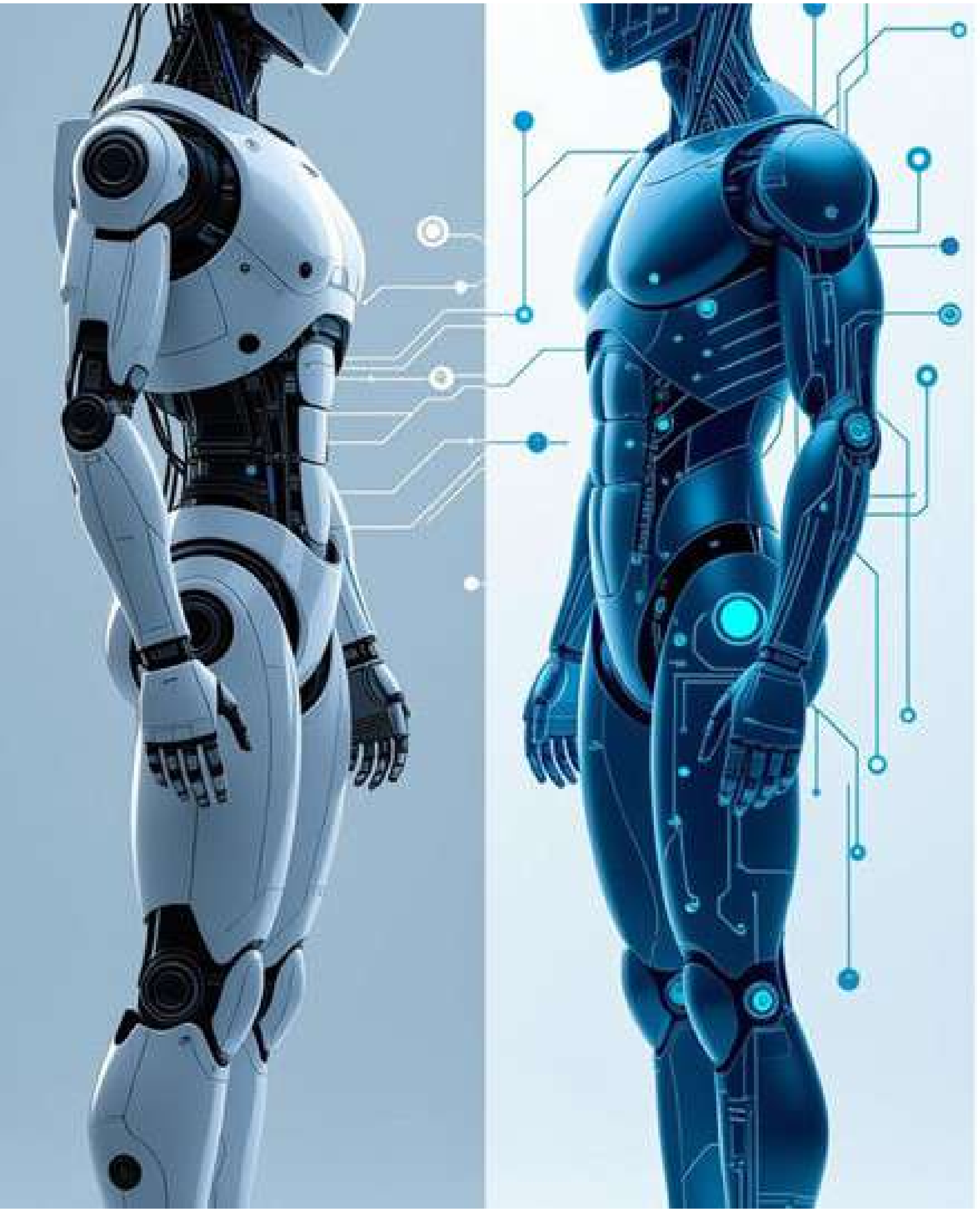
Agentic AI

Talk of the Town

Introduction to Agents & Agentic AI

- ✓ This lecture will explore the fundamental concepts, architectures, and applications of agents in artificial intelligence.
- ✓ We'll examine the transition from simple agents to complex agentic systems, emphasizing their role in creating autonomous, decision-making entities.
- ✓ The focus is to equip you with the knowledge to design and analyze intelligent agents.

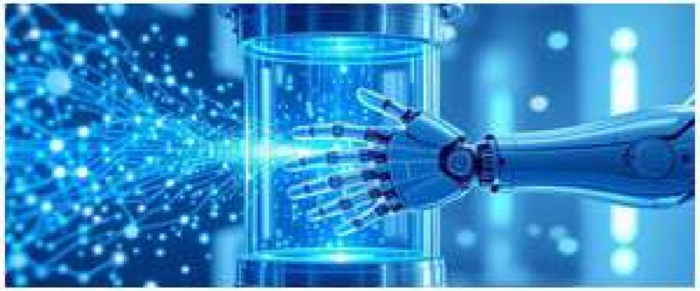




Key Definitions: Agent and Agentic AI

- **Agent**: An Agent is an entity that perceives its environment through sensors and acts upon that environment through effectors. Agents can be hardware (robots) or software (software agents).
- **Agentic AI:** Agentic AI refers to AI systems that exhibit autonomous, intelligent behavior, capable of making decisions and acting independently to achieve specific goals. It involves more than just executing pre-programmed instructions.

Understanding Agents: Passive vs. Active



Passive Agents

- Act only based on current input (no internal state or goal).
- No memory, no learning, no planning.
- Cannot adapt to change — behavior is fixed.
    - Simple reflex agent (IF obstacle → turn left).
    - Light that turns on when motion is detected.
    - Basic language model giving one-line definition with no memory.

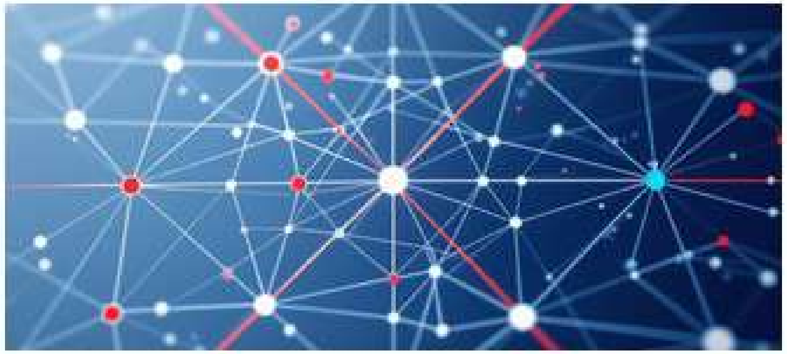
Active Agents

- Maintain internal state or memory.
- Learn from past experiences or plan based on goals.
- Can pursue objectives, adapt, and optimize behavior over time.
    - Goal-based agent (e.g., GPS route planner).
    - Utility-based agent (e.g., AI choosing most efficient workflow).
    - LLM agent using tools + memory + reasoning.

Mapping Passive/Active to Classical Architectures

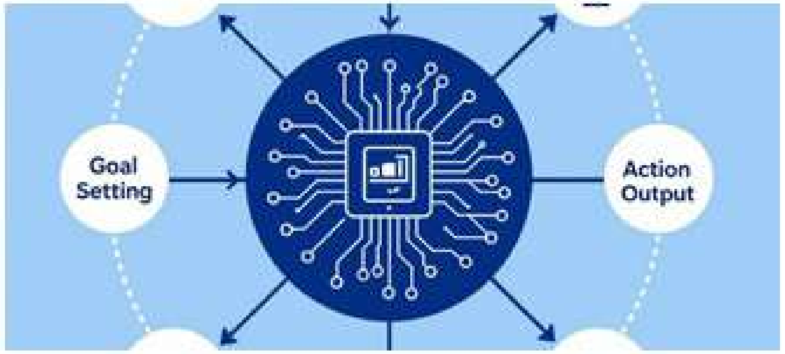
Agent Architecture	Passive or Active?	Reason
Simple Reflex Agent	Passive	No memory, no goals, no reasoning
Model-Based Reflex	Semi-Active	Has internal state, limited adaptation
Goal-Based Agent	Active	Plans actions toward goal using world model
Utility-Based Agent	Active	Chooses optimal action by evaluating outcomes

Reactive vs. Deliberative Agents



Reactive Agents

- **Reactive Agents** respond immediately to their **environment** based on **pre-defined rules** or **reflexes**. They lack **internal planning** or **reasoning**, acting solely on immediate stimuli. This makes them suitable for simple, predictable environments where quick responses are necessary.



Deliberative Agents

- **Deliberative Agents** engage in **reasoning**, **planning**, and **decision-making** based on a **model of the world**. They set **goals** and **strategize** to achieve them, allowing them to handle complex and unpredictable environments.

From Internal State to Behavior Strategy

We just discussed whether agents are passive or active — that's about whether they have memory or goals.
Now we'll go deeper into how they act.
Do they instantly react to their inputs, or do they take time to think and plan?

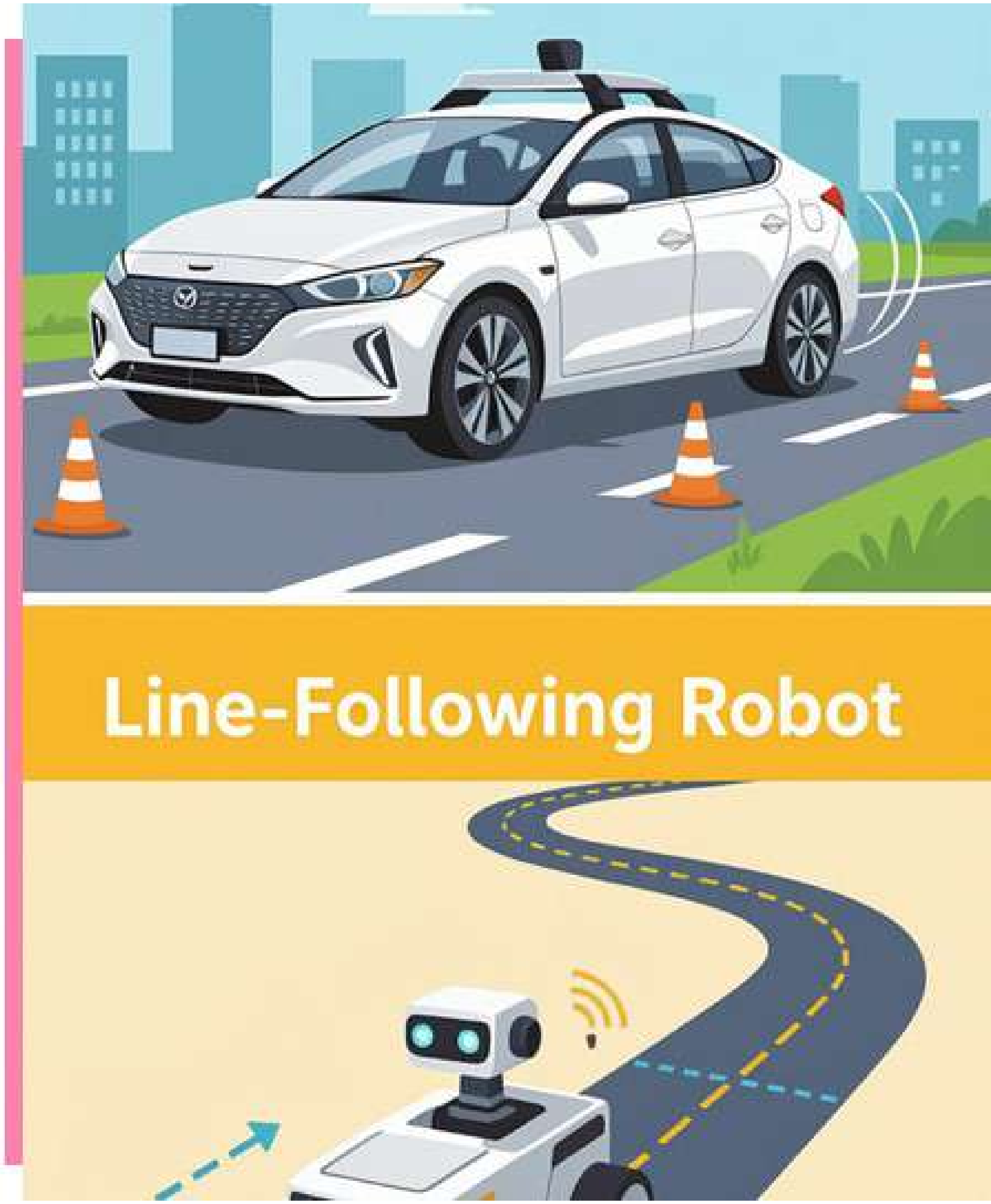
Characteristic	Passive vs. Active Agents	Reactive vs. Deliberative Agents
Memory / Internal State	Passive → No memory Active → Has memory	Both can have memory, but use it differently
Goal-Oriented	Active agents pursue goals	Deliberative agents explicitly plan to reach goals
Planning	Not in Passive agents	Only Deliberative agents plan actions
Speed of Response	Passive = Instant	Reactive = Instant, Deliberative = Slower but smarter
Decision-Making	Predefined rules or reflexes	Deliberative agents use reasoning and evaluation

Example: Reactive Agents in Robotics

Reactive Agents excel in tasks requiring quick responses to immediate stimuli:

- **Obstacle Avoidance in Autonomous Vehicles:** Sensors detect obstacles, and the agent reacts by steering to avoid collisions.
- **Line Following Robots:** Robots follow predefined paths using simple sensor-based reactions.

These examples illustrate how reactive agents are effective in dynamic environments where quick responses are critical.

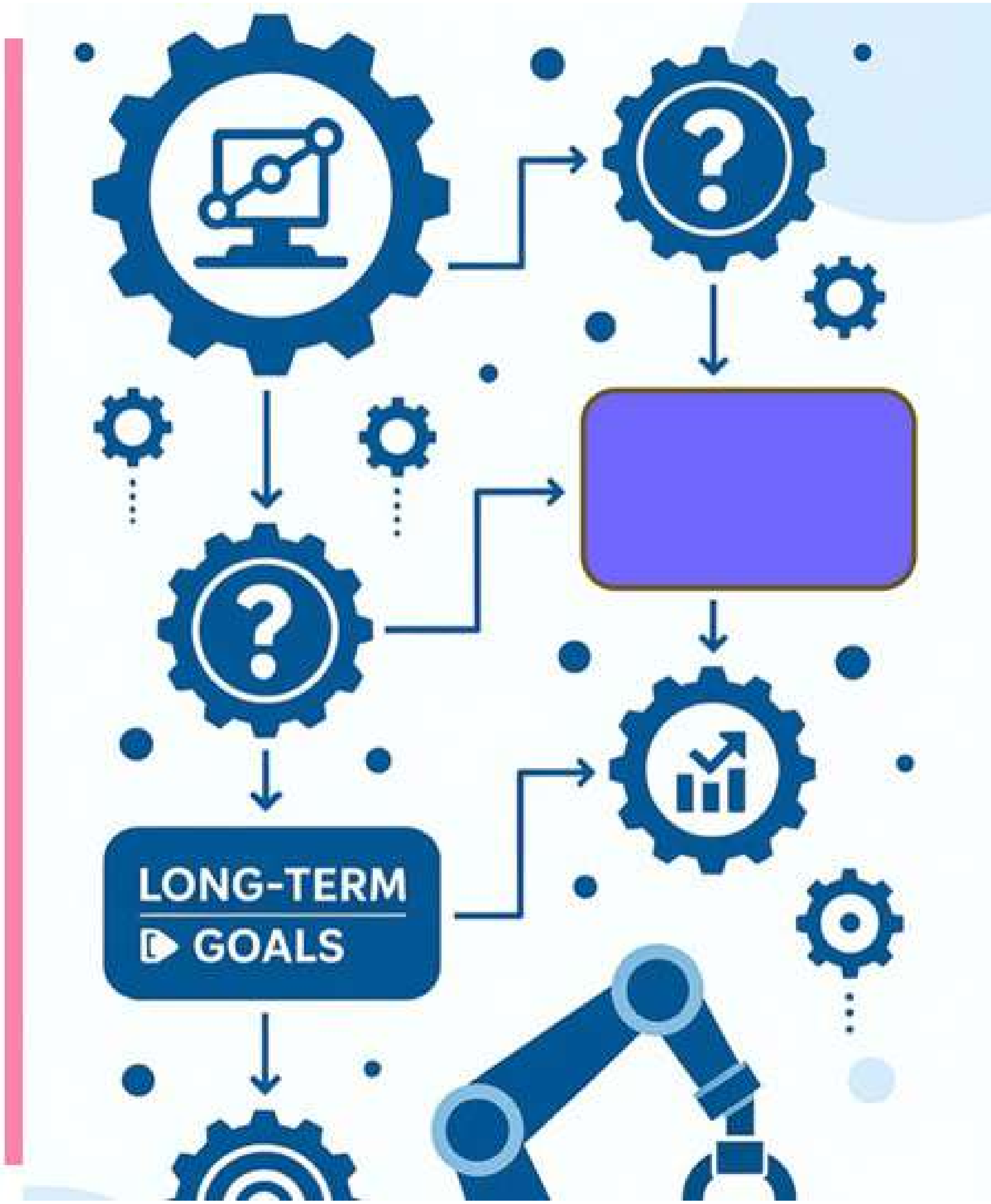


Example: Deliberative Agents in Decision-Making

Deliberative Agents are essential for complex decision-making scenarios:

- **Advanced Robotics:** Robots plan complex movements, such as navigating unknown terrains or manipulating objects.
- **Automated Decision-Making Systems:** AI systems analyze data, predict outcomes, and make strategic decisions in fields like finance or logistics.

Deliberative agents are beneficial when considering long-term goals and uncertain future outcomes.



Software vs. Embodied Agents

Feature	Software Agents	Embodied Agents
Form	Purely digital	Physical (robots, sensors, actuators) or virtual environment
Environment	Virtual (web, code, APIs)	Real-world or simulated environments
Examples	Web crawlers, Chatbots, AutoGPT	Drones, Self-driving cars, Robot arms, game Agents
Sensors/Effectors	APIs, memory, input parsers	Cameras, LIDAR, motors, manipulators
Cost	Low setup, scalable	Expensive hardware, setup needed
Focus	Cognitive capability	Physical interaction and navigation

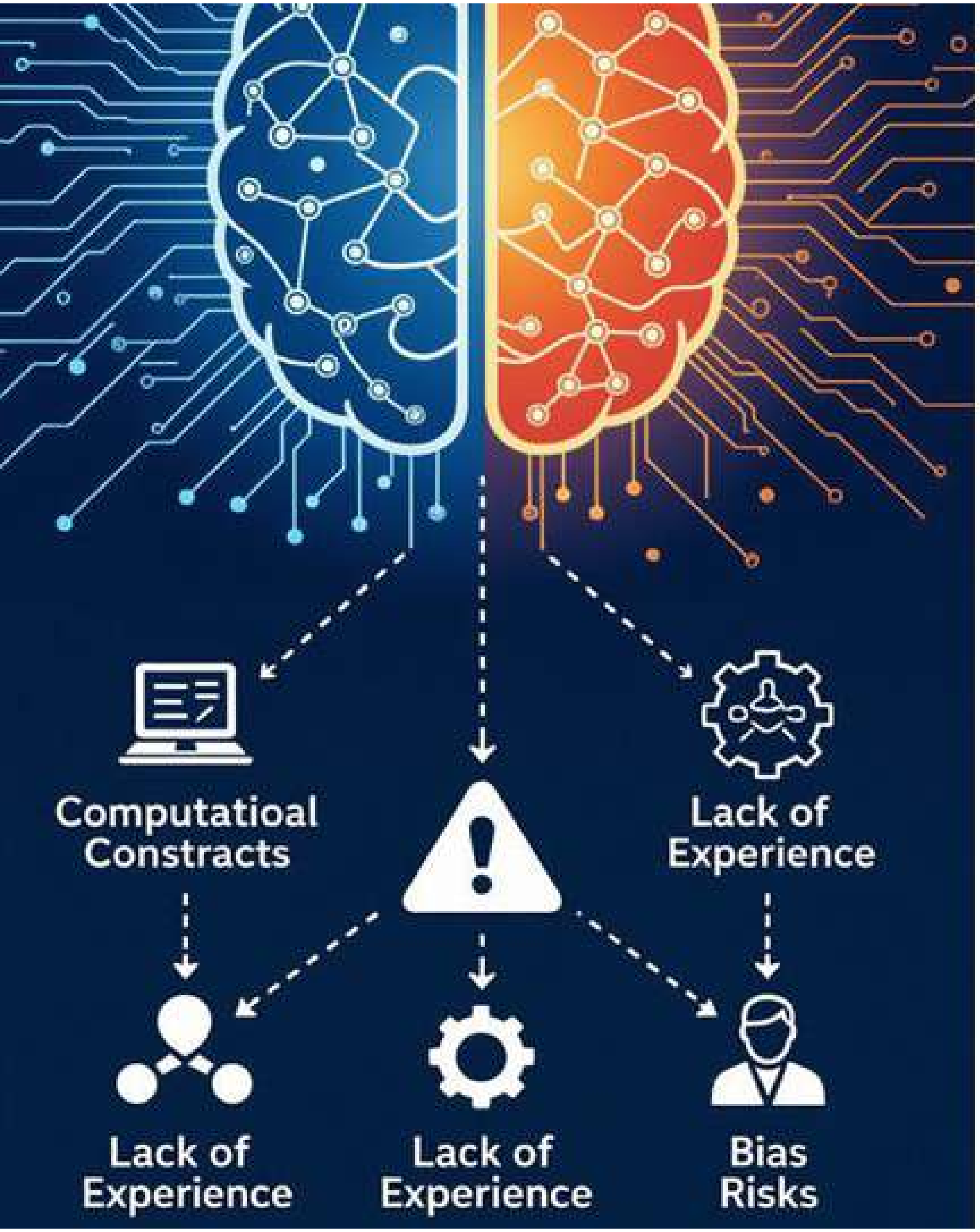
Introduction to Agentic AI

Agentic AI represents a significant advancement in AI capabilities:

- It enables AI systems to operate autonomously, setting and achieving goals without constant human intervention.
- Agentic AI requires agents to reason, plan, learn, and adapt to changing environments.

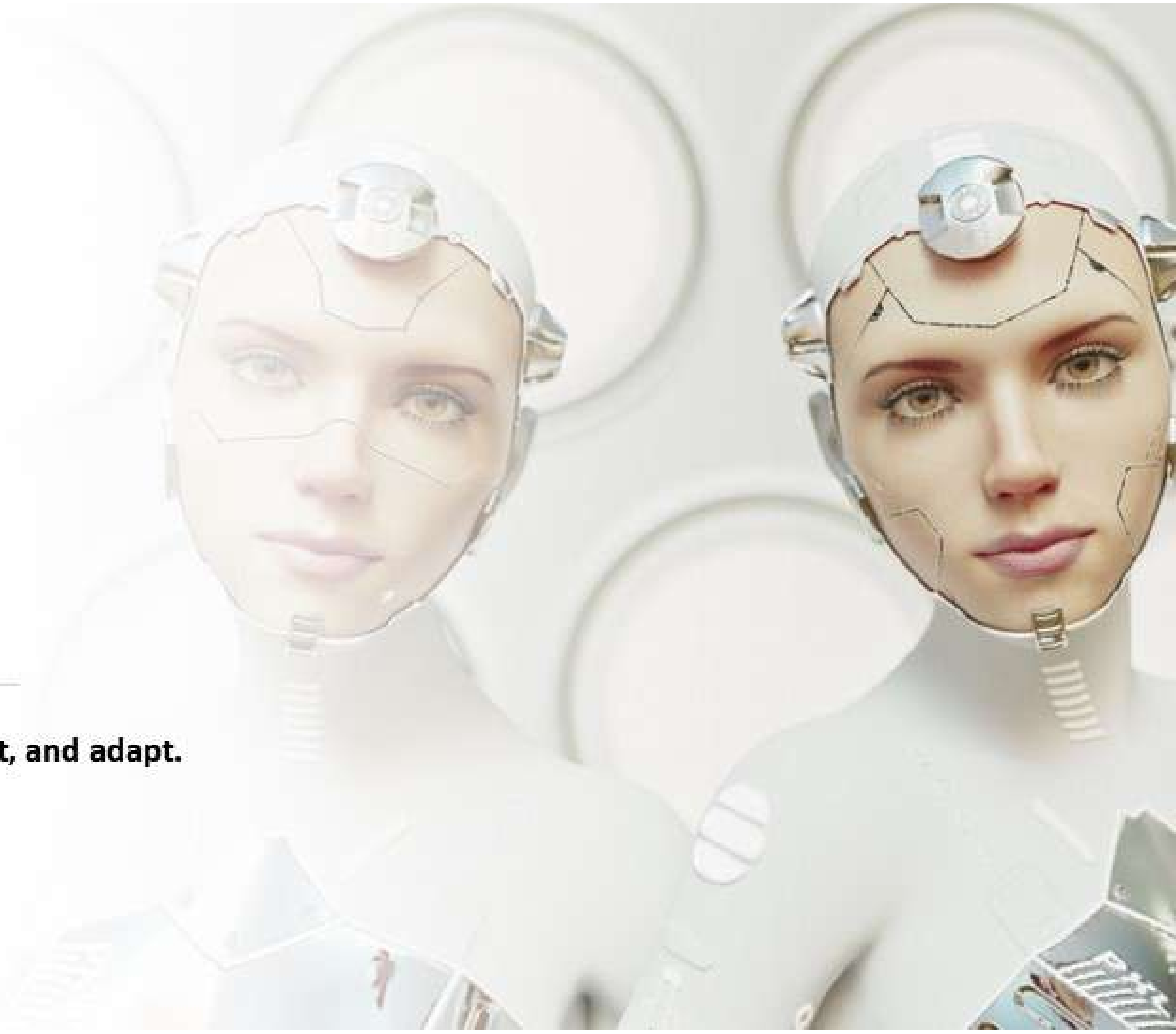
Agentic AI is about creating systems that can think and act for themselves.

Agent
Reason
Plan
Learn
Learn
Adapt



LLMs as Reasoning Agents

- ✓ LLMs Role: Large Language Models (LLMs) can function as reasoning agents by leveraging their vast knowledge and language processing capabilities:
  - ✓ They can understand complex instructions, generate plans, and execute tasks by interacting with various tools and APIs.
- ✓ Limitations: However, LLMs may face limitations in real-time decision-making due to:
  - ✓ Computational constraints
  - ✓ Lack of real-world experience
  - ✓ Potential for generating inaccurate or biased outputs

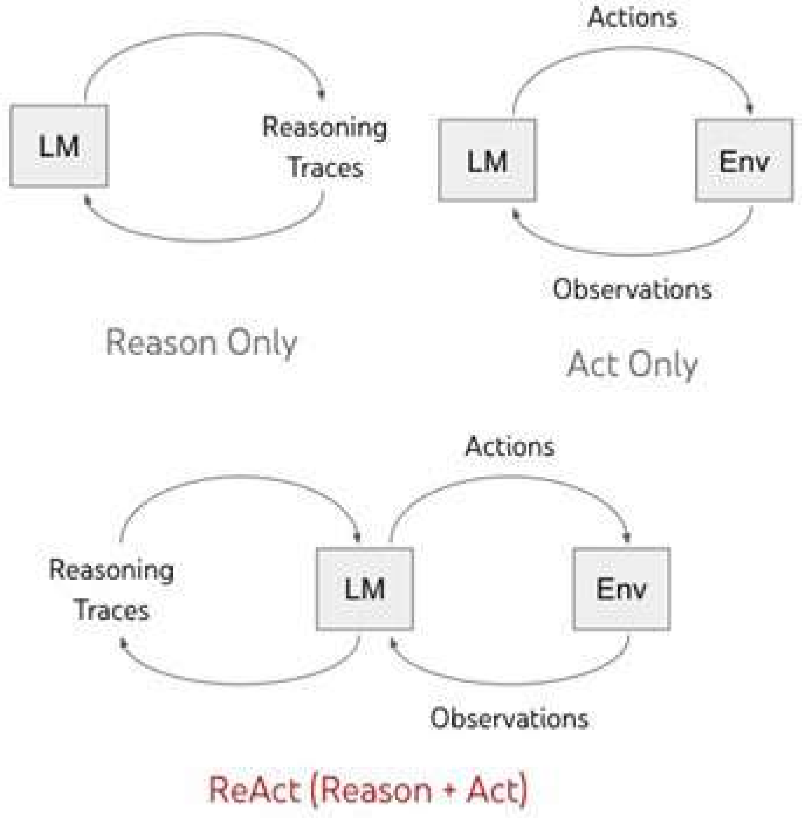


Agentic AI Reasoning Frameworks

Frameworks guide how agents think, act, and adapt.

ReAct (Reason + Act)

- ➢Combines reasoning and acting in cycles.
- ➢Agent: Think → Do → Reflect.
- ➢Example: Solving a puzzle by planning a step, executing, then adjusting.
- ➢Strength: Flexible, adapts during execution.



ReAct Example

Process: Reason → Act → Reflect

- **Reason**: "I need population numbers of Japan and Pakistan."
- **Act**: Search web for "Japan population 2025."
- **Reflect**: Found: ~125 million. Now search Pakistan population.
- **Act**: Search web for "Pakistan population 2025."
- **Reflect**: Found: ~240 million. Compare → Japan < Pakistan.

Final Answer:

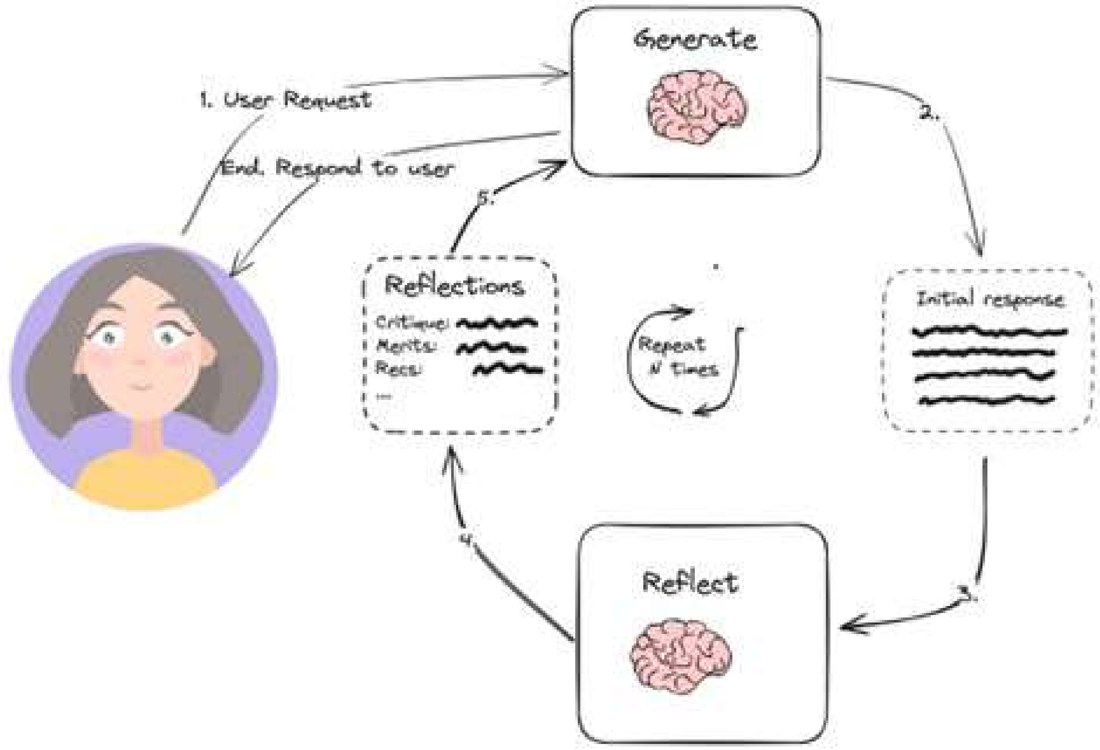
- Japan ~125M, Pakistan ~240M → Japan's population is smaller.

Strength: Shows reasoning + action + correction if data missing.

Reflexion

Process: Act + Self-Feedback

- Agents generate self-feedback after each action.
- Keeps a "memory" of what went wrong or right.
- Learns from mistakes autonomously.



Reflexion Example

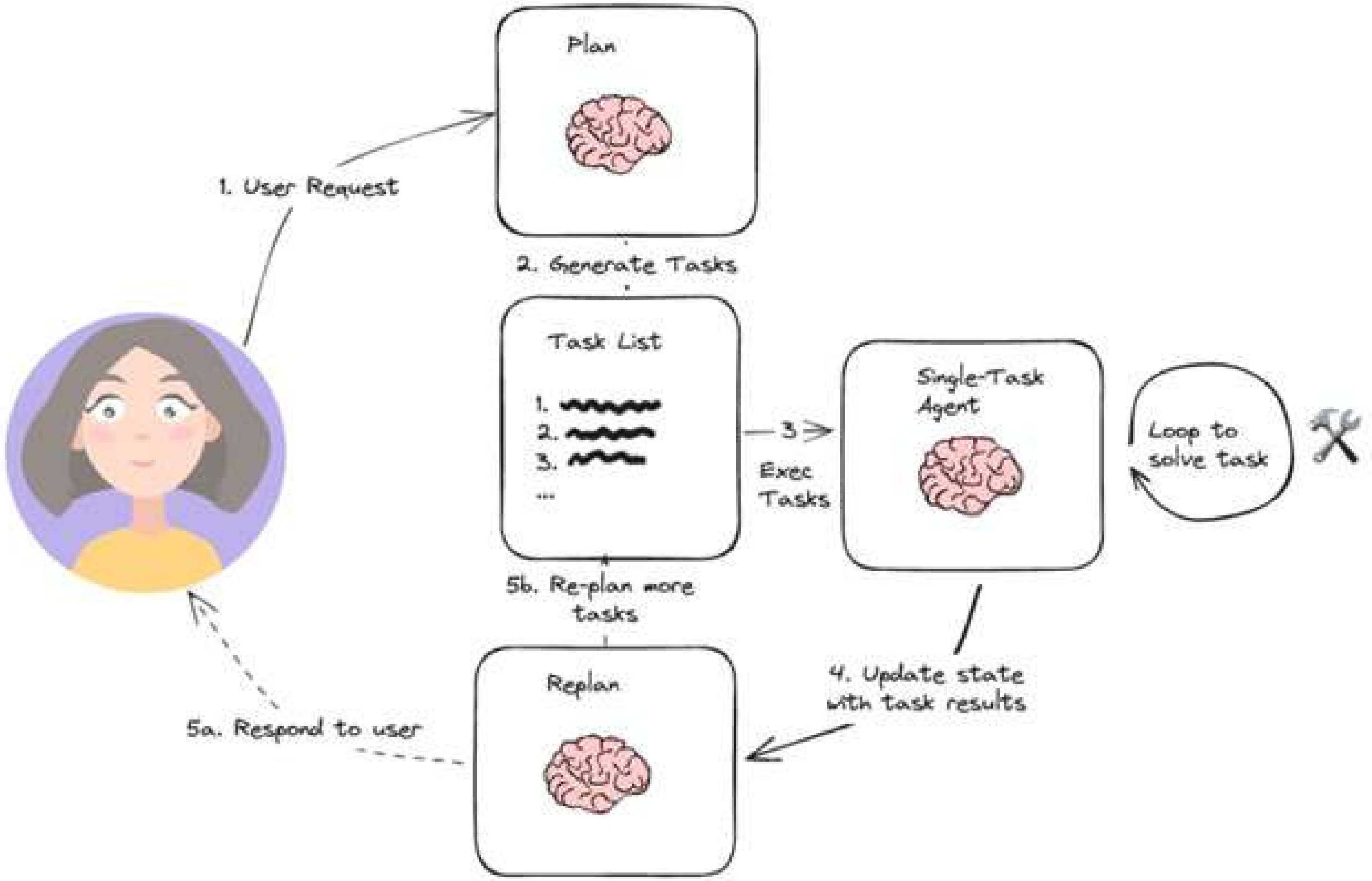
- Agent searches “Japan population 2025.”
- Gets 125M. Writes feedback: “ Data looks correct.”
- Agent searches “Pakistan population 2025.”
- Accidentally finds old 2010 value.
- Self-feedback: “This seems outdated, try again.”
- Corrects itself, finds 240M.

Final Answer:

- Japan smaller.
  Strength: Learns from mistakes and retries.

Plan-and-Execute (PnE)

- Two phases: Plan first, then Execute.
- Suitable for long, structured tasks.
- Example: Writing a research report → outline → draft → final.
- **Strength**: Organized execution.
- **Weakness**: Rigid if environment changes.



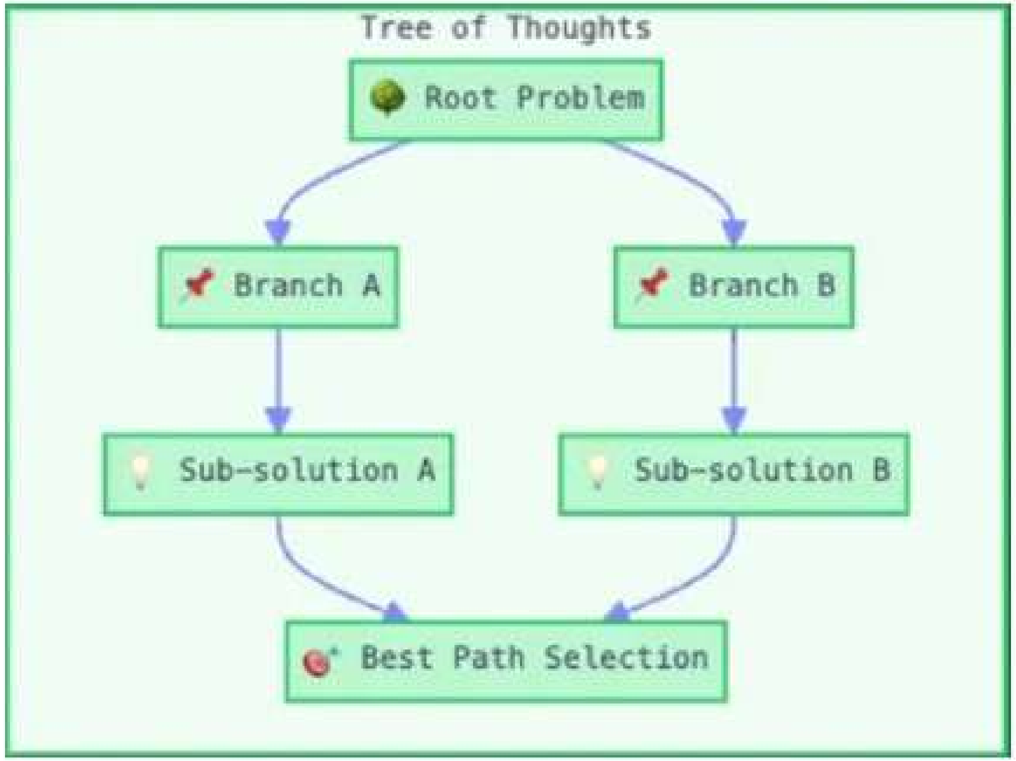
Plan-and-Execute Example

Process: Plan first → Execute steps

- **Plan:**
  - Find Japan population.
  - Find Pakistan population.
  - Compare values.
- **Execute:** Runs each step in order.
- **Final Answer:** Japan smaller.
- **Strength**: Clear structure, but less flexible if new issues appear.

Tree-of-Thought (ToT)

- Agent explores multiple reasoning paths (like branches of a tree).
- Evaluates options before committing to an action.
- Example: Chess move planning → analyze different strategies.
- Strength: Better decision-making.



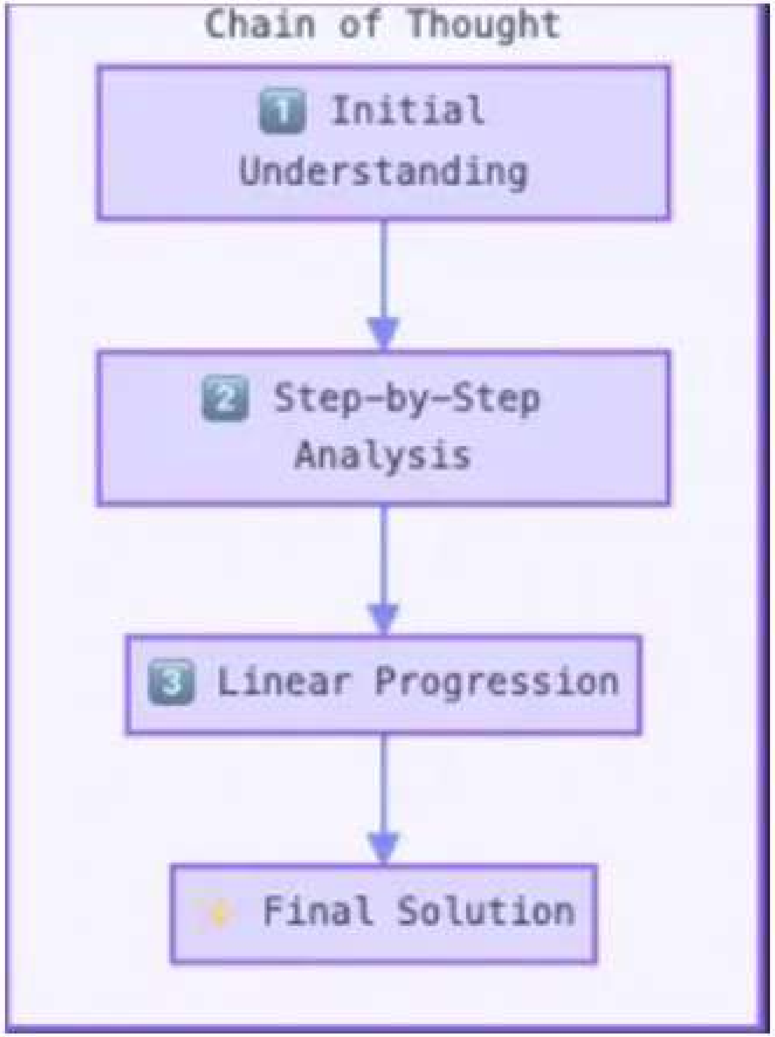
Tree-of-Thought Example

Process: Explore multiple reasoning paths

- **Path A:** "Check UN data → compare values."
- **Path B:** "Check World Bank data → compare values."
- **Path C:** "Check Wikipedia → compare values."
- Agent evaluates which is most reliable (UN/World Bank).
- **Final Answer:** Uses best source for comparison.
- **Strength**: Considers alternatives, better reliability.

Chain-of-Thought (CoT)

- Agent thinks step by step, showing reasoning explicitly.
- Example: Solving a math problem with scratch work.
- Strength: Transparent, improves accuracy.
- Often used as a building block in other frameworks.



Chain-of-Thought (CoT) Example

Chain-of-Thought :

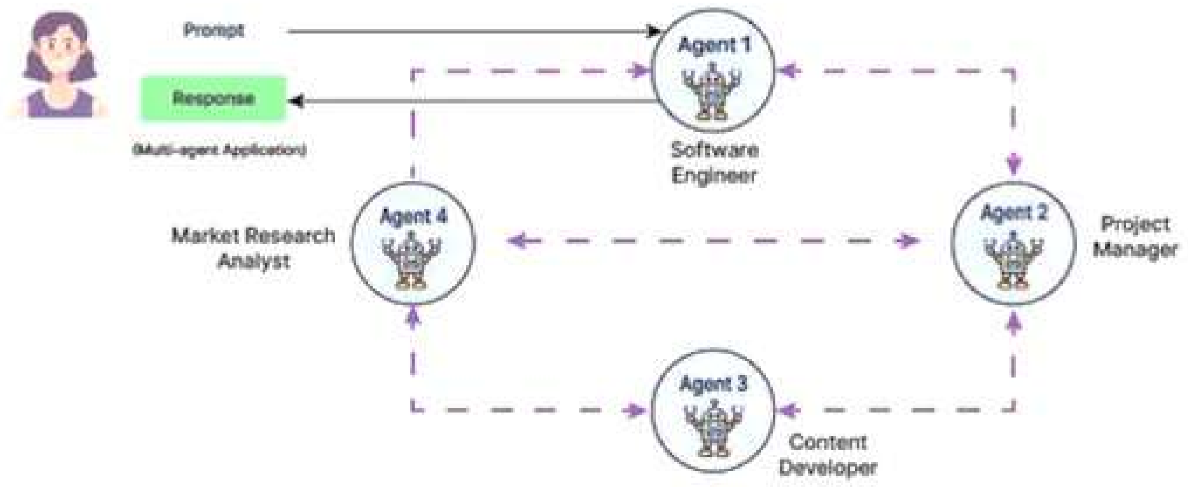
1. Reasoning: “37 × 46 = (37 × 40) + (37 × 6).”
2. Calculation: “37 × 40 = 1480, 37 × 6 = 222.”
3. Add results: “1480 + 222 = 1702.”

Final Answer: 1702

Strength: Shows transparent step-by-step reasoning.

Multi-Agent Systems

- Multiple agents with different roles (planner, critic, executor).
- Agents can collaborate, debate, or specialize.Mirrors teamwork in humans.
- Strength: More robust and creative.



Example: Multi-Agent Systems

Task: *"Write a 200-word summary of climate change's economic impacts."*

- **Normal Prompting:**

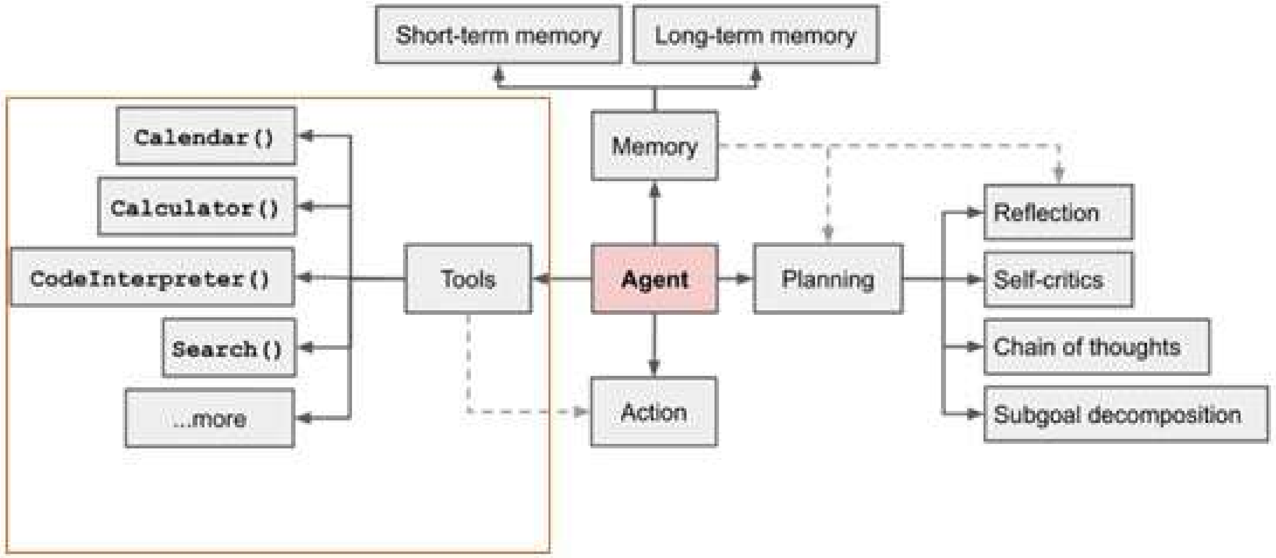
Agent tries to do everything in one shot.

- **Multi-Agent Setup:**
  - **Agent A (Researcher):** Finds key facts from reports.
  - **Agent B (Writer):** Converts facts into structured text.
  - **Agent C (Reviewer):** Checks clarity, grammar, and accuracy.
  - Agents collaborate, passing outputs between them.

Final Answer: A polished 200-word summary with checked facts.
Strength: Division of roles = higher quality + collaboration.

LLM + Tools (LangChain Agents)

- Agent can call external tools: search engines, APIs, calculators.
- Extends beyond language into action in the real world.
- Strength: Practical, powerful in real applications.



LLMs + Tools (LangChain Agent)

Task: *"What was the closing price of Tesla stock yesterday?"*

- **Normal Prompting:**

Agent replies: "Sorry, I don't have real-time data."

- **LangChain Agent with Tools:**
    1. Reasoning: "Need to fetch Tesla stock data."
    2. Action: Call API tool for stock prices.
    3. Result: "Closing price: $235.71."
    4. Reflection: Present the answer clearly.

Final Answer: Tesla closed at $235.71 yesterday.

Strength: Goes beyond text — uses **external tools/APIs** to give real answers.

Framework	Strengths	Best Use Case
ReAct	Flexible, adaptive	Dynamic problem solving
Reflexion	Self-correction	Iterative tasks
PnE	Structured	Long-term projects
ToT	Explores options	Strategy, planning
CoT	Clear reasoning	Math, logic problems
Multi-Agent	Collaboration	Complex goals
LLM + Tools	Resourceful	Real-world workflows

Agentic AI Development / Engineering Frameworks

Software toolkits to build agent systems with memory, tools, multi-agent workflows

Development Frameworks

- LangChain – Agent chaining, RAG, memory.
- AutoGen (Microsoft) – Multi-agent collaboration.
- CrewAI – Role-based agent teamwork.
- LlamaIndex – Knowledge retrieval.
- BabyAGI – Lightweight autonomous tasking.
- SuperAGI – Scalable multi-agent automation.
- Voyager – Self-learning autonomous agents.

Analogy to Understand the Difference

- Reasoning Frameworks = The **thinking style** of an agent (like problem-solving strategies).
- Development Frameworks = The **software environment** that equips agents with tools, memory, and actions.

Example:

- ReAct tells the agent HOW to think.
- LangChain provides WHERE to implement that thinking with APIs and workflows.