

lab

April 3, 2026

1 Lab 07 — Tool Use & Function Calling

Build a multi-tool, multi-turn agent loop with Claude.

Exercises: 1. Implement `execute_tool()` for three tools 2. Build the full tool loop (`run_with_tools()`) 3. Test parallel tool calls

```
[ ]: # Setup
import json
from dotenv import load_dotenv
import anthropic

load_dotenv()
client = anthropic.Anthropic()
MODEL = "claude-sonnet-4-20250514"
print("Setup complete.")
```

```
[ ]: # Tool Definitions (provided)

TOOLS = [
    {
        "name": "calculator",
        "description": "Perform basic arithmetic: add, subtract, multiply, ↵
↵divide.",
        "input_schema": {
            "type": "object",
            "properties": {
                "operation": {
                    "type": "string",
                    "enum": ["add", "subtract", "multiply", "divide"],
                    "description": "The arithmetic operation to perform."
                },
                "a": {"type": "number", "description": "First operand."},
                "b": {"type": "number", "description": "Second operand."}
            },
            "required": ["operation", "a", "b"]
        }
    },
]
```

```

{
    "name": "unit_converter",
    "description": "Convert a value between common units (km/miles, kg/lb,↵
↵celsius/fahrenheit).",
    "input_schema": {
        "type": "object",
        "properties": {
            "value": {"type": "number", "description": "The numeric value↵
↵to convert."},
            "from_unit": {
                "type": "string",
                "enum": ["km", "miles", "kg", "lb", "celsius",↵
↵"fahrenheit"],
                "description": "Source unit."
            },
            "to_unit": {
                "type": "string",
                "enum": ["km", "miles", "kg", "lb", "celsius",↵
↵"fahrenheit"],
                "description": "Target unit."
            }
        },
        "required": ["value", "from_unit", "to_unit"]
    }
},
{
    "name": "string_tools",
    "description": "Perform string operations: upper, lower, reverse,↵
↵count_words.",
    "input_schema": {
        "type": "object",
        "properties": {
            "operation": {
                "type": "string",
                "enum": ["upper", "lower", "reverse", "count_words"],
                "description": "The string operation to perform."
            },
            "text": {"type": "string", "description": "The input text."}
        },
        "required": ["operation", "text"]
    }
}
]
print(f"{len(TOOLS)} tools defined: {[t['name'] for t in TOOLS]}")

```

1.1 Exercise 1 — Implement `execute_tool()`

Dispatch to the correct logic for each tool and return a **string** result.

Conversion factors: | From | To | Factor | |——|——|———| | km | miles | 0.621371 | | miles | km | 1.60934 | | kg | lb | 2.20462 | | lb | kg | 0.453592 | | celsius | fahrenheit | $(C * 9/5) + 32$ | | fahrenheit | celsius | $(F - 32) * 5/9$ |

```
[ ]: def execute_tool(tool_name: str, tool_input: dict) -> str:
    """Execute a tool by name with the given input dict. Return result as a
    ↪string."""
    # TODO: Implement dispatch for calculator, unit_converter, string_tools
    raise NotImplementedError("Exercise 1: implement execute_tool")

[ ]: # Test Exercise 1
print(execute_tool("calculator", {"operation": "multiply", "a": 17, "b": 843}))
# Expected: 14331.0 (or similar)

print(execute_tool("unit_converter", {"value": 100, "from_unit": "km",
    ↪"to_unit": "miles"}))
# Expected: 62.1371 miles

print(execute_tool("string_tools", {"operation": "reverse", "text": "Hello,
    ↪World!"}))
# Expected: !dlroW ,olleH

print(execute_tool("string_tools", {"operation": "count_words", "text": "the
    ↪quick brown fox"}))
# Expected: 4
```

1.2 Exercise 2 — The Full Tool Loop

Implement `run_with_tools(user_message, tools, max_iterations=10)` that: 1. Sends the message to Claude with tools 2. Handles `tool_use` stop reason by calling `execute_tool()` and feeding results back 3. Returns Claude's final text when `stop_reason == "end_turn"` 4. Returns a fallback string if `max_iterations` is exceeded

```
[ ]: def run_with_tools(user_message: str, tools: list, max_iterations: int = 10) ->
    ↪str:
    """Send user_message to Claude with tools. Handle the full tool loop.
    ↪Return final text."""
    # TODO: Implement the agentic tool loop
    raise NotImplementedError("Exercise 2: implement run_with_tools")

[ ]: # Test Exercise 2
answer = run_with_tools(
    "What is the square root of 144 multiplied by the number of words in "
```

```
        "'the quick brown fox'? Use your tools to compute the exact answer.",
        TOOLS
    )
    print("FINAL ANSWER:", answer)
```

1.3 Exercise 3 — Parallel Tool Calls

Send a query requiring two independent tool calls and observe whether Claude parallelises them in a single round.

```
[ ]: # Exercise 3: Parallel tool calls
# TODO: Implement the parallel-tool-call test loop below.
#       Track how many tool calls occur per round.

query = (
    "Convert 100 USD to EUR (use 1 USD = 0.92 EUR, so multiply 100 by 0.92) "
    "AND calculate 15% tip on $47.50 (multiply 47.50 by 0.15). "
    "Give me both results. Use your tools."
)

raise NotImplementedError("Exercise 3: implement the parallel tool call test")
```