



A HANDS-ON PYTHON GUIDE

Build & Evaluate Your First

RAG Assistant

Find the evidence.
Trace the answer.



Sanjaay Gowda T S

Retrieval, citations and model evaluation

FREE SAMPLE / START HERE + CHAPTER 1

sanjaaygowdats.ai

Inside this guide

Start here..... 3

1. Trace the first question 7

Read the chapters in order, run the companion project, then compare your results with the worked examples and exercise solutions.

Start here

Build and Evaluate Your First RAG Assistant is a practical guide to a small, inspectable question-answering system. You will turn a folder of text documents into searchable evidence, connect that evidence to a local language model, and examine both successful and failed results.

The project uses a fictional shop, Northstar Desk. Its returns, delivery, support and account policies are invented teaching data. They are not the policies of Sanjaay's store. Addresses ending in `example.test` are examples and do not receive mail.

You need basic Python knowledge: functions, dictionaries, lists, command-line arguments and JSON. Use Python 3.10 or later. The companion code uses the Python standard library. A working internet connection is needed to obtain Python, Ollama and a model; the evidence-only exercise then runs offline. The generative exercise requires a separately installed, locally running Ollama model and sufficient memory for that model.

The outcome and the boundary

By the end, you will have a local command-line assistant with inspectable retrieval, an optional language-model connection, exact citation IDs and repeatable evaluation cases. You will also have evidence that a response can pass automated checks while remaining wrong. This is a bounded learning project, with a lexical baseline you can read in one sitting. It is not an embedding-search implementation or a production customer-support service.

The book assumes that you can run a Python script and read a dictionary. If command-line work is new, run the evidence exercises first and leave model installation until Chapter 5. Each chapter has one exercise. Appendix A gives worked answers and acceptance criteria; try the exercise before comparing your result. In the EPUB, wide tables can scroll sideways on narrow screens.

Term	Meaning in this guide
Corpus	The reviewed documents available for one search.
Chunk	A bounded piece of source text with an identity.
Retrieval	Ranking candidate chunks against a question.
Generation	A model composing a new answer from supplied context.
Grounding	Checking that answer claims are supported by that context.
Abstention	Declining to answer when the evidence is insufficient.
Regression	A previously acceptable case failing after a change.

Read Lewis and colleagues' original RAG paper for the research architecture [R1]. Our standard-library lexical project demonstrates the retrieval-plus-generation pattern without reproducing that paper's training method or results.

Two modes, two promises

Evidence mode retrieves and prints source excerpts. It does not generate an answer. It is useful for checking what the retriever actually found, and it runs without a model or API key. Its decision label is `evidence_only`.

Ollama mode sends a question and retrieved excerpts to a local model. It asks for a short answer with exact source IDs, then validates the response shape and citation IDs. This is the generative RAG path. A valid citation is not proof that a claim is true; human review remains necessary.

Edition 1.1.1 includes 17 companion unit tests and a recorded real Ollama run on the bundled synthetic policies. Nine model calls were made across 14 runs, including a repeated question and a lower-threshold experiment. Five runs stopped before generation because retrieval returned no evidence. The small model made both citation and factual errors. Chapters 4-6 examine those errors rather than presenting the run as a quality benchmark. The recorded answer-support judgments were made by the authoring agent; independent human technical and accessibility review is still pending.

Your working folder

Extract companion-project.zip and open a terminal inside the rag-assistant folder. Keep these files together:

```
rag-assistant/  
rag.py  
cases.json  
documents/  
  returns.md  
  delivery.md  
  support.md  
  accounts.md  
tests/test_rag.py  
examples/retrieval-report.json  
README.md  
LICENSE.txt
```

Run `python --version`. On Windows, `py -3` may replace `python` in the commands below. On macOS or Linux, `python3` may replace `python`. No `pip install` command is needed for the companion project.

A first successful setup check

Extract the ZIP to a new folder rather than running code inside an archive viewer. In a terminal, change into the extracted `rag-assistant` directory. Confirm that `rag.py`, `cases.json` and `documents` are siblings. Run these commands before installing a model:

SHELL

```
python --version  
python -m unittest discover -s tests -v  
python rag.py evaluate
```

The tests should report 17 passing cases. The evaluation should report 12 synthetic questions. If Python is not found, install it from the official Python website and reopen the terminal [R6]. If the error says it cannot open `rag.py`, check the working directory. If it says no documents were found, confirm the ZIP was extracted with its directories intact.

Checkpoint: a passing setup means the supplied tests ran on your machine. It does not establish that all possible inputs are supported. Keep the complete folder; citation hashes and reports depend on the bundled text.

Read, run, inspect

Work through the chapters in order the first time. Keep the default documents unchanged until you have run the supplied cases. When experimenting, copy documents into a separate folder and pass that folder with `--docs`. Change one thing at a time and keep the old report for comparison.

Completion means you can explain a retrieved passage, identify a missing answer, demonstrate a citation check and describe a failure that your metrics do not catch. It does not mean the system is ready to answer private customer questions in production.

1. Trace the first question

Begin with evidence

An assistant can produce a fluent sentence before it has found anything relevant. Starting with evidence makes that separation visible. Run:

SHELL

```
python rag.py ask "What is the return window for unused desk accessories?"
```

Inspect decision, answer, citations and retrieved in the JSON output. The decision should be `evidence_only`. The answer contains source text including the 30-calendar-day return rule. The retrieved list includes `returns.md`, a content-derived ID and a numeric score. You may see more than one matching chunk; inspect all returned excerpts.

The ID looks like `returns.md:1` followed by a short content hash. It identifies a file, a chunk position and the exact text version. It is more useful than a decorative citation such as `[1]` with no corresponding source. The program does not open links, submit requests to the fictional shop or process a return.

Follow the path

The pipeline is: documents, chunks, ranked passages, evidence decision, optional generation, citation validation. Each stage has a separate responsibility. Ingestion preserves source identity. Retrieval selects candidates. Generation writes language from those candidates. Validation checks selected properties of the result.

Do not call the evidence score confidence. It measures lexical vector similarity. A high score can come from a passage that discusses the right subject while omitting the requested fact. A low score can hide a valid answer expressed with different words.

Read one actual result

The following excerpt is generated from `examples/worked-retrieval.json`, recorded with the bundled documents, 110-word chunks, top-k of 3 and a minimum score of 0.18. Only the short structural fields are printed here; the CLI also prints the complete retrieved source text.

Recorded excerpt - full answer text omitted here

```
{
  "decision": "evidence_only",
  "citations": [
    "returns.md:1:600682483a84"
  ],
  "retrieved": [
    {
      "source": "returns.md",
      "id": "returns.md:1:600682483a84",
      "score": 0.379301
    }
  ]
}
```

Read the fields in order. `evidence_only` tells you that no language model was involved. The citation identifies the exact returned text. The source is `returns.md`. The score, 0.379301, ranks this chunk for this question; it is not a 37.9% probability of correctness.

Open `documents/returns.md` and locate the sentence about unused accessories. It says **30 calendar days of delivery**. The number, the kind of day and the starting event all matter. A response that says only "30 days" is less complete. A response that starts the clock at the order date is wrong even though it uses the correct number.

The inspectable RAG path

Figure 1. Follow reviewed documents through chunks, retrieval, optional local generation, citation validation and a separate claim-support review.

Read the figure from the top row, left to right, then the bottom row. Documents become chunks; the retriever selects evidence; a local model can generate an answer; the validator checks shape and source IDs; a reviewer checks whether the source supports the claim. Evidence mode exits after retrieval and quotes the selected chunks. It does not silently perform the final review.

Compare an unrelated question

SHELL

```
python rag.py ask "Which planets have rings?"
```

The supplied documents have no astronomy evidence. This run should return `insufficient_evidence`, an empty citation list and no retrieved passages. The model is never called when the retriever supplies no evidence, even if you select Ollama mode.

This is a cheap, deterministic refusal. It handles one kind of missing evidence: no lexical match above the threshold. It cannot prove that a matching passage answers the question. You will examine that distinction in Chapter 6.

Exercise: question versus topic

Ask what personalised accessories may be returned. Read the entire matching passage, including its exception for damage. Then ask whether personalised accessories can be returned after 90 days. Does retrieving the correct policy settle both the product condition and the date? Write down which facts are present and which assumptions a generated answer would need to avoid.

The useful outcome is a source-backed explanation. A plausible yes or no without attention to all conditions is not enough.

Continue the build

The complete edition includes nine chapters, six explanatory figures, measured retrieval comparisons, recorded local-model results, an exercise answer guide and a runnable companion project.

[Explore the full edition on sanjaaygowdats.ai](https://sanjaaygowdats.ai)

This free sample may be shared unchanged. The full ebook is licensed to one reader. The companion project is separately MIT licensed.