LangChain Cheat Sheet

Getting Started with LangChain (slides + demos)

Lucas Soares — 15 May 2025

Core idea

• LLM: predicts the next token in a sequence.
• LangChain: framework for building LLM-powered apps with reusable abstractions (models, prompts, parsers, chains).
• Typical session flow: slides → notebook demo → Q&A; (+ optional exercise).

LangChain components

• **Models** - wrappers over LLM APIs (OpenAI, etc.).
• **Prompt Templates** - reusable prompts with variables ("{concept}").
• **Output Parsers** - convert raw model output into usable types (string/JSON/etc.).

Pattern

```
chain = prompt | llm |
StrOutputParser()
chain.invoke({'text': '...'})
```

Chains + LCEL

• A **chain** composes steps: prompt → model → parser.
• LCEL uses pipe syntax (**|**) to compose and reuse chains.

Pattern

```
chain = prompt | llm | parser
result = chain.invoke({'topic':
'...'} )
```

Why it matters: you can swap pieces (prompt/model/parser) without rewriting everything.

LCEL: runnables

• Many LangChain building blocks implement the **Runnable** interface.
• Standard methods: **invoke**, **batch**, **stream** (async variants available).
• Inputs/outputs vary by component (prompt, model, parser, chain).

OPTIONAL EXERCISE

Build a summarization chain with **prompt | model | parser** and test it on a paragraph from your notes.

Stretch: add **stream()** so you can see tokens arrive live.

LangChain Cheat Sheet

RAG + Agents + Deployment

RAG: chat over documents

- **RAG** = Retrieval Augmented Generation (LLM + relevant retrieved docs).
- Motivation: work around context-length limits and improve factual grounding.
- **Embeddings**: convert text into vectors so you can do similarity search.

Pipeline: Load → Split → Embed → Store → Retrieve → Generate

TIP

If you remember one thing: embeddings let you find the most relevant chunks fast, then you feed only those to the LLM.

Self-check

- **LangChain advantage**: unified abstractions that streamline development. (B)
- **Prompt template**: reusable prompt structure with variables. (B)
- **Embeddings in RAG**: vectors for similarity comparison. (B)
- **Agent loop**: iterative decide-act-observe cycle. (B)

References

- LangChain: Introduction
- LangChain: Agents Concepts
- LangGraph: Concepts
- Gen Agents (arXiv:2304.03442)

Agents

- **Agent loop**: decide action → call tool(s) → observe results → repeat until done.
- **Tools**: functions with an input schema + execution logic (often bundled in toolkits).
- Use agents when you need routing decisions, tool selection, and iterative problem solving.

LangGraph

- Models agent workflows as **graphs**.
- **Nodes**: functions that update a shared **State**.
- **Edges**: fixed/conditional transitions (routing logic).
- Designed for controllability, persistence (human-in-the-loop), and streaming-first UX.

LangServe

- Deploy chains/runnables as a REST API (FastAPI + Pydantic validation).
- Common endpoints: **/invoke**, **/batch**, **/stream**.
- Includes a **/playground** UI; optional tracing to LangSmith.

OPTIONAL EXERCISE

Build a simple RAG pipeline (load → split → embed → store), then answer questions about a PDF.