LangChain: Building Applications with LLMs

A beginner-friendly introduction to LangChain

What is LangChain?

An open-source framework for developing applications powered by large language models (LLMs).

LangChain provides powerful tools to connect LLMs with:

External Data

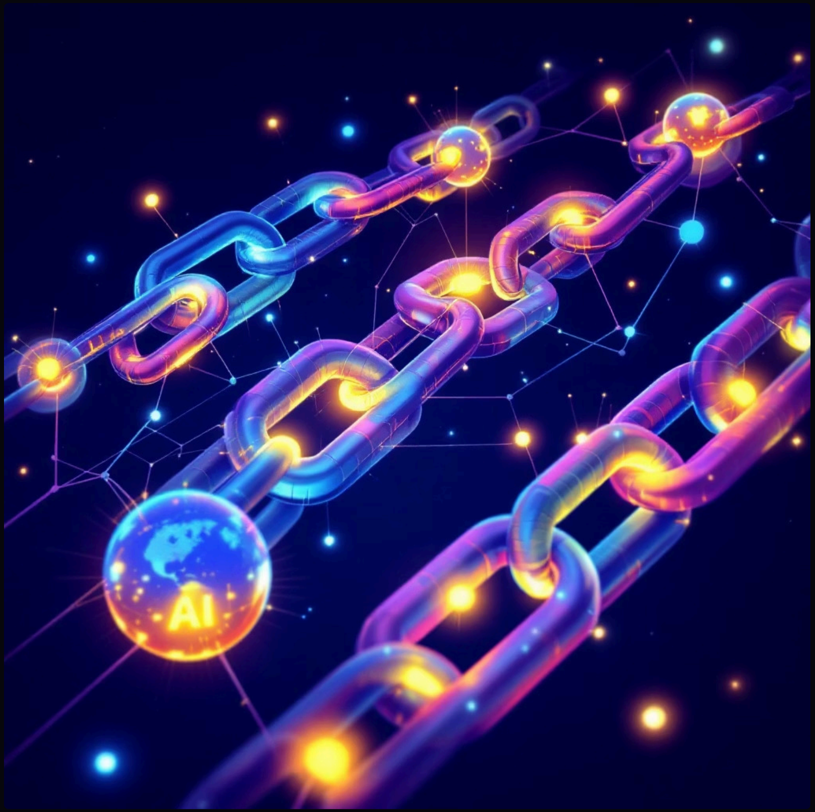
APIs, databases, documents

Chains of Reasoning

Multi-step workflows

Memory

Stateful conversations



Why LangChain?

Simplifies Integration

Makes it easy to integrate LLMs into real-world applications without complex setup

Reusable Components

Pre-built chains, agents, and memory systems that can be combined and customised

Community Ecosystem

Extensive library of integrations and active developer community support

Production Ready

Supports both rapid prototyping and scalable production deployments

Core Concepts

Prompt Templates

Standardise inputs to the LLM with reusable, parameterised prompts

Chains

Link multiple LLM calls together into complex workflows and pipelines

Agents

Dynamically decide actions and tool usage based on LLM responses

Memory

Maintain context and conversation history across interactions

Tools

Connect to APIs, databases, file systems, and external services

Example Use Cases

Chatbots with Memory

Conversational AI that remembers previous interactions and maintains context

Content Generation

Automated writing, summarisation, and creative content production

Document Q&A (RAG)

Retrieval-Augmented Generation for answering questions about your documents

Personal AI Assistants

Intelligent assistants that can perform tasks and answer questions

Automated Data Analysis

AI-powered insights and reporting from your datasets

Installing LangChain

01

Basic Installation

```
pip install langchain openai
```

This installs the core LangChain framework along with OpenAI integration

02

Optional Dependencies

```
pip install chromadb faiss
```

Add vector databases and retrieval capabilities for document Q&A applications

Pro Tip: Consider using a virtual environment to manage dependencies and avoid conflicts with other Python projects.

First Example: Simple Prompt

Basic LLM Interaction

The simplest way to get started with LangChain is to create a basic LLM instance and send it a prompt.

```
# .env OPENAI_API_KEY=key
from langchain.llms import OpenAI

llm = OpenAI()
response = llm("Tell me a fun fact about space.")
print(response)
```

This example demonstrates the fundamental building block of LangChain - direct interaction with a language model.

✓ This basic pattern forms the foundation for all more complex LangChain applications!

Using a Chain

Building Reusable Prompt Templates

```
from langchain import PromptTemplate, LLMChain
from langchain.llms import OpenAI

template = "What is a good name for a company that makes {product}?"
prompt = PromptTemplate(
    input_variables=["product"],
    template=template
)

llm = OpenAI()
chain = LLMChain(llm=llm, prompt=prompt)
print(chain.run("AI chatbots"))
```

Template Creation

Define a reusable prompt structure

Chain Assembly

Combine template with LLM

Execution

Run with dynamic inputs

Adding Memory (Chat Example)

Creating Conversational AI with Context

```
from langchain.chains import ConversationChain
from langchain.memory import ConversationBufferMemory
from langchain.llms import OpenAI

llm = OpenAI()
memory = ConversationBufferMemory()
conversation = ConversationChain(llm=llm, memory=memory)

print(conversation.run("Hi, my name is Alex."))
print(conversation.run("What's my name?"))
```

1

First Interaction

"Hi, my name is Alex."

AI learns and stores the name

2

Follow-up Query

"What's my name?"

AI recalls: "Your name is Alex!"

Memory enables truly conversational experiences!