From Tool Loop to Agent: Takeaways

by Dataquest Labs, Inc.

Syntax

- Returning a problem from a tool as data instead of raising:

```
return {"error": str(e)}
```

- Running the model's Python through the subprocess wrapper:

```
run_command([sys.executable, "-c", code], shell=False)
```

- Running a shell command through the same wrapper:

```
run_command(command, shell=True)
```

- Dispatching a tool call by name:

```
TOOLS[name]["fn"](**arguments)
```

- Building the model request and unpacking it into the call:

```
request = {"model": config.model, "messages": messages, "tools": tools.tool_schemas()}
response = client.chat.completions.create(**request)
```

- Parsing the model's tool-call arguments:

```
arguments = json.loads(tool_call.function.arguments)
```

- Appending a tool result to the conversation:

```
messages.append({"role": "tool", "tool_call_id": tool_call.id, "content": json.dumps(result)})
```

Concepts

- **Agent**: a program that chooses its own steps toward a goal, using general tools instead of a path you planned in advance.
- **General tools**: tools that are not tied to one computation, such as reading a file or running code, which the model combines to handle questions you did not plan for.
- **Error as data**: a tool returns a problem as a value the model can read instead of raising, so a failed call becomes feedback rather than a crash.
- **Workspace**: the sandbox folder the agent works in; the file tools keep every model-chosen path inside it.
- **The agent loop**: call the model, run any tools it asks for, add the results to the messages, and repeat until it answers or hits a limit.
- **Stop reason**: how a run ended, one of final, max_steps, budget, or error, which tells you whether to trust the answer or act next.
- **Tracing**: a per-run record of each event, so you can reconstruct what the agent did when you did not watch it run.
- **Verification through the system prompt**: steps in the prompt that tell the agent to explore first and check its results, making good practice dependable rather than left to chance.

Resources

- subprocess (Python docs)
- pathlib (Python docs)
- json (Python docs)

© Dataquest Labs, Inc. All rights reserved.