python™ Libraries

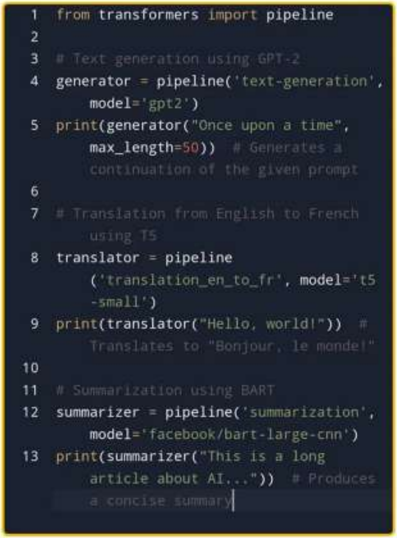
FOR **BUILDING** Generative **AI Apps**

1. Transformers

Your AI toolbox, pre-packed with powerful language models.

- Simplifies access to state-of-the-art models like GPT, BERT, T5, etc.
- Provides pipelines for common NLP tasks: text generation, translation, summarization.
- Abstracts away complex model architectures, making them user-friendly.
- Enables easy fine-tuning on custom datasets for specific needs

1. Transformers: example



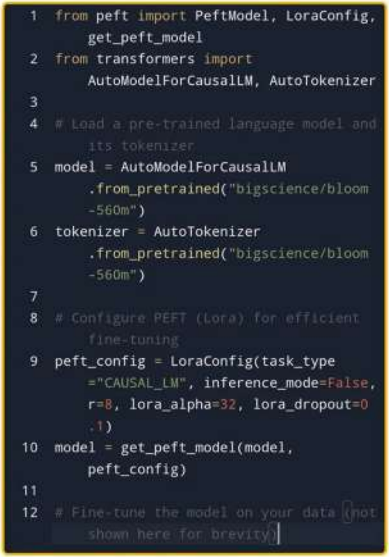
```python
from transformers import pipeline

# Text generation using GPT-2
generator = pipeline('text-generation', model='gpt2')
print(generator("Once upon a time", max_length=50))  # Generates a continuation of the given prompt

# Translation from English to French using T5
translator = pipeline('translation_en_to_fr', model='t5-small')
print(translator("Hello, world!"))  # Translates to "Bonjour, le monde!"

# Summarization using BART
summarizer = pipeline('summarization', model='facebook/bart-large-cnn')
print(summarizer("This is a long article about AI..."))  # Produces a concise summary
```

2. PEFT

Efficiency booster for your AI models

- Fine-tunes large language models with less computational resources
- Leverages techniques like LoRA (Low-Rank Adaptation) to reduce memory footprint
- Makes it feasible to adapt powerful models on commodity hardware
- Accelerates training and inference times

2. PEFT: Example



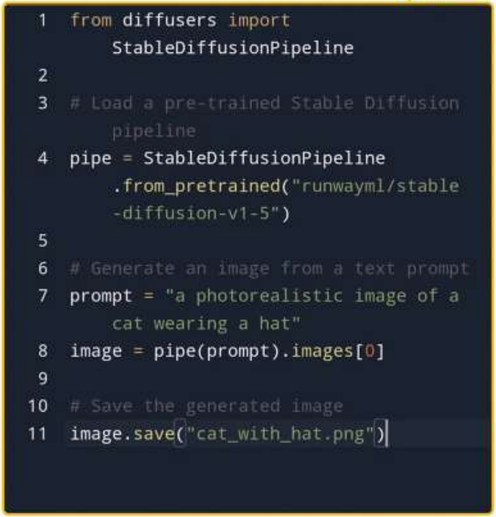
```python
from peft import PeftModel, LoraConfig, get_peft_model
from transformers import AutoModelForCausalLM, AutoTokenizer

# Load a pre-trained language model and its tokenizer
model = AutoModelForCausalLM.from_pretrained("bigscience/bloom-560m")
tokenizer = AutoTokenizer.from_pretrained("bigscience/bloom-560m")

# Configure PEFT (LoRA) for efficient fine-tuning
peft_config = LoraConfig(task_type="CAUSAL_LM", inference_mode=False, r=8, lora_alpha=32, lora_dropout=0.1)
model = get_peft_model(model, peft_config)

# Fine-tune the model on your data (not shown here for brevity)
```

3. Diffusers:

Your gateway to generating images and other media

- Implements various diffusion models for high-quality image generation
- Supports text-to-image, image-to-image, and inpainting tasks
- Provides pre-trained models and easy customization options
- Makes creating visually appealing content accessible

3. Diffusers: Example



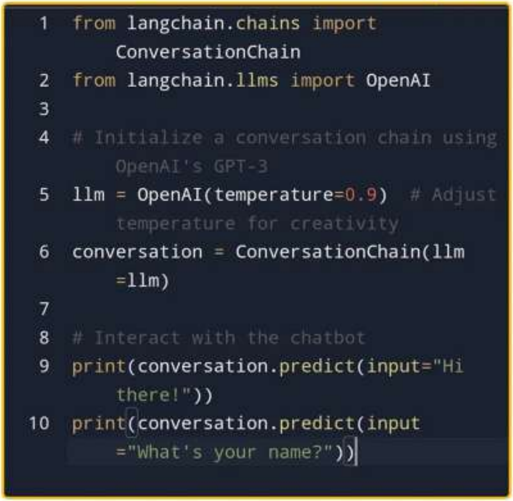
```python
from diffusers import StableDiffusionPipeline

# Load a pre-trained Stable Diffusion pipeline
pipe = StableDiffusionPipeline.from_pretrained("runwayml/stable-diffusion-v1-5")

# Generate an image from a text prompt
prompt = "a photorealistic image of a cat wearing a hat"
image = pipe(prompt).images[0]

# Save the generated image
image.save("cat_with_hat.png")
```

4. LangChain

The architect for building conversational AI

- Chains together multiple language models and tools for complex tasks
- Manages memory and context for multi-turn conversations
- Integrates with external data sources for knowledge-grounded responses
- Provides a framework for building chatbots, question-answering systems, etc

4. LangChain: Example



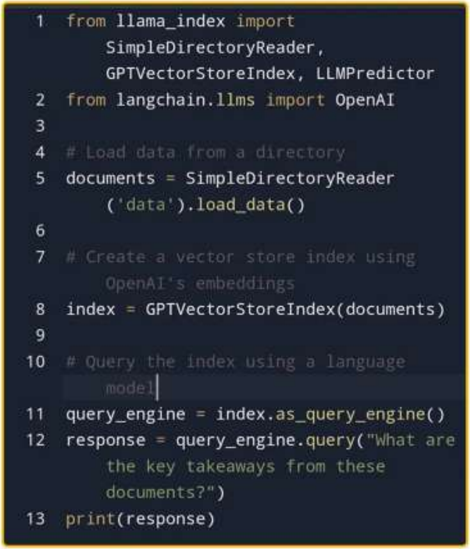
```python
from langchain.chains import ConversationChain
from langchain.llms import OpenAI

# Initialize a conversation chain using OpenAI's GPT-3
llm = OpenAI(temperature=0.9)  # Adjust temperature for creativity
conversation = ConversationChain(llm=llm)

# Interact with the chatbot
print(conversation.predict(input="Hi there!"))
print(conversation.predict(input="What's your name?"))
```

5. LlamaIndex

Your search engine for large language models

- Indexes and structures your data for efficient retrieval
- Enables querying large language models even if they can't fit in memory
- Supports various data sources and indexing techniques
- Helps build applications that leverage knowledge from vast datasets

5. LlamaIndex: Example



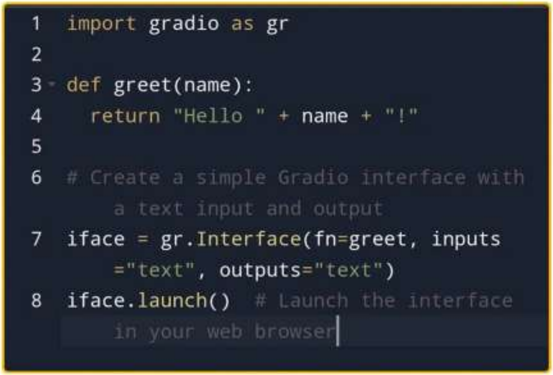
```python
from llama_index import SimpleDirectoryReader, GPTVectorStoreIndex, LLMPredictor
from langchain.llms import OpenAI

# Load data from a directory
documents = SimpleDirectoryReader('data').load_data()

# Create a vector store index using OpenAI's embeddings
index = GPTVectorStoreIndex(documents)

# Query the index using a language model
query_engine = index.as_query_engine()
response = query_engine.query("What are the key takeaways from these documents?")
print(response)
```

6. Chat UI & Gradio

(See the LangChain example above for a basic chatbot implementation. Enhance it with Chat UI for a visually appealing and interactive interface)

- Gradio: Democratize your AI models with user-friendly web interfaces
- Creates interactive demos for your machine learning models
- Supports various input and output types (text, images, audio, etc.)
- Requires minimal coding to get started
- Makes it easy to share your models with others

6. Chat UI & Gradio: Example



```python
import gradio as gr

def greet(name):
  return "Hello " + name + "!"

# Create a simple Gradio interface with
    a text input and output
iface = gr.Interface(fn=greet, inputs
    ="text", outputs="text")
iface.launch()  # Launch the interface
    in your web browser
```

7. OpenLLM & OpenAI Python

OpenLLM: Streamline deployment and management of large language models

- Supports various open-source LLMs and frameworks
- Provides tools for quantization, optimization, and serving
- Simplifies the process of deploying models to production environments

- OpenAI Python: Your bridge to OpenAI's powerful API
    - Access a wide range of AI models and capabilities
    - Generate text, images, code, and more
    - Integrates seamlessly with other Python libraries

(Refer to the LangChain and PEFT examples for how to utilize OpenAI's models and fine-tune them for specific tasks. OpenLLM helps deploy and manage such models effectively)