

lab

April 3, 2026

1 Lesson 06 — Structured Outputs & JSON Mode

In this lab you will implement three increasingly robust techniques for extracting structured data from unstructured text:

1. **Prompt-based JSON** — instruct Claude to return JSON, validate with Pydantic
2. **Tool use** — define a JSON Schema as a tool, let the API enforce it
3. **Retry on error** — feed `ValidationError` back to Claude to recover from failures

The domain is **job posting extraction** — parsing unstructured job descriptions into a structured `JobPosting` object.

```
[ ]: import json
import anthropic
from pydantic import BaseModel, ValidationError
from dotenv import load_dotenv

load_dotenv()
client = anthropic.Anthropic()

# Three job descriptions to use throughout the lab
JD1 = """Senior Python Developer at TechCorp, London. Remote OK. £80-100K.
Must have: 5+ yrs Python, FastAPI, PostgreSQL.
Nice to have: Kubernetes, Rust."""

JD2 = """Marketing Manager - GrowthCo, NYC. On-site. $75,000-$90,000/yr.
Requirements: 3+ years B2B SaaS marketing, HubSpot experience, strong writing
↳skills.
Lead a team of 2."""

JD3 = """Junior ML Engineer, Berlin, Germany. Hybrid (3 days office). €45K-€60K.
Requirements: Python, PyTorch or TensorFlow, math fundamentals.
Recent graduates welcome."""

JOB_DESCRIPTIONS = [JD1, JD2, JD3]
print('Setup complete.')
```

1.1 Exercise 1 — Prompt-Based JSON Extraction

1.1.1 Step 1: Define the Pydantic Model

Define JobPosting with the following fields: - title: str - company: str - location: str - salary_range: str | None (optional, default None) - requirements: list[str] - remote: bool

```
[ ]: class JobPosting(BaseModel):
    # TODO: define fields
    pass

def extract_prompt_based(text: str) -> JobPosting:
    """Extract a JobPosting from raw text using prompt-based JSON."""
    # TODO: call Claude asking for JSON only
    # TODO: strip markdown fences defensively
    # TODO: json.loads() then JobPosting.model_validate()
    raise NotImplementedError("Implement extract_prompt_based")

# Guard: run only when implementation exists
if JobPosting.model_fields: # will be True once fields are defined
    for i, jd in enumerate(JOB_DESCRIPTIONS, 1):
        posting = extract_prompt_based(jd)
        print(f"JD{i}: {posting.model_dump()}")
        print()
```

1.2 Exercise 2 — Tool Use Extraction

1.2.1 Step 1: Define the Tool

Define EXTRACT_TOOL as a dict matching the Anthropic tool schema. The input_schema should be a JSON Schema object corresponding to JobPosting.

Hint: JobPosting.model_json_schema() gives you the Pydantic-generated schema. You may need to clean it up slightly (remove the title top-level key).

1.2.2 Step 2: Implement extract_tool_based

```
[ ]: EXTRACT_TOOL = None # TODO: define tool schema

def extract_tool_based(text: str) -> JobPosting:
    """Extract a JobPosting using Claude tool use (guaranteed schema_
    ↪ conformance)."""
    # TODO: call API with tools=[EXTRACT_TOOL] and tool_choice forcing the tool
    # TODO: find the ToolUseBlock in response.content
    # TODO: validate block.input with JobPosting.model_validate()
    raise NotImplementedError("Implement extract_tool_based")
```

```

# Guard
if EXTRACT_TOOL is not None:
    for i, jd in enumerate(JOB_DESCRIPTIONS, 1):
        posting = extract_tool_based(jd)
        print(f"JD{i}: {posting.model_dump()}")
        print()

```

1.3 Exercise 3 — Retry on Validation Error

Implement `extract_with_retry(text, max_retries=3)` which:

1. Tries to extract and validate
2. On failure, appends the assistant response and a user message containing the error to `messages`
3. Retries up to `max_retries` times

Then test on a deliberately ambiguous input.

```

[ ]: def extract_with_retry(text: str, max_retries: int = 3) -> JobPosting:
    """Extract with retry: feed ValidationError back to Claude to fix."""
    # TODO: implement retry loop with error feedback
    raise NotImplementedError("Implement extract_with_retry")

# Guard: test on ambiguous input
AMBIGUOUS_JD = "We're looking for someone awesome to join our growing team.␣
↳Good pay. Cool office. Tech stuff. Apply now!"

try:
    result = extract_with_retry(AMBIGUOUS_JD)
    print("Extracted:", result.model_dump())
except NotImplementedError:
    print("Not yet implemented - complete extract_with_retry first.")

```