Step 1: Install LangChain and Dependencies

pip install langchain openai

pip install tiktoken python-dotenv

Step 2: Set Up Your OpenAI API Key

Create a `.env` file in your project folder:

OPENAI_API_KEY=your_openai_key_here

```
import os
from dotenv import load_dotenv

load_dotenv()
openai_api_key = os.getenv("OPENAI_API_KEY")
```

Step 3: Use a Simple LLM Chain

This is the most basic use-case: prompt → LLM → response.

```
from langchain.llms import OpenAI
from langchain.prompts import PromptTemplate
from langchain.chains import LLMChain

llm = OpenAI(temperature=0.7, openai_api_key=openai_api_key)

prompt = PromptTemplate(
    input_variables=["topic"],
    template="Write a short story about {topic}"
)

chain = LLMChain(llm=llm, prompt=prompt)

response = chain.run("a time-traveling dog")
print(response)
```

Step 4: Use Memory (Remember Conversation)

pip install langchain[hub]

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

memory = ConversationBufferMemory()

conversation = ConversationChain(
    llm=llm,
    memory=memory,
    verbose=True
)

print(conversation.predict(input="Hi there!"))
print(conversation.predict(input="Who am I?"))
```

Step 5: Use Documents (Load and Ask)

pip install unstructured chromadb

```
from langchain.document_loaders import TextLoader
from langchain.vectorstores import Chroma
from langchain.embeddings import OpenAIEmbeddings
from langchain.chains import RetrievalQA

# Load data
loader = TextLoader("example.txt")
docs = loader.load()

# Embed and store in vector DB
embeddings = OpenAIEmbeddings()
db = Chroma.from_documents(docs, embeddings)

# Retrieval chain
retriever = db.as_retriever()
qa = RetrievalQA.from_chain_type(llm=llm, retriever=retriever)

print(qa.run("What is this text about?"))
```

Step 6: Experiment with Tools (Agent)

LangChain supports **tools** like search, calculator, and even calling Python code.

Install:

pip install langchain-agent openai serpapi

Example: Agent that uses tools like Google Search.

```
from langchain.agents import load_tools, initialize_agent, AgentType

tools = load_tools(["serpapi", "llm-math"], llm=llm)
agent = initialize_agent(tools, llm, agent=AgentType.ZERO_SHOT_REACT_DESCRIPTION)

agent.run("What is the square root of 4900 and current weather in Tokyo?")
```

Give me Full Langchain Code to read a text file and ask question and answer from it

Here's a **complete LangChain Python script** to read a text file, create a vector store from it, and allow the user to ask questions about the content using OpenAI's models.

Requirements

Make sure you have the following Python packages installed:

```
pip install langchain openai chromadb tiktoken
```

Save this as `qa_from_file.py`:-

```python
import os
from langchain.embeddings import OpenAIEmbeddings
from langchain.vectorstores import Chroma
from langchain.text_splitter import CharacterTextSplitter
from langchain.document_loaders import TextLoader
from langchain.chat_models import ChatOpenAI
from langchain.chains import RetrievalQA
from langchain.schema import HumanMessage

# Set your OpenAI API key
os.environ["OPENAI_API_KEY"] = "your_openai_api_key_here"

# === STEP 1: Load the text file ===
file_path = "your_file.txt"  # Replace with your text file path
loader = TextLoader(file_path)
documents = loader.load()

# === STEP 2: Split the text into chunks ===
text_splitter = CharacterTextSplitter(chunk_size=1000, chunk_overlap=100)
docs = text_splitter.split_documents(documents)

# === STEP 3: Create vector store using Chroma ===
embedding = OpenAIEmbeddings()
vectorstore = Chroma.from_documents(docs, embedding)
```

```python
# === STEP 4: Create Retrieval-based QA chain ===
retriever = vectorstore.as_retriever(search_kwargs={"k": 3})
qa_chain = RetrievalQA.from_chain_type(
    llm=ChatOpenAI(model_name="gpt-3.5-turbo"),
    chain_type="stuff",
    retriever=retriever,
    return_source_documents=True
)

# === STEP 5: Ask questions in a loop ===
print("Ask questions about the file. Type 'exit' to quit.")
while True:
    query = input("\nYour question: ")
    if query.lower() in ["exit", "quit"]:
        break
    result = qa_chain({"query": query})
    print("\nAnswer:", result["result"])
```

Replace with your values:

- Set your OpenAI API key: `os.environ["OPENAI_API_KEY"] = "your_openai_api_key_here"`
- Replace `your_file.txt` with the actual path to your text file.

Example

If your `your_file.txt` contains:

```
The Eiffel Tower is located in Paris and was built in 1889 for the World's Fair.
```

Then you can ask:

```
Your question: When was the Eiffel Tower built?
Answer: The Eiffel Tower was built in 1889 for the World's Fair.
```

Now with

Here's the **full working LangChain script** using **FAISS** instead of Chroma, which avoids native DLL errors and works well on most systems.

✅ Install Dependencies

```
pip install langchain openai faiss-cpu tiktoken
```

✅ Full Script: `qa_from_file_faiss.py`

```python
import os
from langchain.embeddings import OpenAIEmbeddings
from langchain.vectorstores import FAISS
from langchain.text_splitter import CharacterTextSplitter
from langchain.document_loaders import TextLoader
from langchain.chat_models import ChatOpenAI
from langchain.chains import RetrievalQA

# === STEP 0: Set your OpenAI API key ===
os.environ["OPENAI_API_KEY"] = "your_openai_api_key_here"

# === STEP 1: Load the text file ===
file_path = "your_file.txt"  # Replace with your actual text file
loader = TextLoader(file_path)
documents = loader.load()

# === STEP 2: Split the text into chunks ===
text_splitter = CharacterTextSplitter(chunk_size=1000, chunk_overlap=100)
docs = text_splitter.split_documents(documents)

# === STEP 3: Embed documents and build FAISS index ===
embedding = OpenAIEmbeddings()
vectorstore = FAISS.from_documents(docs, embedding)

# === STEP 4: Build QA chain using FAISS retriever ===
retriever = vectorstore.as_retriever(search_kwargs={"k": 3})
```

```
qa_chain = RetrievalQA.from_chain_type(
    llm=ChatOpenAI(model_name="gpt-3.5-turbo"),
    chain_type="stuff",
    retriever=retriever
)

# === STEP 5: Interactive Q&A ===
print("Ask questions about the text file. Type 'exit' to quit.")

while True:
    query = input("\nYour question: ")
    if query.lower() in ["exit", "quit"]:
        break
    result = qa_chain.run(query)
    print("\nAnswer:", result)
```

Fixed and Updated Script (FAISS + Community Imports)

First, install updated packages:-

```
pip install -U langchain langchain-community langchain-openai openai faiss-cpu
```

---

qa_from_file_faiss_fixed.py:-

```python
import os
from langchain_community.document_loaders import TextLoader
from langchain.text_splitter import CharacterTextSplitter
from langchain_openai import OpenAIEmbeddings, ChatOpenAI
from langchain_community.vectorstores import FAISS
from langchain.chains import RetrievalQA

# Set your OpenAI API key
os.environ["OPENAI_API_KEY"] = "your_openai_api_key_here"

# Load your text file
file_path = "your_file.txt"  # Replace with your text file
loader = TextLoader(file_path)
documents = loader.load()

# Split text into chunks
text_splitter = CharacterTextSplitter(chunk_size=1000, chunk_overlap=100)
docs = text_splitter.split_documents(documents)

# Embed and store with FAISS
embedding = OpenAIEmbeddings()  # uses text-embedding-ada-002 by default
vectorstore = FAISS.from_documents(docs, embedding)

# Create QA chain
retriever = vectorstore.as_retriever(search_kwargs={"k": 3})
qa_chain = RetrievalQA.from_chain_type(
    llm=ChatOpenAI(model="gpt-3.5-turbo"),
    chain_type="stuff",
    retriever=retriever
)

# Ask questions interactively
print("Ask questions about the file. Type 'exit' to quit.")
```

```
while True:
    query = input("\nYour question: ")
    if query.lower() in ["exit", "quit"]:
        break
    result = qa_chain.run(query)
    print("\nAnswer:", result)
```