LangChain: Complete Basics Guide

Introduction

LangChain is a Python framework to build applications using Large Language Models (LLMs). It helps connect LLMs with data sources, memory, agents, and tools for reasoning and automation.

Core Components

----

1. Chains
   - A chain is a sequence of calls to LLMs or other components.
   - Types include SimpleChain, SequentialChain, and Custom chains.
2. Prompts
   - Manage prompts easily using templates.
   - Fill in variables dynamically for flexible queries.
3. Memory
   - Store conversation context or state across calls.
   - Types include ConversationBufferMemory, KeyValueMemory, etc.
4. Agents
   - Agents can decide what action to take using LLMs.
   - Can call tools like APIs, calculators, or databases dynamically.
5. Tools
   - Functions or APIs that agents can use.
   - Examples: Google Search API, Calculator, Email sender.
6. Vector Stores & RAG
   - Store document embeddings for semantic search.
   - Examples: FAISS, Pinecone, Weaviate, Chroma.
   - RAG (Retrieval-Augmented Generation) retrieves relevant documents and combines with LLM for accurate answers.

Sample Workflow for RAG

----

1. Convert your documents into embeddings using models like Sentence Transformers.

2. Store embeddings in a vector database (FAISS, Pinecone).
3. When a query arrives, retrieve top-k similar documents.
4. Pass retrieved documents to LLM to generate an answer.

Sample Questions & Answers

Q1: What is LangChain used for?
A1: To build applications with LLMs that can reason, fetch data, and perform tasks using chains, prompts, and agents.

Q2: What is RAG?
A2: Retrieval-Augmented Generation; combines documents with LLMs to answer queries accurately.

Q3: Name a vector store supported by LangChain.
A3: FAISS, Pinecone, Weaviate, Chroma.

Q4: What is the purpose of embeddings in LangChain?
A4: Convert text into vectors to measure semantic similarity and enable document retrieval.

Q5: What are agents in LangChain?
A5: Agents are LLM-powered decision-makers that can call tools dynamically based on instructions or queries.

Conclusion

LangChain makes building advanced LLM-powered applications easier by providing modular components for reasoning, memory, document retrieval, and automation. Understanding its basics like chains, prompts, memory, agents, and vector stores is essential for developing robust applications.