Tool Calling & Agents

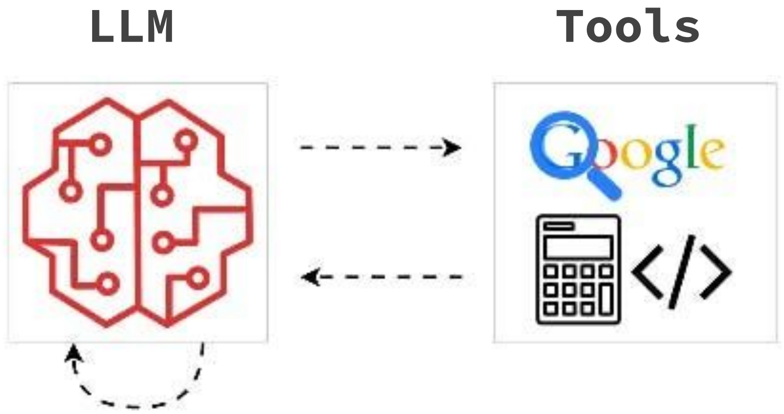
Understand what tool (function) calling is, how it works & building agents

What is Tool Calling?

Tool calling (aka function calling) enables LLMs to **interact with external code & services**

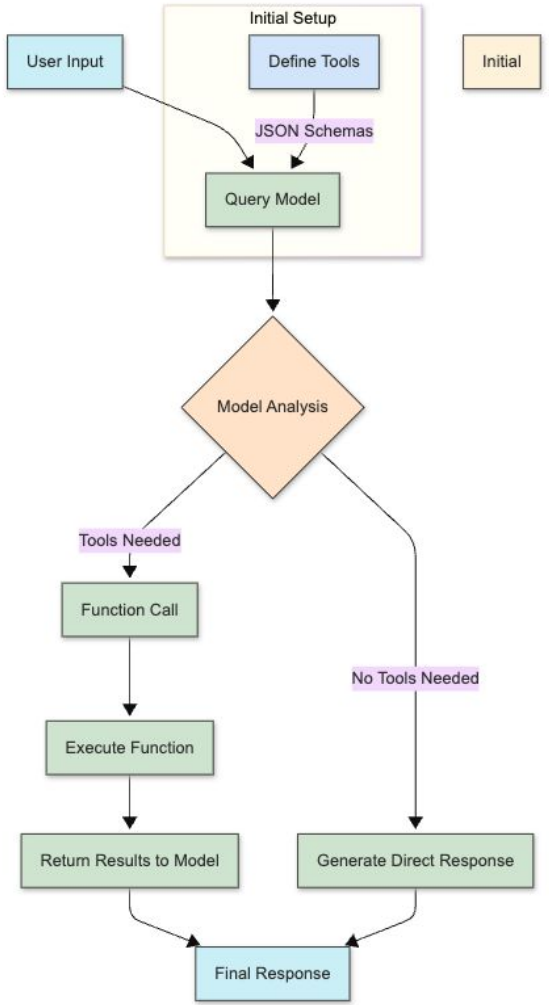
LLMs can **decide when to call functions** you define, providing structured arguments

Bridge between language understanding and actionable code execution



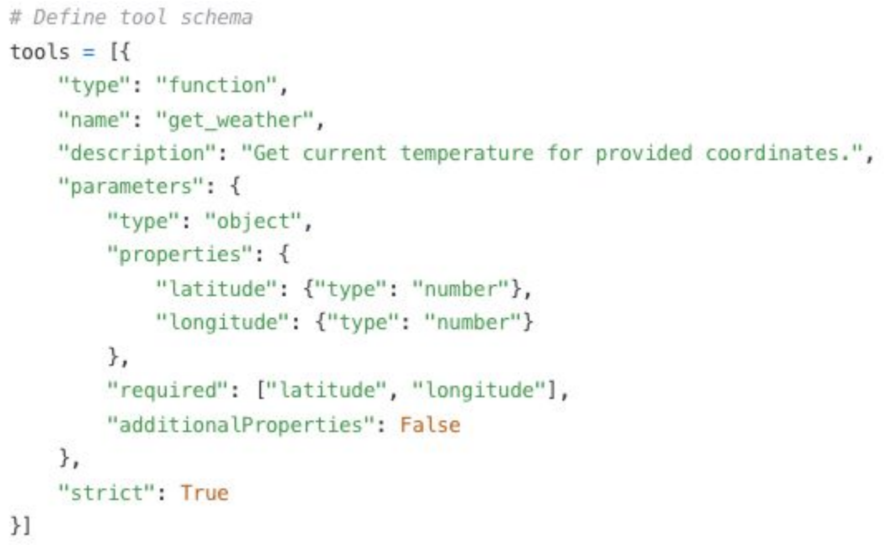
The Tool Calling Flow

1. **Define Tools:** Create JSON schemas for functions (get_weather, search_database, etc.)
2. **Query Model:** Send user input + tool definitions to the model
3. **Model Analysis:** Model determines if tools are needed to answer the query
4. **Function Call:** If needed, model returns function name + structured arguments
5. **Execute Function:** Your code runs the function with provided arguments
6. **Return Results:** Function output is sent back to the model
7. **Final Response:** Model incorporates function results into a coherent answer



The Tool Calling Flow

1. **Define Tools:** Create JSON schemas for functions (get_weather, search_database, etc.)
2. **Query Model:** Send user input + tool definitions to the model
3. **Model Analysis:** Model determines if tools are needed to answer the query
4. **Function Call:** If needed, model returns function name + structured arguments
5. **Execute Function:** Your code runs the function with provided arguments
6. **Return Results:** Function output is sent back to the model
7. **Final Response:** Model incorporates function results into a coherent answer

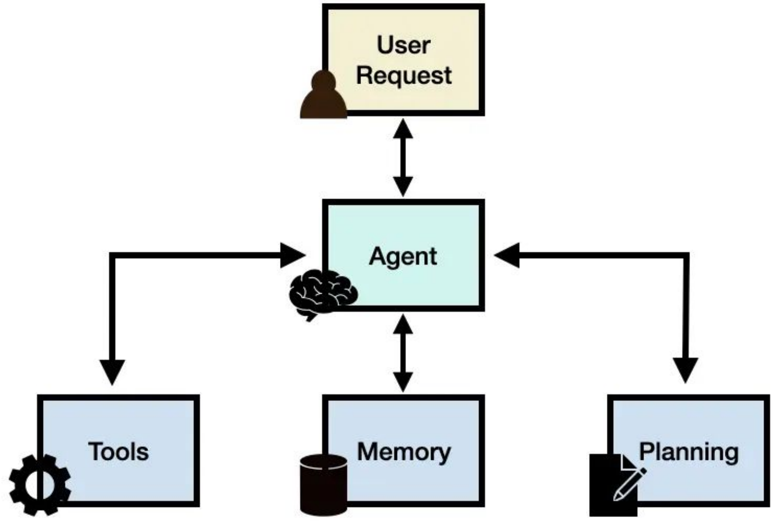


```python
# Define tool schema
tools = [{
    "type": "function",
    "name": "get_weather",
    "description": "Get current temperature for provided coordinates.",
    "parameters": {
        "type": "object",
        "properties": {
            "latitude": {"type": "number"},
            "longitude": {"type": "number"}
        },
        "required": ["latitude", "longitude"],
        "additionalProperties": False
    },
    "strict": True
}]
```

From Tool Calling to Agents - What is an Agent?

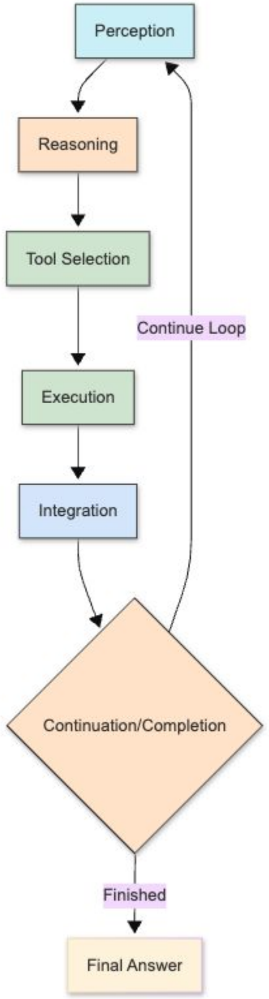
An agent is an autonomous system that uses LLMs to achieve goals through a cycle of:

- Reasoning about the current state
- Planning appropriate actions
- Executing those actions via tools
- Observing results & updating understanding



The Agentic Loop

1. **Perception:** Receive input (user prompt or environment data)
2. **Reasoning:** Determine what action to take (if any)
3. **Tool Selection:** Choose appropriate function(s) to call
4. **Execution:** Run selected function(s) with appropriate arguments
5. **Integration:** Process results and update knowledge state
6. **Continuation/Completion:** Either continue loop or provide final answer



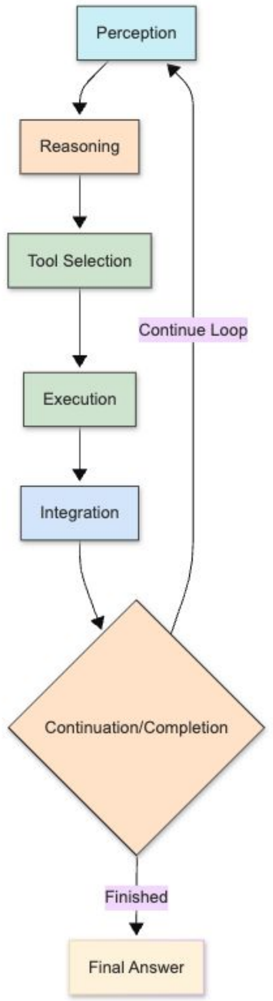
The Agentic Loop

```
# Initialize conversation
messages = [{"role": "user", "content": "Check weather in Paris and Berlin"}]

# Enter agentic loop
while True:
    # 1. Get model response
    response = client.responses.create(model="gpt-4", input=messages, tools=tools

    # 2. Check for tool calls & execute them
    if response.output and any(item.type == "function_call" for item in response.
        for tool_call in response.output:
            if tool_call.type == "function_call":
                # Execute function
                args = json.loads(tool_call.arguments)
                result = execute_function(tool_call.name, args)

                # Add result to conversation
                messages.append(tool_call)  # Add function call
                messages.append({           # Add function result
                    "type": "function_call_output",
                    "call_id": tool_call.call_id,
                    "output": str(result)
                })
    # 3. Check if agent has final answer
    elif response.output_text:
        print(response.output_text)
        break
```



Advanced Agent Patterns

Objective-Based Agents:

- Define success criteria (objective function)
- Continue loop until objective is achieved
- Example: "Search weather in 5 different cities"

Key Implementation Patterns:

- **Parallel Tool Calls:** Multiple functions in one turn
- **Tool Choice Control:** Auto, Required, or Forced selection modes
- **Strict Mode:** Ensuring function arguments match schema exactly
- **Streaming:** Real-time progress updates during function calls

Best Practices

- Write **clear function descriptions** and **parameter documentation**
- Keep function **schemas simple** and intuitive
- Apply **software engineering principles** (least surprise, no invalid states)
- Prefer **fewer, more powerful functions** over many specialized ones

Next Steps 🚀

Let’s get hands on and practice learning how to do tool calling, followed by building agentic loops!