COMPLETE FREE-TO-START GUIDE

# BUILD AI AGENTS FROM ZERO

A practical 8-week path using free software and local models.

Learn the foundations, build useful agents, and finish with four portfolio projects.

## What you are building

An AI agent takes a goal, uses a model to choose from approved tools, checks what happened, and responds or asks a person for help. Begin with one narrow task; add autonomy only after the basic version works.

| GOAL | MODEL | TOOLS | CHECK | RESULT |
|---|---|---|---|---|
| What needs doing? | Interpret the request | Call a safe function | Validate the result | Answer or ask a human |

### Recommended no-API-bill starter stack

**Python + VS Code + Git**

Write, run, and version your projects. All can be used without paying for an AI API.

**Ollama + a local model**

Run a compatible model on your computer. Model size and speed depend on RAM, storage, and hardware.

**Ollama Python library + Pydantic**

Connect your Python code to the local model and validate structured inputs and outputs.

**SQLite + pytest + Streamlit**

Store simple local data, check expected behavior, and build a small local demo interface.

**What “free” means here:** The main learning path uses software you can install and run locally, without a paid model API. Your computer, electricity, internet, and storage still have costs. Cloud free tiers have quotas and can change; check terms before using them.

# Free tools by job

You do not need every tool. Start with the bold starter choices, then add optional frameworks only when a project needs them. Open-source licenses and model terms differ, so review them before commercial use.

| AREA | STARTER / FREE OPTIONS | WHAT YOU USE IT FOR |
|---|---|---|
| Code editor | Python, VS Code, Jupyter | Write and run Python; notebooks are useful for experiments. |
| Version control | Git; GitHub Free for remote repos | Track changes, back up code, and share a portfolio. |
| Local model runtime | Ollama; LM Studio (desktop option) | Download and run compatible open models locally; no per-request API bill. |
| Python model connection | Ollama Python library; HTTP requests | Send prompts to the local model and receive outputs. |
| Agent frameworks | LangGraph; CrewAI; LlamaIndex | Optional orchestration, multi-step graphs, agent teams, and retrieval. |
| Structured data / validation | JSON; Pydantic | Define tool inputs and validate model outputs before using them. |
| Local storage / retrieval | SQLite; Chroma; FAISS | Save simple data; add semantic search over your own documents. |
| API / backend | FastAPI | Expose a local agent as a small API and inspect its API docs. |
| Web interaction | Playwright for Python | Automate controlled browser tasks; use only permitted sites and data. |
| Simple user interface | Streamlit; Gradio | Create a local chat or demo interface with Python. |
| Testing / quality | pytest; Ruff | Check expected outputs, catch regressions, and format/lint Python. |
| Data for practice | Synthetic CSVs; public sample datasets | Practice safely without uploading private personal or company data. |

**Keep your setup simple:** Python + VS Code + Git + Ollama + the Ollama Python package + pytest is enough to learn a first agent. Use a framework, vector store, or hosted service only when you can explain why the project needs it.

# Weeks 1-3 | Build the foundations

Aim for 5-7 hours per week. The schedule is flexible: finish each small build before moving on.

### 01 | Python essentials

**Learn:** Variables, conditions, loops, functions, lists, dictionaries, exceptions, files, modules, and virtual environments.

**Free tools:** Python, VS Code, Jupyter, Git

**Build:** Write a CSV reader that counts rows, handles blank values, and summarizes totals.

### 02 | APIs and data

**Learn:** HTTP methods, status codes, JSON, requests, environment variables, and simple schemas.

**Free tools:** requests, Pydantic, public APIs, SQLite

**Build:** Call a public API, validate the response, and save selected fields in SQLite.

### 03 | LLM basics - local first

**Learn:** Prompts, system instructions, context limits, temperature, structured output, and common model errors.

**Free tools:** Ollama, local compatible model, Ollama Python library

**Build:** Create a local message classifier; test it against 20 made-up examples.

## Milestone at the end of week 3

You can write a Python program, call a local model, pass it data, validate its answer, and explain where the model may fail. Save the code in Git and add a short README.

# Weeks 4-6 | Turn a model into an agent

First make one safe tool work. Then add a controlled workflow and retrieval only if needed.

### 04 | Tool calling

**Learn:** Define small Python functions; describe each tool; validate arguments; return clear tool results; add a step limit.

**Free tools:** Ollama tool calling or a simple Python dispatcher; Pydantic; pytest

**Build:** Build an FAQ helper that searches local text files and a calculator tool.

### 05 | State and workflow

**Learn:** Track the current task, handle tool errors, stop loops, and separate model decisions from fixed business rules.

**Free tools:** Plain Python first; LangGraph as an optional workflow framework

**Build:** Build a two-step workflow: search a note, then draft a grounded answer.

### 06 | Documents and retrieval

**Learn:** Chunk documents, create embeddings, retrieve relevant snippets, and show the supporting source.

**Free tools:** Chroma or FAISS; sentence-transformers; Ollama local embeddings where supported

**Build:** Create a local document Q&A; demo that answers “not found” when evidence is missing.

## Avoid these beginner traps

Do not give an agent unrestricted shell, browser, email, or database access. Do not add multi-agent orchestration just to make the project look advanced. Prove a single-agent workflow first.

# Weeks 7-8 | Test, present, and improve

A useful agent needs predictable limits and clear evidence that it works. Keep the demo local unless you are ready to manage hosting, privacy, and costs.

### 07 | Interface and API

**Learn:** Build a simple chat or form UI; return clear errors; keep secrets outside code.

**Free tools:** Streamlit or Gradio; FastAPI if you need an API

**Build:** Package one project so another person can run it locally from your instructions.

### 08 | Evaluation and safety

**Learn:** Create test cases; measure correct answers and tool use; add logs, timeouts, step limits, and human approval.

**Free tools:** pytest; Ruff; Git; synthetic test data

**Build:** Show a demo, test results, known limitations, and the next improvement in your README.

## Simple evaluation checklist

| | |
|---|---|
| [ ] 20 realistic sample requests | [ ] Correct tool chosen |
| [ ] Invalid input handled safely | [ ] Missing answer triggers “I don’t know” |
| [ ] Step limit stops loops | [ ] Consequential actions need approval |

## Free vs. free-tier

Local software and local models can avoid model API charges, but need suitable computer hardware. Online notebooks, hosted models, and app hosting may offer limited free access; quotas, eligibility, and terms can change. Never promise viewers unlimited free cloud use.

# 16 AI agent ideas to explore

Start with read-only projects. Ask for human approval before an agent sends messages, changes records, spends money, or deletes files.

### 01 FAQ agent

Answers from an approved local knowledge base.

### 02 Research assistant

Collects findings and returns source links.

### 03 Email triage helper

Labels sample messages and drafts for review.

### 04 Meeting notes helper

Extracts decisions, owners, and due dates.

### 05 CSV analysis agent

Answers questions using checked calculations.

### 06 SQL query helper

Drafts and explains queries for human review.

### 07 UiPath error guide

Finds relevant tips in your troubleshooting notes.

### 08 Job match assistant

Ranks public job listings against user criteria.

### 09 Document intake helper

Extracts sample form fields and flags gaps.

### 10 Invoice checker

Compares sample invoice and purchase-order fields.

### 11 Web research helper

Summarizes allowed web pages with links.

### 12 Learning coach

Makes quizzes and suggests practice topics.

### 13 Code review helper

Checks a small code change against a checklist.

### 14 Test case generator

Creates draft tests from a requirement.

### 15 Personal planner

Turns a goal into suggested tasks and dates.

### 16 Workflow log monitor

Highlights repeated errors in sample logs.

# 4 projects to build in order

Finish these to show a clear progression from retrieval to a useful, tested agent. Use synthetic or public sample data.

### 1. Local FAQ agent

Tools: local text search + Ollama.

**Done when:** Success check: cites the matching snippet and says it cannot find an answer when the note is missing.

### 2. CSV insights agent

Tools: Python calculations + SQLite or pandas.

**Done when:** Success check: calculations match manually checked sample questions; model does not invent values.

### 3. Job match assistant

Tools: sample job list + profile fields + ranking rules.

**Done when:** Success check: every match gives a reason and link; no fabricated requirements.

### 4. UiPath error guide

Tools: a small curated troubleshooting knowledge base + local retrieval.

**Done when:** Success check: cites the note used and asks for the error text/version if context is missing.

## Official learning links

Python tutorial | VS Code | Git documentation | Ollama quickstart | LangGraph
Chroma | FastAPI | Playwright Python | Streamlit | pytest

**Want to build your first agent? Start with the local FAQ project.**

Tool availability and licensing can change. Confirm license terms for each framework, model, dataset, and extension before commercial use. This guide is educational and favors local-first tools; it does not guarantee that hosted services will remain free.