GENAI ROADMAP

120 concepts to become a GenAI engineer

8 areas, 15 concepts each. Every concept explained in one line, the order to study each area, and two practice questions to test yourself.

How to use it: one area at a time, tick each concept only when you can explain it out loud in under a minute, then answer the two practice questions without notes.

@chalkesahil · chalkesahil.ai

AREA 01 / 08

Python for AI

Study order: Get fluent calling APIs from Python before any framework.

01	**Functions and modules**	Small pieces you can test and reuse.
02	**Virtual environments**	venv or uv so dependencies never clash.
03	**Type hints**	Frameworks read them to build tool schemas.
04	**Pydantic**	Validate structured data, including model output.
05	**JSON**	The format models, tools and APIs all speak.
06	**HTTP requests**	requests or httpx to call any API.
07	**async and await**	Run many model calls at once.
08	**Environment variables**	Keep API keys out of your code.
09	**Error handling**	Catch timeouts and bad responses.
10	**Retries with backoff**	Rate limits are normal; retry politely.
11	**FastAPI**	Wrap your AI feature in an API.
12	**Logging**	Record every prompt and response.
13	**Jupyter notebooks**	Try ideas fast before writing the app.
14	**pandas basics**	Load and clean the data you feed a model.
15	**Git**	Version prompts and code together.

Practice

1. Call an LLM API and return clean JSON from a function.
2. Send 20 requests in parallel and handle the ones that fail.

AREA 02 / 08

LLM basics

Study order: Learn how models behave before you build on one.

01	**Tokens**	Models read and bill in tokens, not words.
02	**Context window**	Everything the model can see in one call.
03	**Temperature**	Higher is more varied, lower is more consistent.
04	**System prompt**	Standing instructions for every reply.
05	**Chat roles**	System, user and assistant turns.
06	**Next-token prediction**	The model writes one token at a time.
07	**Hallucinations**	Confident wrong answers; design for them.
08	**Model choice**	Bigger is smarter, smaller is cheaper and faster.
09	**Pricing**	Input and output tokens cost differently.
10	**Latency**	Long prompts and long answers both add wait.
11	**Rate limits**	Requests and tokens per minute.
12	**Structured output**	Ask for JSON that matches a schema.
13	**Reasoning models**	Models that think longer before they answer.
14	**Multimodal input**	Many models read images and files too.
15	**Open vs closed models**	Run Llama or Qwen yourself, or call an API.

Practice

1. Why does the same prompt give two different answers?
2. Estimate the monthly cost of 10,000 chats a day.

AREA 03 / 08

Prompting

Study order: Write prompts like specs: role, task, context, format, examples.

01	**Clear instructions**	Say exactly what you want and what to avoid.
02	**Role and context**	Tell the model who it is helping and why.
03	**Few-shot examples**	Show two or three good answers.
04	**Output format**	Specify keys, length and tone.
05	**Step-by-step reasoning**	Let the model think on hard tasks.
06	**Delimiters**	Separate instructions from data.
07	**Prompt templates**	Reusable prompts with variables.
08	**Prompt chaining**	Split a big task into smaller prompts.
09	**Self-check**	Ask the model to verify its answer.
10	**Tool calling**	Let the model ask your code to run a function.
11	**Function schemas**	Describe each tool's inputs clearly.
12	**Prompt injection**	Text in data that tries to take over.
13	**Guardrails**	Block unsafe inputs and outputs.
14	**Long context tips**	Put the question after long documents.
15	**Prompt versioning**	Track prompt changes like code.

Practice

1. Write a prompt that returns a product review as JSON.
2. A user pastes text that says ignore your rules. How do you defend?

AREA 04 / 08

RAG

Study order: Give the model your data at question time instead of retraining it.

01	**What RAG is**	Retrieve relevant text, then generate the answer.
02	**Document loading**	Read PDFs, pages and docs into text.
03	**Chunking**	Split documents into searchable pieces.
04	**Chunk overlap**	Keep context across chunk edges.
05	**Embeddings**	Turn text into vectors by meaning.
06	**Similarity search**	Find the chunks closest to the question.
07	**Top-k**	How many chunks to pass to the model.
08	**Metadata filters**	Search only the right docs.
09	**Hybrid search**	Combine keyword and vector search.
10	**Reranking**	Reorder results with a stronger model.
11	**Query rewriting**	Turn a vague question into a better search.
12	**Context assembly**	Fit the best chunks into the prompt.
13	**Citations**	Show which source each answer came from.
14	**Re-indexing**	Keep the index fresh when docs change.
15	**RAG failure modes**	Wrong chunk, missing chunk, or ignored chunk.

Practice

1. Build a chatbot that answers from 50 PDFs with citations.
2. Answers are wrong though the right doc exists. Where do you look?

AREA 05 / 08

Vector databases

Study order: Know what the index does before picking a product.

01	**Vectors**	Lists of numbers that capture meaning.
02	**Cosine similarity**	How close two vectors point.
03	**Dot product**	A fast similarity score.
04	**Dimensions**	Vector size, set by the embedding model.
05	**Approximate nearest neighbour**	Fast search that trades a little accuracy.
06	**HNSW**	A graph index most vector databases use.
07	**IVF**	An index that searches clusters.
08	**pgvector**	Vector search inside Postgres.
09	**Managed vs self-hosted**	Pinecone or Qdrant Cloud versus running your own.
10	**Collections**	Separate indexes for separate data.
11	**Metadata**	Fields stored next to each vector.
12	**Upserts and deletes**	Keep the index in sync with source data.
13	**Recall vs speed**	Tune how hard the search looks.
14	**Batching**	Insert many vectors at once.
15	**Cost**	Storage, memory and queries add up.

Practice

1. Store 10,000 chunks in pgvector and query the top 5.
2. Search got slow at 5 million vectors. What do you change?

AREA 06 / 08

Fine-tuning

Study order: Prompt first, add RAG next, fine-tune last.

01	**When to fine-tune**	For style and format, not for new facts.
02	**Datasets**	Hundreds to thousands of good examples.
03	**Chat data format**	Input and ideal output in turns.
04	**Train and validation split**	Hold some examples back to test.
05	**LoRA**	Train a small add-on instead of the whole model.
06	**QLoRA**	LoRA on a quantised model to save memory.
07	**Quantisation**	Smaller numbers so models fit on less GPU.
08	**Hugging Face Transformers**	The common library for open models.
09	**PEFT**	Hugging Face library for LoRA-style training.
10	**Epochs and learning rate**	How long and how fast it learns.
11	**Overfitting**	Great on training data, bad on new input.
12	**Evaluate after tuning**	Compare with the base model on the same tests.
13	**Distillation**	Teach a small model from a big one.
14	**Open-weight models**	Llama, Qwen and Mistral you can tune.
15	**Licences**	Check what each model allows commercially.

Practice

1. Fine-tune a small model to write support replies in your tone.
2. Your tuned model got worse at maths. Why?

AREA 07 / 08

Evals

Study order: If you cannot measure it, you cannot ship it.

01	**Golden dataset**	Questions with known good answers.
02	**Rule checks**	Exact match, regex and JSON validity.
03	**LLM as judge**	A model grades another model's answer.
04	**Rubrics**	Clear scoring criteria for the judge.
05	**Regression tests**	Make sure a change did not break old cases.
06	**A/B comparison**	Compare two prompts or models side by side.
07	**Groundedness**	Is the answer supported by the sources?
08	**Hallucination checks**	Catch claims with no source.
09	**Latency and cost tracking**	Quality is not the only metric.
10	**Human review**	Spot-check what automated scores miss.
11	**Evals in CI**	Run them on every prompt change.
12	**Tracing**	See every step of a request.
13	**Red-teaming**	Try to break your own app.
14	**Safety tests**	Harmful requests should be refused.
15	**Feedback loops**	Turn user thumbs-down into new test cases.

Practice

1. Build a 50-question eval for a RAG bot.
2. A new prompt scores higher but users complain. What happened?

AREA 08 / 08

Deployment

Study order: Ship it fast, cheap and safely.

01	**API serving**	FastAPI or a serverless function in front.
02	**Streaming responses**	Users see text immediately.
03	**Caching**	Reuse answers to repeat questions.
04	**Rate limiting**	Protect your API and your bill.
05	**Cost control**	Budgets, alerts and shorter prompts.
06	**Model routing**	Easy questions to cheap models.
07	**Fallbacks**	Switch models when one fails.
08	**Observability**	Log prompts, outputs, latency and errors.
09	**Secrets management**	Keys in a vault, not in code.
10	**PII handling**	Mask personal data before sending it.
11	**Docker**	Package the app the same everywhere.
12	**GPU inference**	vLLM or similar to serve open models.
13	**Autoscaling**	Add capacity when traffic spikes.
14	**Versioning**	Pin model and prompt versions.
15	**Monitoring quality**	Watch answers drift after launch.

Practice

1. Deploy a RAG API with streaming, caching and rate limits.
2. Your bill tripled overnight. How do you find the cause?