Tools in LangChain

23 May 2025 23:07

What is a Tool?

23 April 2025 15:45

A tool is just a Python function (or API) that is packaged in a way the LLM can understand and call when needed.

LLMs (like GPT) are great at:

- Reasoning (think)
- Language generation (speak)

But they **can't do things like**:

- Access live data (weather, news)
- Do reliable math
- Call APIs
- Run code
- Interact with a database

Built-in Tools

23 April 2025 15:46

A built-in tool is a tool that LangChain already provides for you —it's pre-built, production-ready, and requires minimal or no setup.

You don't have to write the function logic yourself — you just import and use it.

`DuckDuckGoSearchRun`	Web search via DuckDuckGo
`WikipediaQueryRun`	Wikipedia summary
`PythonREPLTool`	Run raw Python code
`ShellTool`	Run shell commands
`RequestsGetTool`	Make HTTP GET requests
`GmailSendMessageTool`	Send emails via Gmail
`SlackSendMessageTool`	Post message to Slack
`SQLDatabaseQueryTool`	Run SQL queries

Custom Tools

23 April 2025 15:46

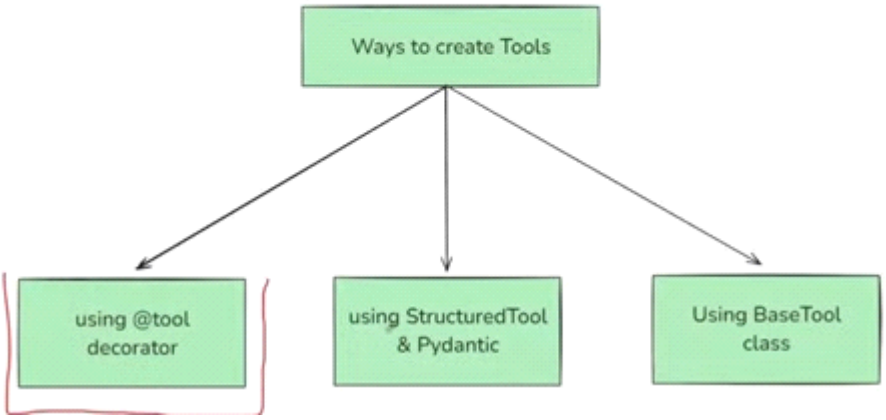
A custom tool is a tool that you define yourself.

Use them when:

- You want to call **your own APIs**
- You want to **encapsulate business logic**
- You want the LLM to interact with **your database, product, or app**

Ways to create Custom Tools

23 April 2025 15:46



A **Structured Tool** in LangChain is a special type of tool where the input to the tool follows a structured schema, typically defined using a Pydantic model.

BaseTool is the abstract base class for all tools in LangChain.
It defines the core structure and interface that any tool must follow, whether it's a simple one-liner or a fully customized function.

All other tool types like **@tool, StructuredTool** are built on top of BaseTool

Toolkits

23 April 2025 15:47

A toolkit is just a collection (bundle) of related tools that serve a common purpose — packaged together for convenience and **reusability**.

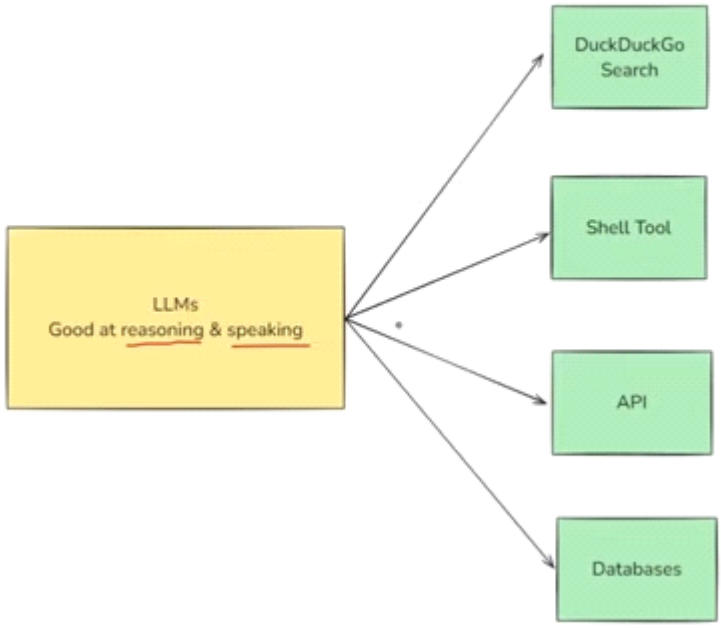
In LangChain:

- A toolkit might be: GoogleDriveToolKit
- And it can contain the following tools

- `GoogleDriveCreateFileTool` : Upload a file
- `GoogleDriveSearchTool` : Search for a file by name/content
- `GoogleDriveReadFileTool` : Read contents of a file

Quick Revision

25 April 2025 16:18



Tool Calling in LangChain

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 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`

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

AI Agent → tools / tool c.

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."