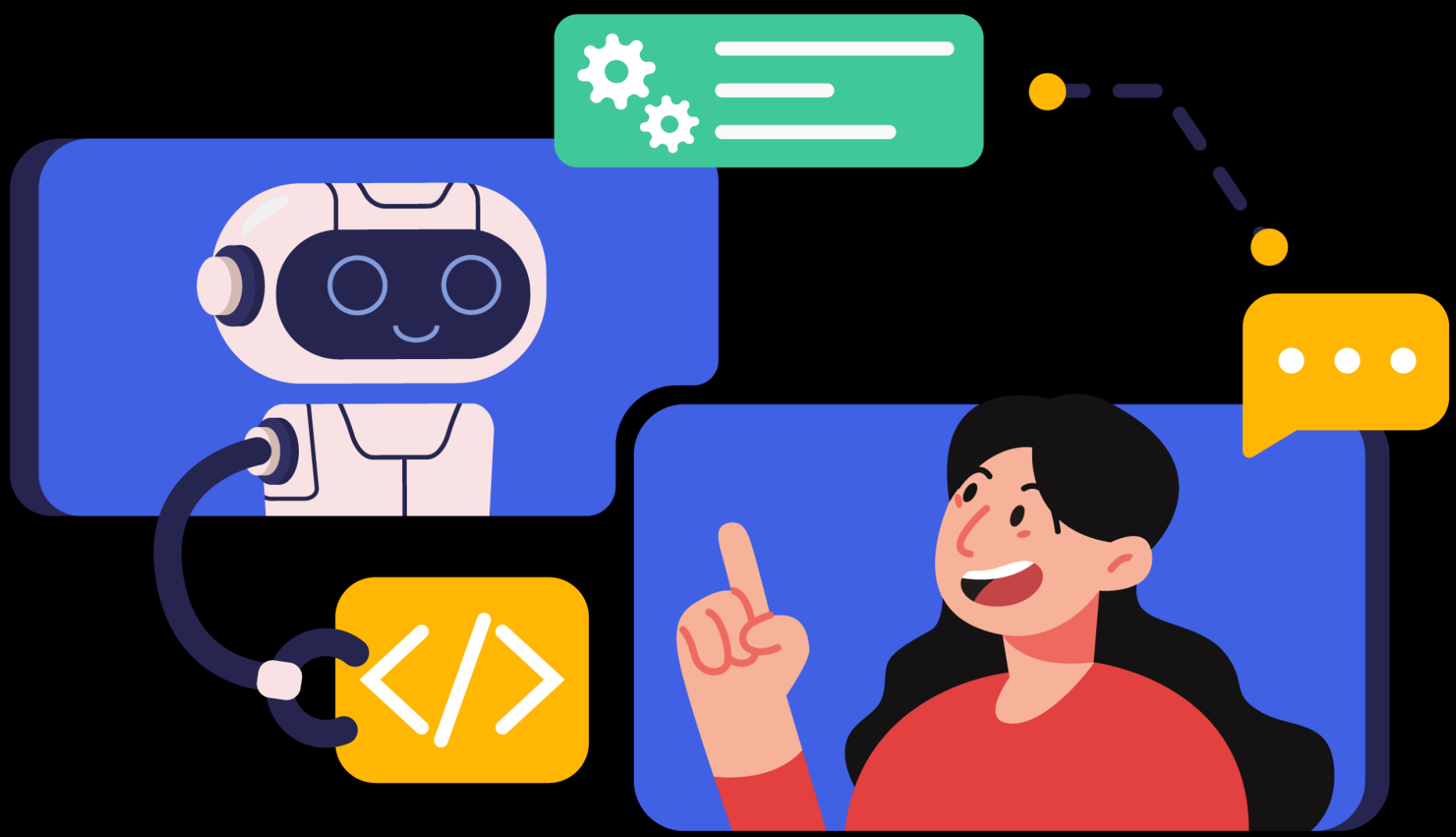
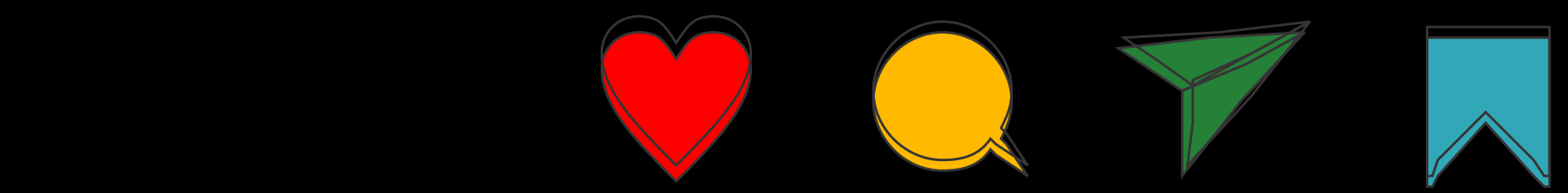


LangChain

CREATED BY

UDAYABHANU NAYAK



By 🎯Udayabhanu Nayak 💡

Lang Chain 🔗

LangChain is a framework for developing applications powered by language models.

- GitHub: https://github.com/hwchase17/langchain (https://github.com/hwchase17/langchain)
- Docs: https://python.langchain.com/en/latest/index.html (https://python.langchain.com/en/latest/index.html)

Overview:

- Installation
- LLMs
- Prompt Templates
- Chains
- Agents and Tools
- Memory
- Document Loaders
- Indexes

Setup the Environment

Setting up the environment involves configuring necessary variables and dependencies that LangChain and its components might need to function, such as API keys for accessing various language models like OpenAI's GPT or Hugging Face models.

Large Language Models (LLMs)

LLMs are advanced AI models capable of understanding and generating human-like text. In the context of LangChain, LLMs are the core component used for text generation, interpretation, and other natural language processing tasks. They can be from various providers like OpenAI, Hugging Face, etc.

Prompt Templates

Prompt Templates in LangChain are structured formats for generating prompts that are fed into LLMs. They ensure that the input to the model is consistent and structured, which can improve the quality and relevance of the model's output. They often include placeholders for dynamic content that can be filled in with specific details during runtime.

Chains

Chains in LangChain are sequences of operations that can include prompting LLMs, processing their outputs, and potentially feeding those outputs into subsequent steps. They enable complex workflows where the output of one step can inform the input of the next, allowing for sophisticated interactions with LLMs.

Agents and Tools

Agents in LangChain are entities that decide which actions to take based on user input, employing various tools to accomplish tasks. Tools are functionalities or integrations that perform specific actions, such as conducting a web search, performing mathematical calculations, accessing databases, etc. Agents use reasoning to determine which tools to apply to solve a given problem or answer a question.

Memory

Memory in LangChain allows the system to remember previous interactions, inputs, or generated outputs. This can be crucial for creating contextually aware applications that build upon previous exchanges, making interactions with LLMs more coherent and context-sensitive.

Document Loaders

Document Loaders are components in LangChain that facilitate the loading and processing of documents, such as PDFs or text files. They can extract content from these documents, making it accessible to LLMs for processing, analysis, or

Step 01: Installation

In [ ]:
```
!pip install langchain
```

In [ ]:
```
!pip install openai
```

Step 02: Setup the Environment

In [ ]:
```
import os

os.environ['OPENAI_API_KEY'] = "your_openai_api_key"
os.environ["HUGGINGFACEHUB_API_TOKEN"] = "your_huggingface_api_token
```

```
Replace your_openai_api_key and your_huggingface_api_token with
your actual API keys.
```

Step 03: Large Language Models

Initialize an OpenAI LLM with a higher temperature for more creative outputs:

In [ ]:
```
from langchain.llms import OpenAI

llm = OpenAI(temperature=0.9)
```

Step 04: Prompt Templates

Manage and optimize prompts effectively using LangChain's PromptTemplate:

In [ ]:
```
from langchain.prompts import PromptTemplate

prompt_template = PromptTemplate(
    input_variables=['cuisine'],
    template="I want to open a restaurant for {cuisine} food. Suggest
)
```

Step 05: Chains

Combine LLMs and prompt templates into multi-step workflows with LLMChain and SequentialChain:

In [ ]:
```python
from langchain.chains import LLMChain, SequentialChain

# Initialize an LLMChain with the LLM and prompt template
chain = LLMChain(llm=llm, prompt=prompt_template)

# For sequential chains, combine multiple chains
sequential_chain = SequentialChain(chains=[chain1, chain2])
```

Step 06: Agents and Tools

Agents decide which actions to take based on user input, utilizing various tools like Google Search, Wikipedia, and math operations:

In [ ]:
```python
from langchain.agents import initialize_agent, load_tools
from langchain.agents import AgentType

# Load required tools
tools = load_tools(["serpapi", "llm-math"], llm=llm)

# Initialize the agent
agent = initialize_agent(tools, llm, agent=AgentType.ZERO_SHOT_REACT_
```

Step 07: Memory

Incorporate memory into chains to remember past interactions, enhancing the contextuality of responses:

In [ ]:
```python
from langchain.memory import ConversationBufferMemory

memory = ConversationBufferMemory()
chain = LLMChain(llm=llm, prompt=prompt_template, memory=memory)
```

Step 08: Document Loaders

Load and extract information from documents, such as PDFs, with document loaders like PyPDFLoader:

In [ ]:
```
from langchain.document_loaders import PyPDFLoader

loader = PyPDFLoader("path_to_your_pdf_file.pdf")
pages = loader.load()
```

Setup the Environment

Setting up the environment involves configuring necessary variables and dependencies that LangChain and its components might need to function, such as API keys for accessing various language models like OpenAI's GPT or Hugging Face models.

Large Language Models (LLMs)

LLMs are advanced AI models capable of understanding and generating human-like text. In the context of LangChain, LLMs are the core component used for text generation, interpretation, and other natural language processing tasks. They can be from various providers like OpenAI, Hugging Face, etc.

Prompt Templates

Prompt Templates in LangChain are structured formats for generating prompts that are fed into LLMs. They ensure that the input to the model is consistent and structured, which can improve the quality and relevance of the model's output. They often include placeholders for dynamic content that can be filled in with specific details during runtime.

Chains

Chains in LangChain are sequences of operations that can include prompting LLMs, processing their outputs, and potentially feeding those outputs into subsequent steps. They enable complex workflows where the output of one step can inform the input of the next, allowing for sophisticated interactions with LLMs.

Agents and Tools

Agents in LangChain are entities that decide which actions to take based on user input, employing various tools to accomplish tasks. Tools are functionalities or integrations that perform specific actions, such as conducting a web search, performing mathematical calculations, accessing databases, etc. Agents use reasoning to determine which tools to apply to solve a given problem or answer a question.

Memory

Memory in LangChain allows the system to remember previous interactions, inputs, or generated outputs. This can be crucial for creating contextually aware applications that build upon previous exchanges, making interactions with LLMs more coherent and context-sensitive.

Document Loaders

Document Loaders are components in LangChain that facilitate the loading and processing of documents, such as PDFs or text files. They can extract content from

Thank You

¶