**Setup and basic details roadmap**

Here are some of the **most well-known frameworks for Generative AI (GenAI)**, grouped by their main ecosystem and usage:

**1. Python-based Deep Learning Frameworks**

These are the core engines behind most GenAI applications.

* **TensorFlow** (by Google)
  + Widely used for training and deploying large-scale AI models, including text, image, and speech generation.
  + Has Keras as a high-level API for ease of use.
* **PyTorch** (by Meta)
  + Favored for research and rapid prototyping of GenAI models.
  + Hugely popular in the open-source community and supports libraries like Hugging Face Transformers.

**2. Specialized GenAI Model Frameworks**

Frameworks built specifically for **Large Language Models (LLMs), diffusion models, and multimodal AI**:

* **Hugging Face Transformers**
  + Pretrained models for text, vision, audio, and multimodal AI.
  + Includes LLaMA, GPT, BLOOM, Falcon, and Stable Diffusion models.
* **LangChain**
  + Designed for building LLM-powered applications with context management, chaining prompts, and connecting to tools/APIs.
* **LlamaIndex (formerly GPT Index)**
  + Helps connect LLMs with private or custom data sources.
* **Diffusers (by Hugging Face)**
  + For image/video generation using diffusion models like Stable Diffusion.

**3. Enterprise & Production-Oriented GenAI Frameworks**

These help scale GenAI into real applications:

* **OpenAI API**
  + Provides access to GPT models (text), DALL·E (image), Whisper (audio).
* **Cohere AI**
  + NLP models optimized for retrieval-augmented generation (RAG).
* **Anthropic Claude API**
  + Focus on safe and explainable conversational AI.
* **Amazon Bedrock**
  + Fully managed service for running foundation models from multiple providers.

**4. Vector Database & RAG Tooling**

For Retrieval-Augmented Generation pipelines:

* **Pinecone**, **Weaviate**, **Milvus** – for storing & searching embeddings.
* **Chroma** – open-source vector DB often used with LangChain.

**5. Other Notable Frameworks**

* **Fast.ai** – high-level wrapper on PyTorch for quick experimentation.
* **DeepSpeed** (by Microsoft) – for efficient large model training.
* **vLLM** – optimized inference engine for LLMs.

If you want, I can prepare a **visual diagram** mapping these frameworks into **Training**, **Fine-tuning**, **Inference**, and **Application-layer** categories so it’s easy to remember. That way you’ll see where each GenAI framework fits in the lifecycle.

**Starting roadmap** for LangChain.

**1. What LangChain Is**

LangChain is a **framework** for building applications using **large language models (LLMs)**.  
It helps with:

* **Prompt engineering** (managing and chaining prompts)
* **Connecting to external data** (databases, APIs, documents)
* **Retrieval-Augmented Generation (RAG)**
* **Managing conversational memory**

Think of it as a toolkit for going from *"I have an LLM"* → *"I have a working AI app."*

**2. Prerequisites**

Before starting, you should have:

* **Python basics** (functions, imports, pip)
* Some **knowledge of OpenAI API** or another LLM provider (Hugging Face, Cohere, Anthropic)
* Installed Python 3.9+

**3. Installation**

pip install langchain

pip install langchain-openai

pip install openai # if using OpenAI models

pip install chromadb # for local vector store

**4. Your First LangChain App**

Example: **A simple Q&A bot using OpenAI + LangChain**

from langchain\_openai import OpenAI

from langchain.prompts import PromptTemplate

from langchain.chains import LLMChain

# Create LLM

llm = OpenAI(temperature=0.7)

# Create a prompt template

template = "You are a helpful assistant. Answer the following question:\n{question}"

prompt = PromptTemplate(input\_variables=["question"], template=template)

# Create chain

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

# Run the chain

response = chain.run("What is the capital of France?")

print(response)

✅ Output: Paris

**5. Core Concepts in LangChain**

* **LLM** → The AI model (OpenAI GPT, Hugging Face, etc.)
* **Prompt Templates** → Reusable prompt structures
* **Chains** → Connect multiple steps (e.g., query DB → summarize → answer)
* **Agents** → LLMs that decide which tool to use next (like calling APIs)
* **Memory** → Stores chat history
* **Vector Stores** → Store & search embeddings for RAG

**6. Next Step – Adding RAG**

You can make your LLM answer **from your own data** instead of the public internet:

1. Convert your documents into **embeddings**
2. Store in a **vector DB** (e.g., Chroma, Pinecone)
3. Use LangChain’s RetrievalQA to fetch relevant info before answering

Example:

from langchain.document\_loaders import TextLoader

from langchain.vectorstores import Chroma

from langchain.embeddings.openai import OpenAIEmbeddings

from langchain.chains import RetrievalQA

# Load docs

loader = TextLoader("my\_data.txt")

docs = loader.load()

# Create embeddings and vector store

embeddings = OpenAIEmbeddings()

db = Chroma.from\_documents(docs, embeddings)

# Create retrieval chain

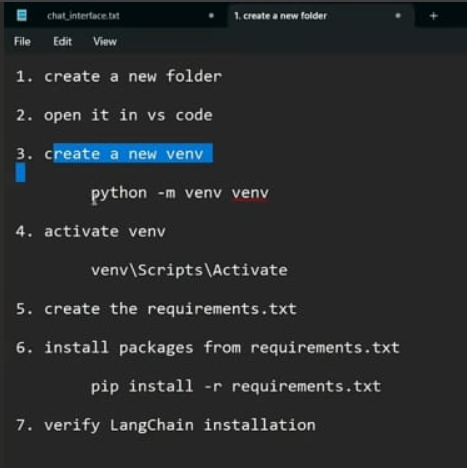
retriever = db.as\_retriever()

qa = RetrievalQA.from\_chain\_type(llm=llm, retriever=retriever)

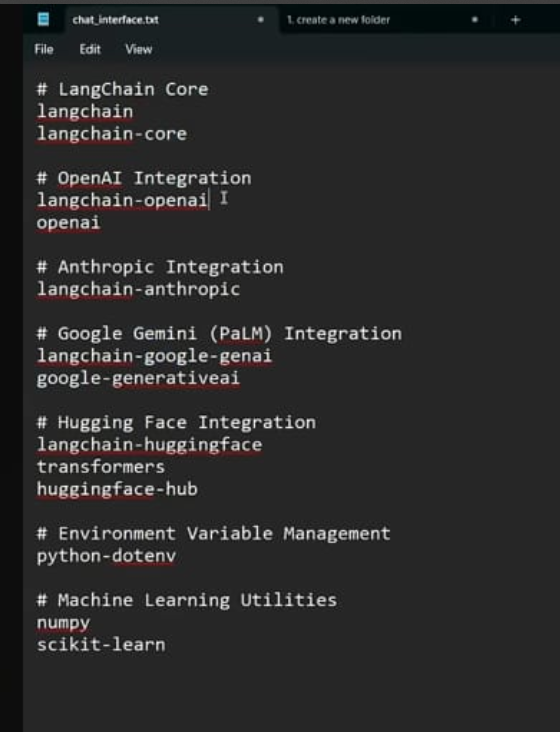
print(qa.run("What does my document say about sales in 2023?"))

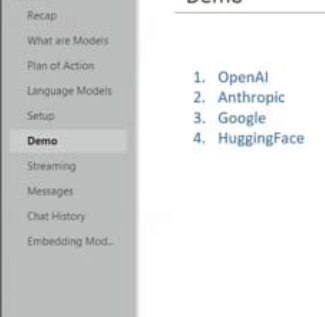
**7. Learning Resources**

* 📚 LangChain Docs - https://python.langchain.com/docs/introduction/
* 🎥 YouTube Tutorials (search: "LangChain RAG tutorial") - <https://www.youtube.com/playlist?list=PLKnIA16_RmvaTbihpo4MtzVm4XOQa0ER0>
* <https://www.youtube.com/watch?v=X0btK9X0Xnk>
* <https://www.youtube.com/watch?v=J5_-l7WIO_w>
* 🛠 Playground: LangChain Hub - <https://smith.langchain.com/>



pip install -r requirements.txt -> where requirements.txt given below





**ModuleNotFoundError**: Module langchain\_community.document\_loaders not found. Please install langchain-community to access this module. You can install it using `pip install -U langchain-community`

After completing RAG (text to text model ) we can go with below

**Roadmap for Creative AI (2025 edition)**

**Step 1. Foundation in Diffusion Models**

* Learn how **diffusion models** work (denoising process, latent diffusion).
* Framework:
  + **🤗 Hugging Face Diffusers** → 🔥 *most widely used open-source toolkit* for:
    - Text → Image (Stable Diffusion, SDXL)
    - Image → Image (img2img, inpainting, outpainting)
    - Text → Video (with AnimateDiff, Zeroscope, Stable Video Diffusion)
  + Other libs: ComfyUI (no-code workflow for artists).

👉 **Why:** Diffusers give you hands-on with pipelines for image/video/audio generation.

**Step 2. HD Image Generation**

* **Frameworks:**
  + Stable Diffusion XL (SDXL) (high-quality, photorealistic images).
  + ControlNet → add control over pose, edges, depth maps.
* Learn:
  + Text prompts, prompt engineering.
  + Fine-tuning with **LoRA** for company-specific styles.
* Example use: Marketing banners, product design mockups.

**Step 3. Text → Video & Image → Video**

* **Frameworks:**
  + Hugging Face Diffusers + **Stable Video Diffusion (SVD)**.
  + Pika Labs, RunwayML, Kaiber (popular SaaS, but can be inspiration).
* Learn to:
  + Convert single image → motion (AnimateDiff).
  + Multi-frame consistency → story clips.
* Example use: **Promo ads, explainer videos, cinematic reels**.

**Step 4. Teaching Audio / Dubbing**

* **Speech synthesis & dubbing frameworks:**
  + Coqui TTS (open-source, multi-speaker cloning).
  + OpenAI TTS (new GPT models for ultra-realistic voices).
  + XTTS (cross-lingual text-to-speech).
* Learn:
  + Voice cloning from samples.
  + Lip-sync for videos (Wav2Lip, SadTalker).
* Example use: **Educational dubbing in different languages.**

**Step 5. Image Editing & Enhancement**

* **Tasks:** inpainting (fill missing parts), super-resolution (HD upscale).
* **Frameworks:**
  + StableSR (super-resolution for SD).
  + Real-ESRGAN (upscale images to 4K).
  + PhotoMaker (face swap, identity preservation).
* Example: e-commerce images, movie posters.

**Step 6. Advanced Video Generation**

* **Text → Video Research models (2024-25):**
  + **OpenAI Sora** (🔥 upcoming game-changer, text → full video).
  + **ModelScope T2V** (open-source, Chinese researchers).
  + **Gen-2 (Runway)** → story-level video generation.
* Example use: **Storytelling, short films, marketing reels.**

**Step 7. Integration & Deployment**

* **UI frameworks:**
  + Streamlit → easy demo apps.
  + Gradio → great for model playgrounds.
* **Deployment:**
  + Hugging Face Spaces (free hosting).
  + Docker + Cloud (AWS, GCP, Azure).
  + GPU required (NVIDIA A100 / RTX 4090 recommended).

**✅ Suggested Learning Order**

1. **Diffusers basics** → Text2Image, Image2Image.
2. Add **ControlNet + LoRA** for style control.
3. Move to **Video** → AnimateDiff, Stable Video Diffusion.
4. Explore **Audio dubbing** (Coqui TTS, OpenAI TTS).
5. Work on **HD enhancement** (Real-ESRGAN).
6. Build **UI app** → Streamlit/Gradio chatbot + creative studio.

**📌 Example Real-World Applications**

* **E-commerce** → Auto-generate product images, 360° videos.
* **Education** → Auto-dub teaching videos into local languages.
* **Marketing** → Generate ad videos & posters from brand guidelines.
* **Film/Animation** → Storyboarding from text + script → video pipeline.

👉 My advice:  
If your **main goal is HD image/video/story generation** → **start with Hugging Face Diffusers**. It’s the **TensorFlow of creative AI** — everything else plugs into it.

Would you like me to **pick one use-case (say education dubbing or e-commerce ads)** and build a **step-by-step Streamlit production app with Diffusers + TTS + Video** so you can see it all working together?