Introduction to Claude's Tool Use

- **academy.kspl.tech | Koenig AI Academy**

Prerequisites check

- **Before you start, verify that you can do three things:**
- **Read and write a small Python or TypeScript script.**
- **Store an API key in an environment variable rather than hard-coding it.**
- **Understand JSON objects, required fields, and string/number types.**
- **academy.kspl.tech | Koenig AI Academy**

The mental model: model chooses, host executes

- A tool-use exchange has four parts:
- You send Claude a user request plus a list of available tools.
- Claude returns either normal text or a tool_use block.
- Your host validates the tool name and input, executes local code, and returns a tool_result.
- academy.kspl.tech | Koenig AI Academy

A first implementation shape

- The exact SDK syntax can change, but the control flow is stable:
- "name": "get_stock_price",
- "description": "Return the latest known price for a ticker symbol from the demo portfolio feed.
- "input_schema": {
- academy.kspl.tech | Koenig AI Academy

Why input schemas are not optional

- **Without a schema, every tool call becomes a guess. The model may send stock, symbol, ticker_sym**
- **Schemas protect both sides:**
- **Claude gets a compact contract for which fields to fill.**
- **Your host can reject malformed input before touching external systems.**
- **academy.kspl.tech | Koenig AI Academy**

Common first failures

- The first failure is over-broad tools. A tool named run_python or query_database is easy to dem
- The second failure is hidden side effects. A tool named sync_customer might read from Salesforc
- The third failure is treating model output as trusted JSON. Even when using tool schemas, valid
- question: "In Claude tool use, who executes the tool code?",
- academy.kspl.tech | Koenig AI Academy

Try it next

- **Deploy a gateway with RBAC, rate limits, and JSONL auditing**
- **academy.kspl.tech | Koenig AI Academy**