

lab

April 3, 2026

1 Lab 08 — RAG Basics

Build an end-to-end RAG pipeline with ChromaDB, sentence-transformers, and Claude.

Exercises: 1. Embed and store 10 documents in ChromaDB 2. Implement `rag_answer()` — retrieve + generate 3. Metadata filtering comparison

```
[ ]: # Setup
import os
from pathlib import Path
from dotenv import load_dotenv
import anthropic
import chromadb
from sentence_transformers import SentenceTransformer

load_dotenv()
client = anthropic.Anthropic()
MODEL = "claude-sonnet-4-20250514"
embed_model = SentenceTransformer("all-MiniLM-L6-v2")
chroma_client = chromadb.Client()
print("Setup complete.")
```

1.1 Exercise 1 — Embed and Store Documents

Load 10 text files from `demo/sample_docs/`, embed them, and store in a ChromaDB collection. Add metadata: `topic="python"` for docs 1-5, `topic="stdlib"` for docs 6-10.

```
[ ]: # TODO: Load documents, embed them, store in ChromaDB with metadata
# Hints:
# - Use Path("demo/sample_docs/").glob("*.txt") to load files
# - Use embed_model.encode(docs).tolist() to get embeddings
# - Use chroma_client.create_collection("python_docs") to create collection
# - Use collection.add(ids=..., documents=..., embeddings=..., metadatas=...)

raise NotImplementedError("Exercise 1: load, embed, and store documents")
```

```
[ ]: # Test Exercise 1
results = collection.query(query_texts=["file path handling"], n_results=3)
print("Top 3 results for 'file path handling':")
for doc in results["documents"][0]:
    print(f" - {doc[:80]}...")
```

1.2 Exercise 2 — RAG Answer Function

Implement `rag_answer(question, collection, top_k=3)` that retrieves context and generates an answer with Claude.

```
[ ]: def rag_answer(question: str, collection, top_k: int = 3) -> str:
    """Retrieve top_k docs from collection, build prompt, get Claude's answer.
    ↪ """
    # TODO: Implement RAG answer function
    # 1. Embed the question
    # 2. Query the collection for top_k results
    # 3. Build a prompt with retrieved context
    # 4. Send to Claude and return the answer
    raise NotImplementedError("Exercise 2: implement rag_answer")

[ ]: # Test Exercise 2
print(rag_answer("How do I read a JSON file in Python?", collection))
print("----")
print(rag_answer("What is a defaultdict?", collection))
```

1.3 Exercise 3 — Metadata Filtering

Compare filtered vs unfiltered queries using the `topic` metadata field.

```
[ ]: # TODO: Implement filtered vs unfiltered comparison
# 1. Query for "iteration patterns" with where={"topic": "python"}
# 2. Query for the same without filter
# 3. Print and compare results

raise NotImplementedError("Exercise 3: implement metadata filtering comparison")
```