LangChain 是一个用于构建基于大型语言模型（LLM）应用的开发框架。它的核心目标是帮助开发者将语言模型与外部数据源、工具系统以及复杂逻辑流程进行组合，从而构建出真正可用的智能应用系统。随着大模型技术的发展，仅仅调用一个模型接口已经无法满足企业级应用需求，因此 LangChain
提供了一整套模块化设计来支持复杂场景开发。

一、LangChain 的核心模块

1. LLM（语言模型接口层）

LangChain 提供统一的接口封装，用于调用不同的大模型服务，例如
OpenAI、Anthropic、本地模型等。开发者只需要使用统一
API，即可在不同模型之间进行切换。这种抽象设计降低了供应商绑定风险。

2. Prompt Templates（提示词模板）

提示词是驱动大模型输出质量的关键。LangChain
提供模板系统，可以动态插入变量，使提示构建更加结构化。例如可以定义：

“ 请根据以下内容回答问题：{context}。问题是：{question}。 ”

在实际调用时再填入具体内容。这种方式有利于构建可维护的 Prompt 工程体系。

3. Chains（链式调用）

Chain 是 LangChain 的核心思想之一。它允许将多个步骤串联在一起，例如：

用户问题 → 检索相关文档 → 拼接 Prompt → 调用模型 → 输出结果。

这种链式结构可以构建复杂的业务逻辑流程。

4. Agents（智能代理）

Agent 允许模型根据当前问题自主决定调用哪个工具。例如：

当用户询问天气时，Agent 可以调用天气 API；

当用户询问计算问题时，Agent 可以调用 Python 计算工具。

这种机制使模型具备一定的“ 决策能力 ”。

5. Tools（工具）

工具是 Agent 可调用的外部功能，例如搜索引擎、数据库查询、计算器、API 请求等。工具通常以函数形式定义，并由 Agent 调用。

6. Memory（记忆模块）

为了支持多轮对话，LangChain
提供记忆机制。记忆可以保存历史对话内容，并在后续调用中注入到 Prompt 中。常见类型包括：

- ConversationBufferMemory

- ConversationSummaryMemory

- VectorStoreRetrieverMemory

二、RAG（检索增强生成）

RAG 是 LangChain 应用中最重要的模式之一。它的基本流程为：

第一步：将文档进行切分（Text Splitter）。

第二步：使用 Embedding 模型将文本向量化。

第三步：存储到向量数据库（如 Chroma、FAISS）。

第四步：用户提问时进行相似度检索。

第五步：将检索结果作为上下文拼接到 Prompt 中。

第六步：调用大模型生成最终答案。

这种方式解决了大模型“知识截止时间”和“幻觉”问题，使模型能够基于私有知识库回答问题。

三、向量数据库（Vector Store）

LangChain 支持多种向量数据库，例如：

- Chroma

- FAISS

- Pinecone

- Weaviate

向量数据库用于存储文本嵌入向量，并支持相似度搜索。

四、LangChain 的架构优势

1. 模块化设计

2. 高扩展性

3. 支持多模型

4. 易于集成外部系统

五、典型应用场景

1. 智能知识库系统

2. 企业问答机器人

3. 自动代码生成系统

4. 文档分析系统

5. 多工具自动决策系统

六、LangChain 与 LLM 应用开发趋势

随着大模型逐渐成为基础设施，应用开发逐渐从“ 单次问答 ”转向“ 流程编排 ”和“ 智能系统构建 ”。LangChain 正是为此而设计的框架。它不仅解决了模型调用问题，更重要的是解决了工程化问题，包括状态管理、数据接入、流程控制和工具调用。

总结来说，LangChain 是当前大模型应用开发的重要工具。通过掌握 Prompt Engineering、Chain 设计、Agent 架构和 RAG 技术，开发者可以构建真正可落地的 AI 系统。