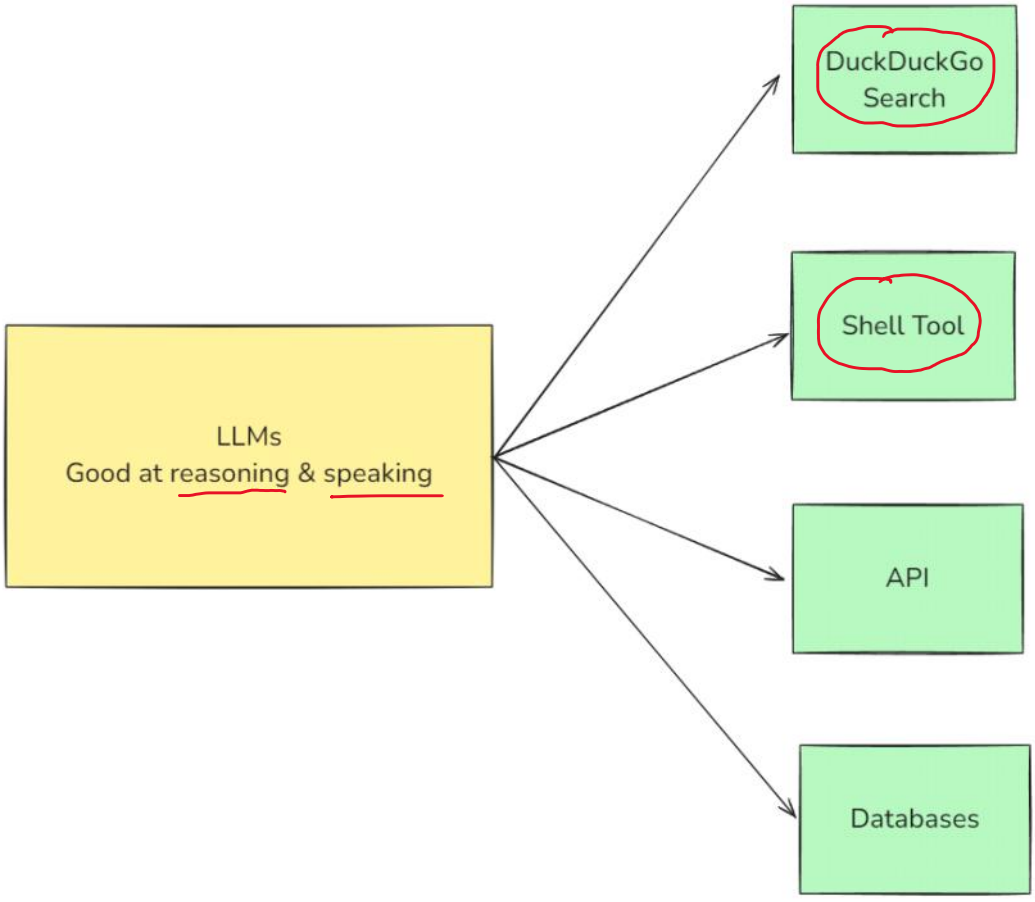
Quick Revision

25 April 2025 16:18

TOOLS

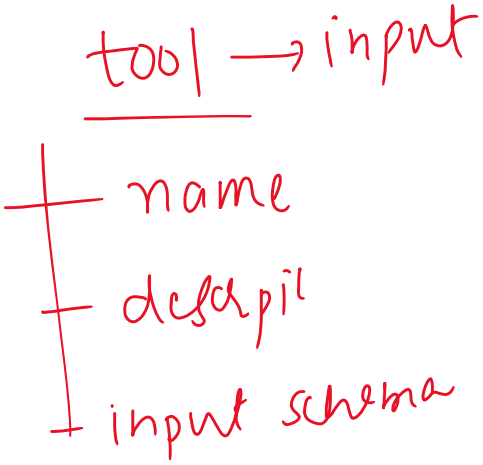


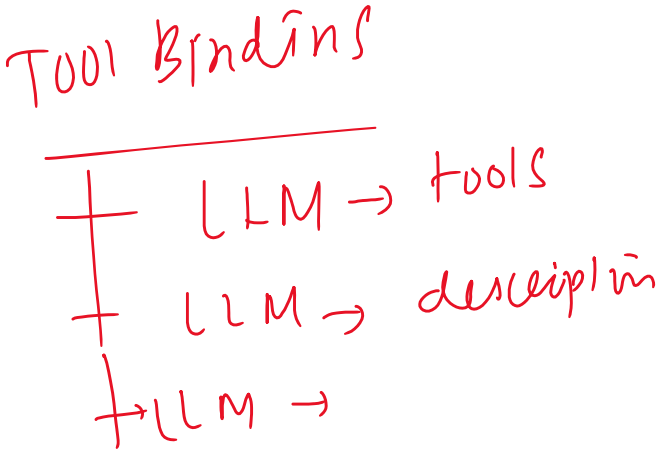
Tool Binding

25 April 2025 16:45

Tool Binding is the step where you **register tools with a Language Model (LLM)** so that:

1. The LLM knows **what tools are available**
2. It knows **what each tool does** (via description)
3. It knows **what input format to use** (via schema)





Tool Calling

25 April 2025 16:54

Tool Calling is the process where the **LLM (language model)** decides, during a conversation or task, that it needs to **use a specific tool (function)** — and generates a structured output with:

- the **name of the tool**
- and the **arguments** to call it with

🚨 The LLM **does not actually run the tool** — it just **suggests** the tool and the input arguments. The **actual execution is handled by LangChain or you**.

> "What's 8 multiplied by 7?"

The LLM responds with a **tool call**:

```json
{
  "tool": "multiply",
  "args": { "a": 8, "b": 7 }
}
```

Tool creation
Tool bindings

name : multiply
schema

Tool Execution

25 April 2025 17:12

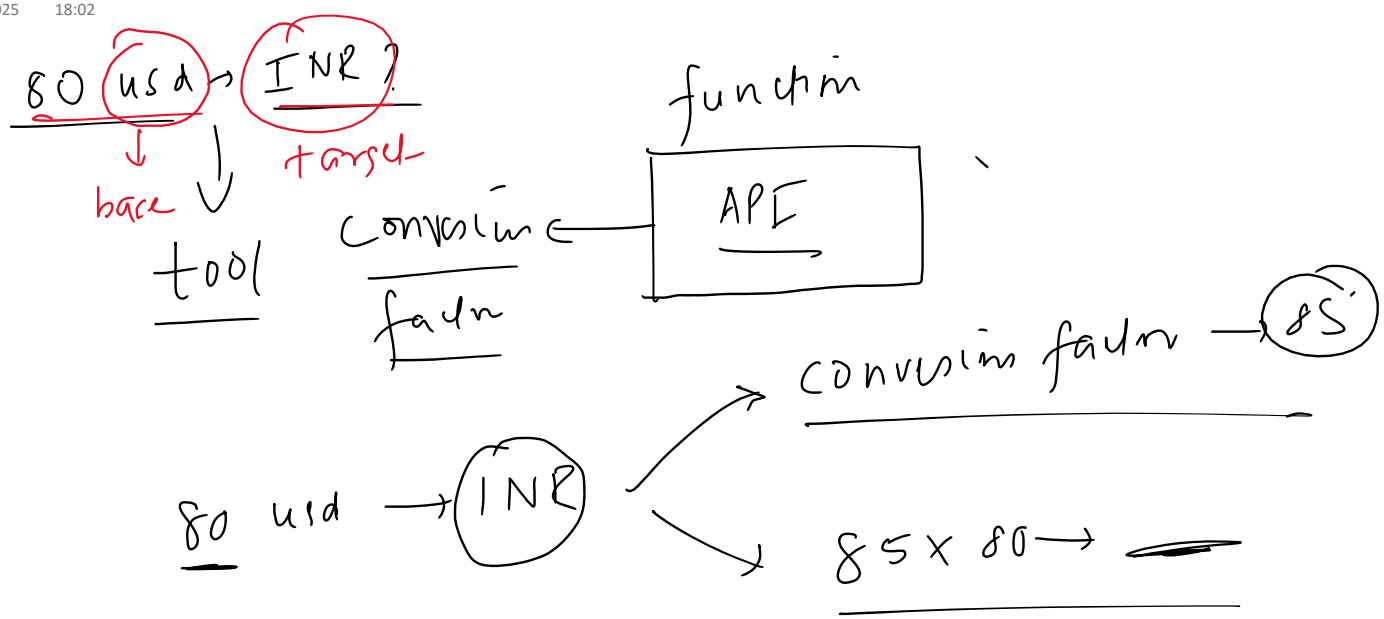
Tool Execution is the step where the **actual Python function (tool)** is run using the input arguments that the **LLM suggested during tool calling**.

In simpler words:

> 🧠 The LLM says:
> *"Hey, call the `multiply` tool with a=8 and b=7."*
>
> ⚙️ **Tool Execution** is when *you or LangChain actually run:*
> `multiply(a=8, b=7)`
> → and get the result: `56`

Currency Conversion Tool

25 April 2025 18:02



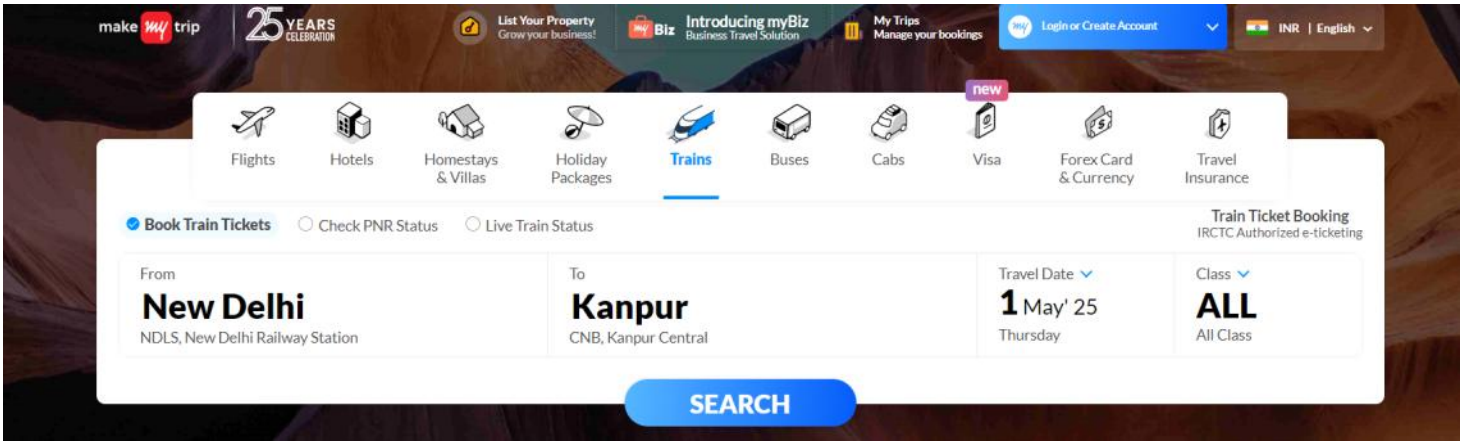
> ❌ "LLM, do **not** try to fill this argument." ✅ "I (the developer/runtime) will inject this value after running earlier tools."

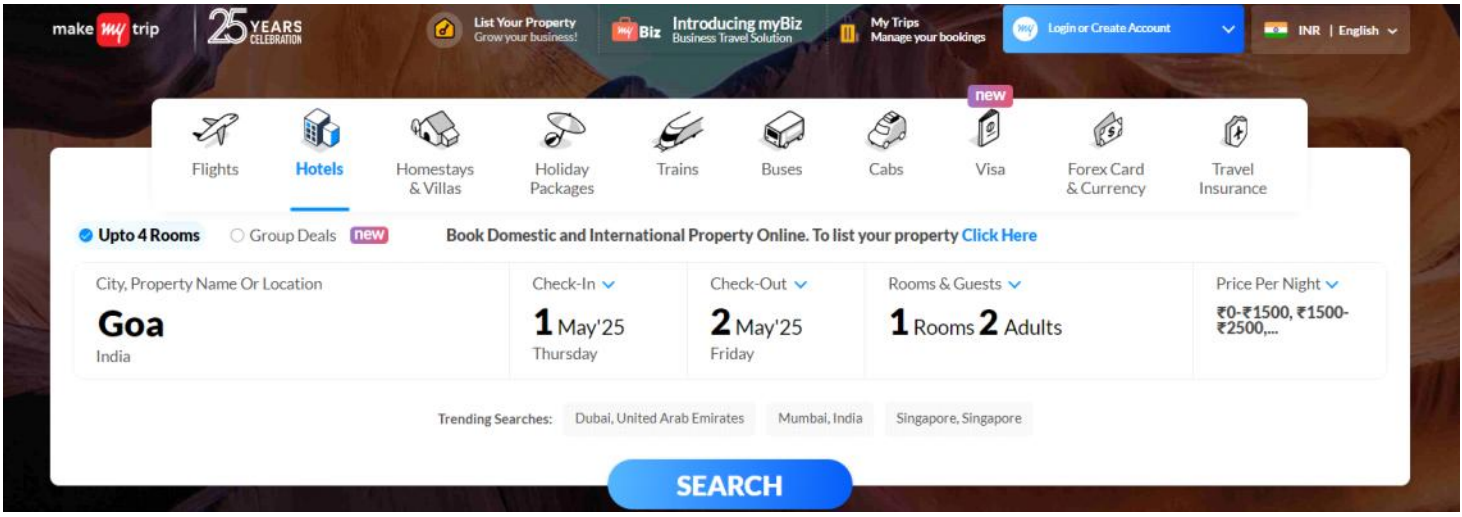
AI Agent → tools / tool calling (X)

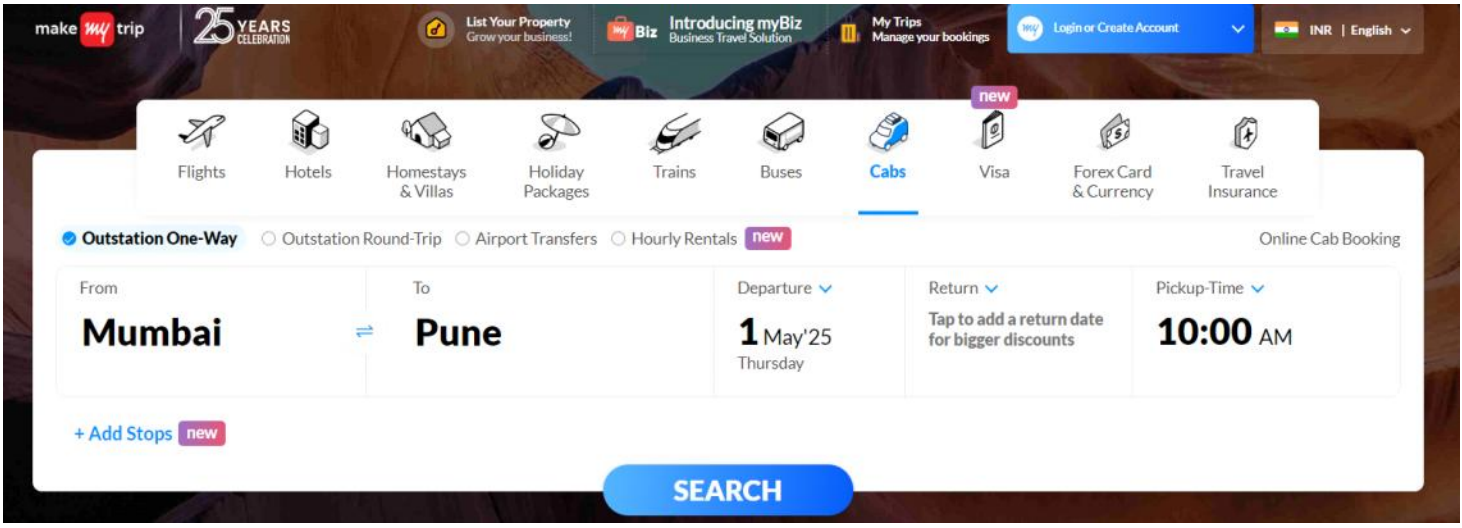
1. **User says**: "Convert 10 USD to INR."
2. **LLM thinks**: "I don't know the rate. First, let me call `get_conversion_factor`."
3. **Tool result comes**: 85.3415
4. **LLM looks at result, THINKS again**: "Now I know the rate, next I should call `convert` with 10 and 85.3415."
5. **Tool result comes**: 853.415 INR
6. **LLM summarizes**: "10 USD is 853.415 INR at current rate."

What are AI Agents

29 April 2025 22:53







✅ **User enters query:**

> "Can you create a budget travel itinerary from Delhi to Goa from 1st to 7th May?"

🧠 1. Understanding your intent

The agent internally interprets:

- ✅ Origin: Delhi
- ✅ Destination: Goa
- ✅ Dates: May 1–7
- ✅ Preference: Budget travel
- ✅ Goal: Plan complete itinerary + optimize cost

Agent thinks: *"I need to plan affordable travel, stay, local movement, and activities over 7 days."*

🔍 2. Research transport options

Agent (internally):

- Calls **IRCTC/train API** and **Flight APIs**
- Compares price, duration, and availability

Agent (to user):

> "The cheapest and most available option is the Goa Express from Delhi on April 30th night (sleeper class ₹800 or 3AC ₹1500).
> It arrives in Goa on May 1st evening. Would you like to go ahead?"

✅ User selects "3AC".

🏨 3. Search for affordable stay

Agent (internally):

- Uses hotel API or scraping tools
- Filters by:
  - Price < ₹700/night
  - Close to Baga or Anjuna beach
  - User reviews

Agent (to user):

> "I found a dorm room at The Hosteller, Baga for ₹650/night with good ratings. Shall I book this for 6 nights (May 1–7)?"

✅ User agrees.

🛵 4. **Plan local travel**

Agent (to user):

> "Scooter rentals are the most cost-effective in Goa.
> Should I pre-book a scooty for ₹300/day? (~₹1800 for 6 days)"

✅ User says yes.

🎯 5. **Generate activity plan**

Agent (internally):

- Calls sightseeing API / uses static local dataset
- Maps popular attractions across North & South Goa
- Builds a balanced plan:
  - Culture, relaxation, nightlife, shopping

Agent (to user):

> "Here's a draft plan:

- **May 2**: Baga, Calangute, Aguada Fort
- **May 3**: Old Goa churches, Fontainhas, local lunch
- **May 4**: South Goa day trip: Palolem, Cabo de Rama
- **May 5**: Anjuna flea market + sunset at Chapora
- **May 6**: Free day + beach cafe dinner"

✅ User loves it.

🛬 6. **Plan return**

Agent (to user):

> "Return options on May 7:

1. Train (₹800–₹1500, overnight)
2. Early morning flight (₹2800, 6:00 AM via SpiceJet)

Choose your preference."

✅ User picks train.

🧾 7. **Budget summary**

Agent (to user):

Item	Cost
Train (round trip)	₹3,000
Stay (6 nights)	₹3,900
Scooter + fuel	₹2,200
Food (local + budget)	₹2,500
Sightseeing & entry	₹1,000
Shopping/misc	₹1,500
Total	**~₹14,100**

> "Total trip cost: ₹14,100. Would you like to finalize the bookings?"

✅ User confirms.

📩 8. **Final Output**

Agent (to user):

> ✅ Bookings confirmed.
> 🧾 Receipt sent to your email.
> 🗓️ Calendar invite added for train departures.
> 📱 You'll get daily reminders and local recommendations during your stay.
> 💬 Need help anytime? Just ask!

> **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.**

Goal-driven	You tell the agent *what you want*, not *how to do it*
Autonomous planning	Agent breaks down the problem and sequences tasks on its own
Tool-using	Agent calls APIs, calculators, search tools, etc.
Context-aware	Maintains memory across steps to inform future actions
Reasoning-capable	Makes decisions dynamically (e.g., "what to do next")
Adaptive	Rethinks plan when things change (e.g., API fails, no data)

Building an agent in LangChain

29 April 2025 22:53

Explanation

29 April 2025 22:53

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)

REACT: SYNERGIZING REASONING AND ACTING IN LANGUAGE MODELS

Shunyu Yao*,1, Jeffrey Zhao[2], Dian Yu[2], Nan Du[2], Izhak Shafran[2], Karthik Narasimhan[1], Yuan Cao[2]

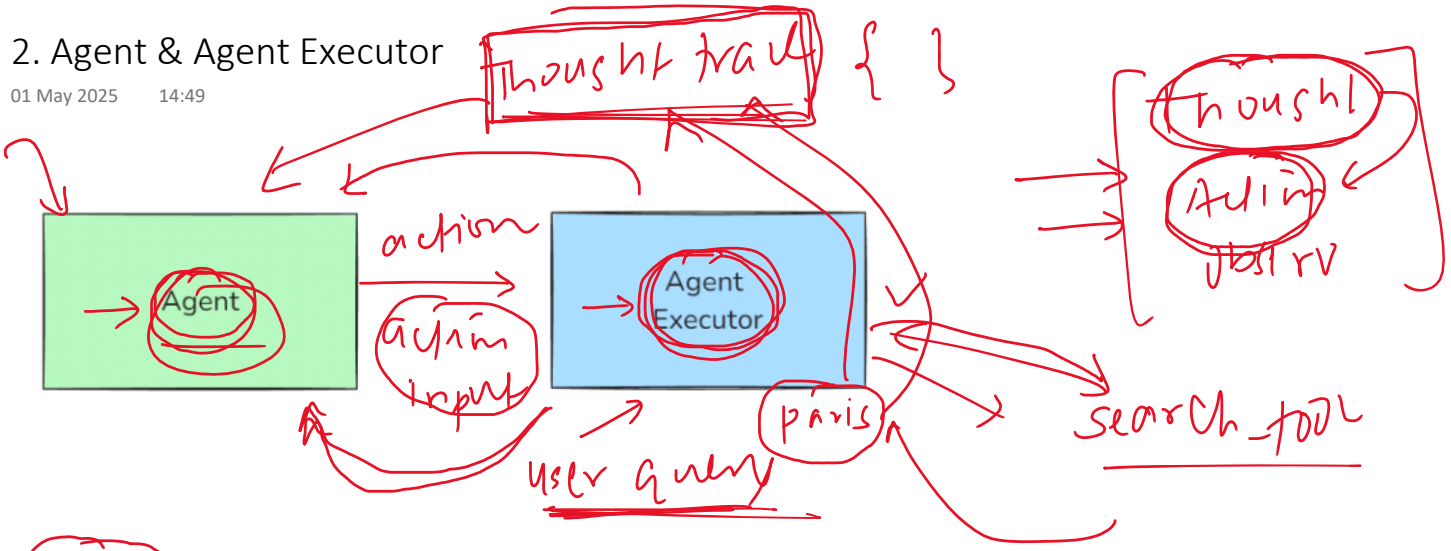
[1]Department of Computer Science, Princeton University
[2]Google Research, Brain team
[1]{shunyuy,karthikn}@princeton.edu
[2]{jeffreyzhao,dianyu,dunan,izhak,yuancao}@google.com

ABSTRACT

While large language models (LLMs) have demonstrated impressive performance across tasks in language understanding and interactive decision making, their abilities for reasoning (e.g. chain-of-thought prompting) and acting (e.g. action plan generation) have primarily been studied as separate topics. In this paper, we explore the use of LLMs to generate both reasoning traces and task-specific actions in an interleaved manner, allowing for greater synergy between the two: reasoning traces help the model induce, track, and update action plans as well as handle exceptions, while actions allow it to interface with and gather additional information from external sources such as knowledge bases or environments. We apply our approach, named ReAct, to a diverse set of language and decision making tasks and demonstrate its effectiveness over state-of-the-art baselines in addition to improved human interpretability and trustworthiness. Concretely, on question answering (HotpotQA) and fact verification (Fever), ReAct overcomes prevalent issues of hallucination and error propagation in chain-of-thought reasoning by interacting with a simple Wikipedia API, and generating human-like task-solving trajectories that are more interpretable than baselines without reasoning traces. Furthermore, on two interactive decision making benchmarks (ALFWorld and WebShop), ReAct outperforms imitation and reinforcement learning methods by an absolute success rate of 34% and 10% respectively, while being prompted with only one or two in-context examples.

2. Agent & Agent Executor

01 May 2025 14:49



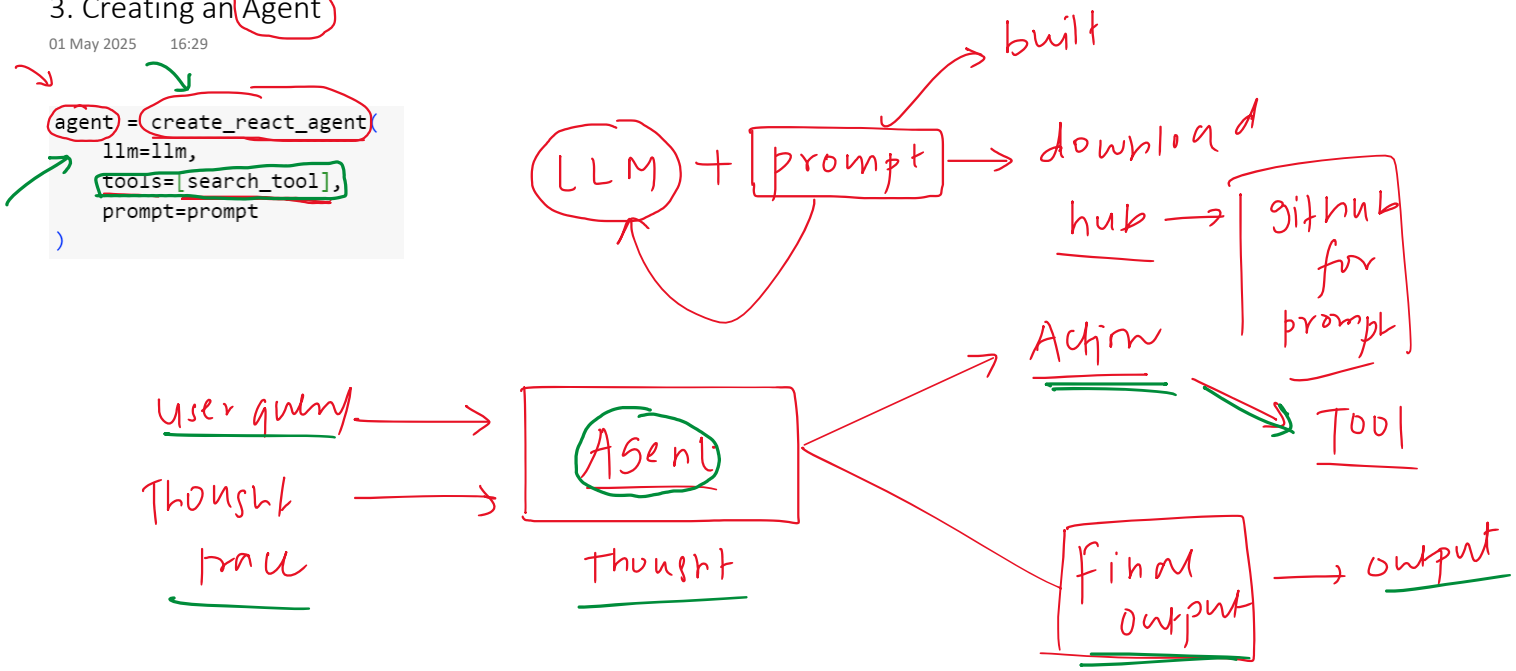
`AgentExecutor` orchestrates the **entire loop**:

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

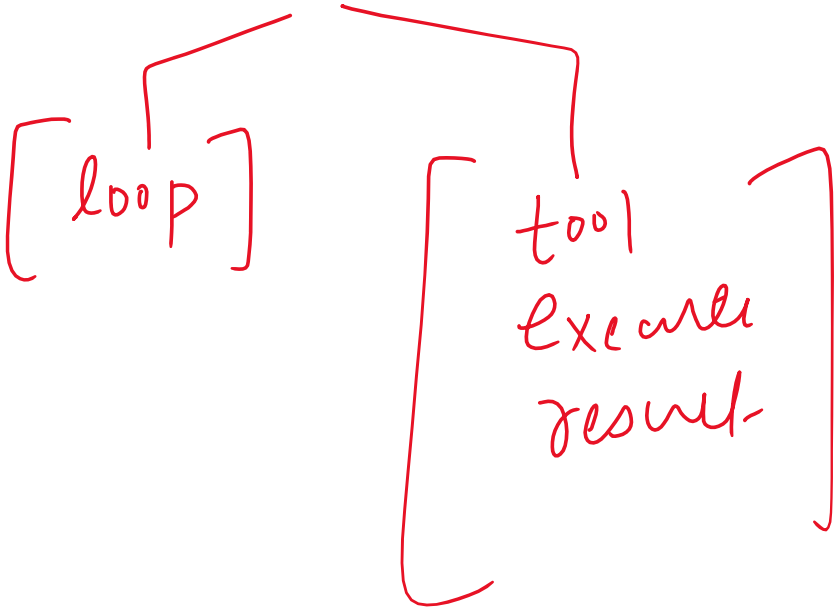
```
agent = create_react_agent(
    llm=llm,
    tools=[search_tool],
    prompt=prompt
)
```



4. Creating an Agent Executor

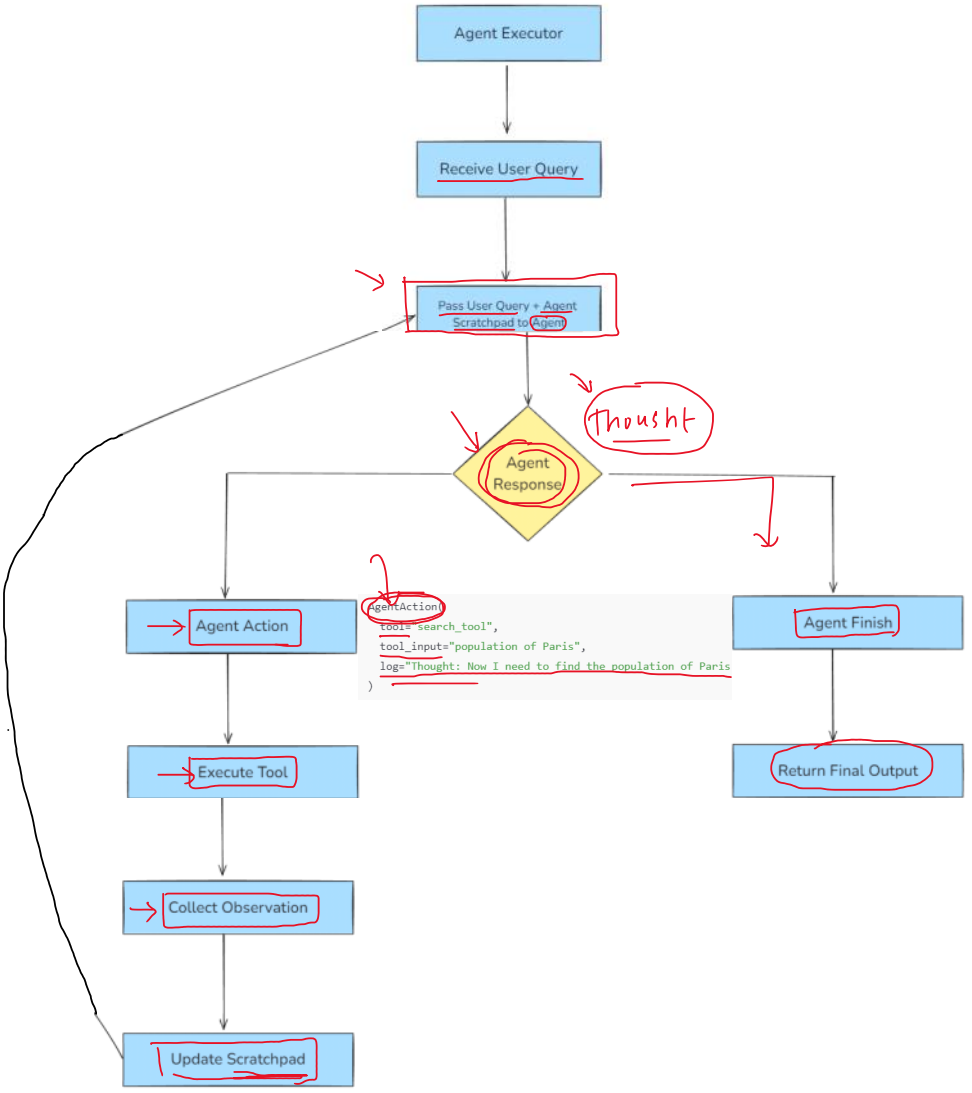
01 May 2025 16:30

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



5. Flow Chart

01 May 2025 16:30



6. Example

01 May 2025 17:23

🧠 Input Query:

> "What is the capital of France and what is its population?"

```
Answer the following questions as best you can. You have access to the following tools:

search_tool: Useful for answering general knowledge questions by querying a search API.

Use the following format:

Question: the input question you must answer
Thought: you should always think about what to do
Action: the action to take, should be one of [search_tool]
Action Input: the input to the action
Observation: the result of the action
... (this Thought/Action/Action Input/Observation can repeat N times)
Thought: I now know the final answer
Final Answer: the final answer to the original input question

Begin!

Question: What is the capital of France and what is its population?
Thought:
```

```
Thought: I need to find the capital of France first.
Action: search_tool
Action Input: "capital of France"
```

```
AgentAction(
  tool="search_tool",
  tool_input="capital of France",
  log="Thought: I need to find the capital of France first.
)
```

```
observation = search_tool("capital of France")
```

→ `"Paris is the capital of France."`

```
Thought: I need to find the capital of France first.
Action: search_tool
Action Input: "capital of France"
Observation: Paris is the capital of France.
```

📌 Current state so far:

- **User Input**:

`What is the capital of France and what is its population?`

- **Agent Scratchpad after Step 2**:

text

```
Thought: I need to find the capital of France first.
Action: search_tool
Action Input: "capital of France"
Observation: Paris is the capital of France.
```

```
Answer the following questions as best you can. You have access to the following tools

search_tool: Useful for answering general knowledge questions by querying a search API.

Use the following format:

Question: the input question you must answer
Thought: you should always think about what to do
Action: the action to take, should be one of [search_tool]
Action Input: the input to the action
Observation: the result of the action
... (this Thought/Action/Action Input/Observation can repeat N times)
Thought: I now know the final answer
Final Answer: the final answer to the original input question

Begin!

Question: What is the capital of France and what is its population?
Thought: I need to find the capital of France first.
Action: search_tool
Action Input: "capital of France"
Observation: Paris is the capital of France.
Thought:
```

```
Thought: Now I need to find the population of Paris.
Action: search_tool
Action Input: "population of Paris"
```

```
AgentAction(
  tool="search_tool",
  tool_input="population of Paris",
```

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

```
observation = search_tool("population of Paris")
```

```
"Paris has a population of approximately 2.1 million."
```

```
Thought: I need to find the capital of France first.
Action: search_tool
Action Input: "capital of France"
Observation: Paris is the capital of France.
Thought: Now I need to find the population of Paris.
Action: search_tool
Action Input: "population of Paris"
Observation: Paris has a population of approximately 2.1 million.
```

```
Answer the following questions as best you can. You have access to the following tools:

search_tool: Useful for answering general knowledge questions by querying a search API.

Use the following format:

Question: the input question you must answer
Thought: you should always think about what to do
Action: the action to take, should be one of [search_tool]
Action Input: the input to the action
Observation: the result of the action
... (this Thought/Action/Action Input/Observation can repeat N times)
Thought: I now know the final answer
Final Answer: the final answer to the original input question

Begin!

Question: What is the capital of France and what is its population?
Thought: I need to find the capital of France first.
Action: search_tool
Action Input: "capital of France"
Observation: Paris is the capital of France.
Thought: Now I need to find the population of Paris.
Action: search_tool
Action Input: "population of Paris"
Observation: Paris has a population of approximately 2.1 million.
Thought:
```

```
Thought: I now know the final answer.
Final Answer: Paris is the capital of France and has a population of approximately 2.1 million.
```

```
Thought: I now know the final answer.
Final Answer: Paris is the capital of France and has a population of approximately 2.1 million.
```

```
AgentFinish(
  return_values={"output": "The capital of France is Paris..."},
  log="Thought: I now know the final answer.\nFinal Answer: ..."
)
```

```
{
  "output": "Paris is the capital of France and has a population of approximately 2.1 million."
}
```