

lab

April 3, 2026

1 Lab 10 — Building Agents

Build a research agent with mock tools using the ReAct pattern.

Exercises: 1. Build 4 mock tools (search, read_url, save_note, get_notes) 2. Implement the full ReAct agent loop 3. Test with 3 different task types and analyse step counts

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

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

```
[ ]: # Mock Data (provided)

MOCK_SEARCH_DB = {
    "python async": [
        {"title": "Python asyncio Guide", "url": "https://example.com/asyncio",
         "snippet": "asyncio is a library for writing concurrent code using
↪ async/await syntax."},
        {"title": "Async Patterns in Python", "url": "https://example.com/
↪ async-patterns",
         "snippet": "Common patterns include gather, TaskGroup, and async
↪ generators."}
    ],
    "python type hints": [
        {"title": "Type Hints in Python", "url": "https://example.com/
↪ type-hints",
         "snippet": "Type hints improve code readability and enable static
↪ analysis with mypy."},
        {"title": "Advanced Typing", "url": "https://example.com/
↪ advanced-typing",
         "snippet": "Protocol, TypeVar, and ParamSpec enable advanced generic
↪ patterns."}
    ]
}
```

```

],
  "python lists vs tuples": [
    {"title": "Lists vs Tuples", "url": "https://example.com/lists-tuples",
     "snippet": "Lists are mutable sequences; tuples are immutable. Tuples
are hashable and slightly faster."}
  ],
  "python web frameworks": [
    {"title": "Python Web Frameworks", "url": "https://example.com/
web-frameworks",
     "snippet": "Django, Flask, and FastAPI are the most popular Python web
frameworks."},
    {"title": "FastAPI Tutorial", "url": "https://example.com/fastapi",
     "snippet": "FastAPI is a modern, high-performance framework for
building APIs with Python 3.7+."}
  ],
  "python decorators": [
    {"title": "Python Decorators", "url": "https://example.com/decorators",
     "snippet": "Decorators modify function behavior using the @decorator
syntax."}
  ],
}

```

```

MOCK_URL_DB = {
  "https://example.com/asyncio": """Python asyncio Guide\n\nasyncio is
Python's built-in library for asynchronous programming. Key concepts:\n-
Coroutines: defined with async def, paused with await\n- Event loop: runs
coroutines concurrently\n- Tasks: wrap coroutines for concurrent
execution\n- gather(): run multiple coroutines concurrently\n- async with /
async for: async context managers and iterators""",
  "https://example.com/async-patterns": """Common Async Patterns\n\n1.
Fan-out/fan-in with asyncio.gather()\n2. Structured concurrency with
TaskGroup (Python 3.11+)\n3. Async generators for streaming data\n4.
Semaphores for rate limiting\n5. Queue-based producer/consumer patterns""",
  "https://example.com/type-hints": """Type Hints in Python\n\nBasic
annotations: def greet(name: str) -> str\nOptional types: Optional[str] or
str | None (3.10+)\nContainer types: list[int], dict[str, Any]\nCustom types:
TypeAlias, NewType\nRuntime checking: isinstance() still works, hints are
not enforced""",
  "https://example.com/advanced-typing": """Advanced Typing
Patterns\n\nProtocol: structural subtyping (duck typing with type
safety)\nTypeVar: generic functions and classes\nParamSpec: preserving
parameter types in decorators\nConcatenate: adding parameters to callable
types\nOverload: multiple function signatures""",
}

```

```

    "https://example.com/lists-tuples": """Lists vs Tuples in Python\n\nLists:
↳mutable, use [] syntax, support append/extend/pop\nTuples: immutable, use ()
↳syntax, hashable, can be dict keys\nPerformance: tuples are slightly faster
↳for iteration and access\nMemory: tuples use less memory than equivalent
↳lists\nUse lists when you need to modify the collection; use tuples for
↳fixed data."""
    "https://example.com/web-frameworks": """Python Web Frameworks
↳Overview\n\nDjango: full-featured, batteries-included, ORM, admin
↳panel\nFlask: lightweight, flexible, great for microservices\nFastAPI:
↳modern, async-native, auto-generated OpenAPI docs\n\nChoose Django for large
↳apps, Flask for simplicity, FastAPI for APIs."""
    "https://example.com/fastapi": """FastAPI Tutorial\n\nFastAPI uses Python
↳type hints for request validation.\nKey features: automatic docs, dependency
↳injection, async support.\nInstall: pip install fastapi uvicorn\nRun:
↳uvicorn main:app --reload"""
    "https://example.com/decorators": """Python Decorators\n\nA decorator is a
↳function that takes a function and returns a modified function.\nCommon uses:
↳ logging, caching, authentication, retry logic.\n@functools.wraps preserves
↳the original function's metadata.\nClass-based decorators use __call__
↳method."""
}

print(f"Mock DB: {len(MOCK_SEARCH_DB)} search entries, {len(MOCK_URL_DB)} URLs")

```

1.1 Exercise 1 — Build Mock Tools

Implement 4 tools: `search`, `read_url`, `save_note`, `get_notes`. Define both the tool schemas (TOOLS list) and the execution function.

```

[ ]: # TODO: Define TOOLS list with tool schemas for Claude's API
# TODO: Create a notes dict for save_note/get_notes
# TODO: Implement execute_tool(tool_name, tool_input) -> str
# - search: look up query in MOCK_SEARCH_DB (match by substring)
# - read_url: look up URL in MOCK_URL_DB
# - save_note: store in notes dict
# - get_notes: return all notes

raise NotImplementedError("Exercise 1: implement mock tools")

```

```

[ ]: # Test Exercise 1
print(execute_tool("search", {"query": "python async"}))
print(execute_tool("save_note", {"key": "test", "content": "hello"}))
print(execute_tool("get_notes", {}))

```

1.2 Exercise 2 — The ReAct Agent Loop

Implement `run_agent(task, tools, max_steps=10)` with thought/action/observation printing.

```
[ ]: def run_agent(task: str, tools: list, max_steps: int = 10) -> str:
    """Run a ReAct agent loop. Returns the final answer text."""
    # TODO: Implement the full ReAct loop
    raise NotImplementedError("Exercise 2: implement run_agent")

[ ]: # Test Exercise 2
result = run_agent(
    "Research Python async programming. What are the key concepts? Save your
    findings.",
    TOOLS
)
print("\nFINAL ANSWER:", result)
```

1.3 Exercise 3 — Multi-Task Testing

Run 3 tasks and analyse step count per task.

```
[ ]: # TODO: Run 3 tasks, record step counts, tools used, and quality ratings
# Tasks:
# 1. "Research Python type hints. Find key concepts and save a summary."
# 2. "Compare Python lists vs tuples. Search for information about both and
    explain the differences."
# 3. "Create a study plan for learning Python web development. Research the key
    frameworks and tools needed."

raise NotImplementedError("Exercise 3: implement multi-task testing")
```