Building OpenAI API-Based Python GenAI Applications

A Guide to the Deitel Videos on the O'Reilly Online Learning Subscription Site

[Estimated reading time for this document: 20 minutes. Estimated time to watch the linked videos and run the Python code: 4.5 hours. **Please share this guide with your friends and colleagues who might find it helpful.**]

This comprehensive guide overviews **Lesson 18, Building OpenAI API-Based Python Generative AI Applications**, from my **Python Fundamentals video course** on O'Reilly Online Learning. The lesson focuses on building Python apps using **OpenAI's generative AI (genAI) APIs**. This document guides you through my **hands-on code examples** and provides **Try It exercises** for experimenting with the APIs. You'll leverage the OpenAI APIs to create intelligent, multimodal apps that understand, generate and manipulate text, code, images, audio and video content.

This guide links you to **31 videos totaling about 3.5 hours** in my **Python Fundamentals** video course in which I present fully coded Python genAI apps that use the OpenAI APIs to

- summarize documents
- determine text's sentiment (positive, neutral or negative)
- use vision capabilities to generate accessible image descriptions
- translate text among spoken languages
- generate and manipulate Python code
- extract from text named entities, such as people, places, organizations, dates, times, events, products, …
- transcribe speech to text
- synthesize speech from text, using one of OpenAI's 11 voices and prompts that control style and tone
- create original images
- transfer art styles to images via text prompts
- transfer styles between images
- generate video closed captions
- filter inappropriate content
- generate and remix videos (under development at the time of this writing—uses OpenAI's recently released Sora 2 API)
- build agentic AI apps (under development at the time of this writing—uses OpenAI's recently released AgentKit)

The remaining videos overview concepts and present genAI prompt and coding exercises you can use to dig deeper into the covered topics.

Videos:

- Building API-Based Python GenAI Applications: Overview (8m 1s)
- This Lesson Is Under Development (2m 25s)—Discusses my plans for enhancing this lesson with new examples as I continually create them.

How I Formed This Guide

I created the initial draft of this guide using **five genAIs—OpenAI's ChatGPT**, **Google's Gemini**, **Anthropic's Claude**, **Microsoft's Copilot** and **Perplexity**. I provided each with

- a detailed prompt,
- a Chapter 18 draft from our forthcoming **Python for Programmers, 2/e** product suite and
- a list of the video titles and links you'll find in this guide.

I then asked Claude to summarize the results, and tuned the summary to create this blog post.

Contacting Me with Questions

The OpenAI APIs are evolving rapidly. If you run into problems while working through the examples or find that something has changed, check the Deitel blog or send an email to paul@deitel.com.

Downloading the Code

Go to the Python Fundamentals, 2/e GitHub Repository to get the source code that accompanies the videos referenced in this guide. The OpenAI API examples are located in the `examples/18` folder. **Book chapter numbers and corresponding video lesson numbers are subject to change while the second edition of our Python product suite is under development.**

Suggested Learning Workflow

If you watch the videos, you'll get a code-example-rich intro to programming with the OpenAI APIs. To learn how to work with various aspects of the OpenAI APIs, I suggest that you:

- **Watch** the video for each example.
- **Run** the provided Python code.
- **Complete** the **"Try It" coding challenges**.
- **Experiment** by creatively combining APIs (e.g., transcribe audio then translate, or generate images with accessibility descriptions).

Key Takeaways

This comprehensive guide and the corresponding videos present practical skills for harnessing the power of **OpenAI's genAI APIs**. You'll:

- Master OpenAI APIs in Python and perform **creative prompt engineering**.
- Build complete, functional, multimodal apps that create and manipulate text, code, images, audio and video.
- Implement responsible **accessibility** and **content moderation** practices.

Caution: GenAIs make mistakes and even "hallucinate." You should always verify their outputs.

Introduction

In this video, I discuss the required official **`openai` Python module**, **OpenAI's fee-based API model**, and monitoring and managing API usage costs.

Video: Introduction (6m)

OpenAI APIs

Here, I overview the OpenAI APIs and models I'll demo in this lesson.

Video: OpenAI APIs (2m 29s)

OpenAI Documentation: API Reference

Try It: Browse the OpenAI API documentation and review the API subcategories.

Try It: Prompt genAIs for an overview of responsible AI practices.

OpenAI Developer Account and API Key

Here, you'll learn how to create your OpenAI developer account, generate an API key and securely store it in an environment variable. This required setup step will enable your apps to authenticate with OpenAI so they can make API calls. You'll understand best practices for securing your API key. **The OpenAI API is a paid service. If, for the moment, you do not want to code with paid APIs, reading this document, watching the videos and reading the code is still valuable.**

Video: OpenAI Developer Account and API Key (8m 45s)

OpenAI Documentation: Account Setup, API Keys

Try It: Create your OpenAI developer account, generate your first API key and store it securely using an environment variable.

Text Generation Via the Responses API

My text-generation examples introduce the **Responses API**, OpenAI's primary text-generation interface. I show how to structure prompts, configure parameters, invoke the API and interpret responses. This API enables sophisticated conversational AI applications and is the foundation for many text-based genAI tasks.

Video: Text Generation Via the Responses API: Overview (4m 58s)

OpenAI Documentation: Text Generation Guide

Text Summarization

Here, I use OpenAI's **natural language understanding** capabilities to condense lengthy text into concise summaries. This example covers crafting **summarization prompts** and controlling summary length and style. Text summarization is invaluable for efficiently processing large documents, articles and reports.

Videos:

- Text Summarization (13m 16s)
- Text Summarization: GenAI Prompting Exercises (3m 12s)

Try It: Create a summarization tool that takes a long article and generates brief, moderate and detailed summaries.

Sentiment Analysis

This example uses OpenAI's **natural language understanding** capabilities to analyze text's **emotional tone and sentiment**. It **classifies text** as **positive**, **negative** or **neutral**, and has model **explain how it came to that conclusion**.

Video: Sentiment Analysis (4m 18s)

Try It: Build a sentiment analyzer that classifies the sentiment of customer reviews and asks the genAI model to provide a confidence score from 0.0 to 1.0 for each, indicating the likelihood that the classification is correct. Confidence scores closer to 1.0 are more likely to be correct.

Vision: Accessible Image Descriptions

In this example, I show OpenAI's **vision capabilities** for analyzing images and use them to **generate detailed, contextual descriptions, making them accessible to users with visual impairments.** You'll understand how to optimize prompts for description styles and detail levels.

Video: Vision: Accessible Image Descriptions (18m 42s)

OpenAI Documentation: Images and Vision Guide

Try It: Create an application that takes URLs for various images and generates both brief and comprehensive accessibility descriptions suitable for screen readers.

Language Detection and Translation

In this example, I use OpenAI's **multilingual capabilities** to **auto-detect what language text is written in** and **translate text to other spoken languages**.

Videos:

- Language Detection and Translation (4m 16s)
- Language Detection and Translation: GenAI Prompting Exercises (1m 2s)

Try It: Build a translation tool that detects the input language and translates to a target language, preserving tone and context.

Code Generation

Discover how AI can generate, explain, and debug code across multiple programming languages. The first video covers **code generation**, **understanding AI-generated code quality**, and **using AI as a coding assistant**. In the second video, I discuss how genAIs can assist you with coding, including code generation, testing, debugging, documenting, refactoring, performance tuning, security and more.

Videos:

- Code Generation (10m 18s)
- Code Generation: Other AI Code Capabilities (2m 42s)
- Code Generation: GenAI Prompting Exercises (2m 37s)

Try It: In a text prompt, describe the requirements for a function you need and submit a request to the **Responses API** to generate that function and provide test cases to show it works correctly. If not, call the Responses API again with the generated code and a prompt to refine the code.

Named Entity Recognition (NER) and Structured Outputs

In this example, I use OpenAI's **natural language understanding** capabilities and **named entity recognition** to extract **structured information** from unstructured text, identifying entities such as people, places, organizations, dates, times, events, products, and more. The example shows that OpenAI's APIs can **return outputs as formatted, human-and-computer readable JSON (JavaScript Object Notation)**. NER is essential for building applications that process and organize information from documents and text sources.

Videos:

- Named Entity Recognition (NER) and Structured Outputs (10m 26s)
- NER and Structured Outputs: Code and Prompt Exercises (3m 10s)

OpenAI Documentation: Structured Model Outputs Guide

Try It: Modify the NER example to **perform parts-of-speech (POS) tagging**—identifying each word's part of speech (e.g., noun, verb, adjective, etc.) in a sentence. Use genAIs to research the commonly used tag sets for POS tagging, then prompt the model to return a structured JSON response with the parts of speech for the words in the supplied text and display each word with its part of speech. Each JSON object should contain key-value pairs for the keys `"word"` and `"tag"`.

Try It: Modify the NER example to **translate text into multiple languages**. Prompt the model to translate the text it receives to the specified languages and to return only JSON-structured data in the following format, then display the results:

```
{
  "original_text": original_text_string,
  "original_language": original_text_language_code,
  "translations": [
    {
      "language": translated_text_language_code,
      "translation": translated_text_string
    }
  ]
}
```

Try It: Create a tool that extracts key entities from news articles and outputs them in a structured JSON format.

Speech Recognition and Speech Synthesis

In this video, I introduce **speech-to-text transcription** and **text-to-speech conversion (speech synthesis)** concepts that are the foundation for working with audio input and output in your AI applications. You'll understand the models used in the transcription and synthesis examples, and explore the speech voices via OpenAI's voice demo site—https://openai.fm.

Video: Speech Recognition and Speech Synthesis: Overview (5m 27s)

OpenAI Documentation:

- Speech to Text Guide
- Text to Speech Guide

Try It: Try all the voices at https://openai.fm. Which do you prefer? Why?

English Speech-to-Text (STT) for Audio Transcription

Here, I **convert spoken audio to text**. Speech-to-text technology enables applications like **automated transcription services**, **voice commands**, and **accessibility features**.

Videos:

- English Speech-to-Text for Audio Transcription (5m 32s)
- English Speech-to-Text for Audio Transcription: Generative AI Prompt Exercises (2m 14s)

OpenAI Documentation: Speech to Text Guide

Try It: Build a transcription tool that converts `.mp3` and `.m4a` audio files to text.

Text-To-Speech (TTS)

Here, I **convert written text into natural-sounding speech** with one of OpenAI's 11 voice options. I discuss selecting voice options, specifying speech style and tone, and generating audio files. Text-to-speech technology is crucial for **creating voice assistants**, **audiobook generation**, and **accessibility applications**.

Videos:

- Text-To-Speech (11m 15s)
- Text-To-Speech: Generative AI Prompting and Coding Exercises (2m 53s)

OpenAI Documentation: Text to Speech Guide

Try It: Create an app that converts documents to audio files with selectable voices.

Image Generation

Here, I **create original images from text descriptions** using OpenAI's latest image-generation model. Image generation opens possibilities for creative content, design mockups, and visual storytelling.

Videos:

- Image Generation: Overview (6m 32s)
- Image Generation (7m 48s)
- Image Generation — Generative AI Prompting Exercises (1m 30s)

OpenAI Documentation: Images and Vision Guide

Try It: Build an image-generation tool that creates variations based on text prompts.

Image Style Transfer

In two examples, I **apply artistic styles to existing images** using the **Images API's edit capability** with **style-transfer prompts** and the **Responses API's image generation tool** to transfer the style of one image to another.

Videos:

- Image Style Transfer: Overview (3m 10s)
- Style Transfer via the Images API's Edit Capability and a Style-Transfer Prompt (10m 12s)
- Style Transfer Via the Responses API's Image Generation Tool (12m)
- Image Style Transfer: Generative AI Prompting Exercises (1m 2s)

OpenAI Documentation: Images and Vision Guide

Try It: Create a style transfer application that transforms user photos into different artistic styles, such as Vincent van Gogh, Leonardo da Vinci and others.

Generating Closed Captions from a Video's Audio Track

In this example, I generate **closed captions** from a video file's audio track using OpenAI's audio transcription capabilities. Closed captions enhance video accessibility and improve content searchability. This example covers caption formatting standards, audio extraction techniques and using the **OpenAI Whisper model**, which supports **generating captions with timestamps**. I then use the open-source VLC Media Player to overlay the closed captions on the corresponding video.

Video: Generating Closed Captions from a Video's Audio Track (9m 7s)

OpenAI Documentation: Speech to Text Guide

Try It: Build a caption generator that programmatically extracts audio from videos and creates properly formatted subtitle files. Investigate the `moviepy` module for conveniently extracting a video's audio track in Python.

Content Moderation

Here, I use **OpenAI's Moderation APIs** to detect and **filter inappropriate or harmful text and images**—essential techniques for platforms hosting user-generated content. Paul presents moderation categories and severity levels, demonstrates the **Moderation API with text inputs** and discusses **image moderation**.

Videos:

- Content Moderation (15m 30s)
- Content Moderation: Generative AI Prompting Exercise (26s)

OpenAI Documentation: Moderation Guide

Try It: Create a content moderation system that screens user submissions and flags potentially problematic content.

Sora 2 Video Generation

This video introduces **OpenAI Sora's video-generation capabilities**. I present prompt-to-video and image-to-video demos. **Coming soon:** I am developing API-based video-generation and video-remixing code examples using OpenAI's recently released **Sora 2 APIs** and will add videos based on these code examples when I complete them.

Video: Sora Video Generation (10m 58s)

OpenAI Documentation: Video Generation with Sora Guide

Try It: Experiment with text-to-video prompts and explore the creative possibilities of AI video generation.

Closing Note

As I develop additional OpenAI API-based apps, I will add new videos to this Python Fundamentals lesson on **Building API-Based Python GenAI Applications**. Some new example possibilities include:

- Generating and remixing videos with OpenAI's Sora 2 API.
- Using OpenAI's Realtime Audio APIs for speech-to-speech apps.
- Building AI agents with OpenAI's AgentKit.
- Single-tool AI agents.
- Multi-tool AI agents.
- Single-agent applications.
- Multi-agent applications.
- Managing AI conversations that maintain state between Responses API calls.

Try It: Review the course materials and start planning your own GenAI applications using the techniques learned. Enjoy!

Additional Resources

- **OpenAI Platform Documentation:** https://platform.openai.com
- **OpenAI Community Forum:** https://community.openai.com
- **Official OpenAI Python Library:** https://github.com/openai/openai-python