S&DS 617 Assignment 0

The goal of this assignment is to set up your OpenAI key and get familiar with how to use it in Python.

To begin, please first follow instructions in README.md. Then, run the below cells and edit when necessary. Make sure you understand each line of the code.

Problem 1: Set Up OpenAI API Key

Get the API key from your environment. If the API key is not found, ensure that you are starting Jupyter Lab or VS Code from the same terminal session where you set the environment variable. Environment variables are session-specific.

In [1]:
```
import openai
import os
from dotenv import load_dotenv
```

In [3]:
```
# Load environment variables from the .env file
load_dotenv()

# Access the OpenAI API key
openai_api_key = os.getenv("OPENAI_API_KEY")

# Use the API key
if openai_api_key:
    print("OpenAI API Key loaded successfully!")
else:
    print("OpenAI API Key not found. Please check your .env file.")
```

```
OpenAI API Key loaded successfully!
```

Problem 2: Use the Model

Now that the API key is set, we can chat with the model.

In [5]:
```
# Set up the OpenAI client
client = openai.OpenAI(api_key=os.environ.get("OPENAI_API_KEY"))

# Make a chat completion request
chat_completion = client.chat.completions.create(
    messages=[
        {
            "role": "user",
            "content": "Please write a haiku",
        }
    ],
    model="gpt-4o",  # Specify the model
)

# Access the response content using attributes
```

```
response_content = chat_completion.choices[0].message.content
print(response_content)
```

```
Golden leaves flutter,
Whispers of autumn's chill breeze,
Silent earth prepares.
```

Ask the model anything! For example, you can ask it to write a song, tell a joke, create a recipe, give travel recommendations, etc. Play around with the model and be creative!

Problem 3: Providing Context

Note we can also give the model extra instructions or information that is not shown to the user directly. To do this we can provide the model with context.

In [6]:
```
# Function to set up OpenAI client
def setup_openai_client():
    return openai.OpenAI(api_key=os.environ.get("OPENAI_API_KEY"))

# Function to get chat response with context
def get_chat_response_with_context(client, messages, model="gpt-4"):
    """
    Sends a chat completion request with the given messages and model.

    Args:
        client: The OpenAI client instance.
        messages: A list of message dictionaries containing "role" and "c
        model: The OpenAI model to use.

    Returns:
        The response content as a string.
    """
    try:
        chat_completion = client.chat.completions.create(
            messages=messages,
            model=model,
        )
        return chat_completion.choices[0].message.content
    except Exception as e:
        print(f"Error occurred: {e}")
        return None

# Function to manage context and make the chat request
def run_model_with_context(client, initial_context, user_message, model="
    """
    Handles the conversation context and sends the request.

    Args:
        client: The OpenAI client instance.
        initial_context: A string providing initial context for the conve
        user_message: A string containing the user's input.
        model: The OpenAI model to use.

    Returns:
        The response content as a string.
    """
```

```
    # Construct the message list
    messages = [
        {"role": "system", "content": initial_context},
        {"role": "user", "content": user_message},
    ]

    # Get the chat response
    return get_chat_response_with_context(client, messages, model)
```

In [8]:

```
# Set up the OpenAI client
client = setup_openai_client()

# Define the initial context and user message
initial_context = """
        You are building a 3D model of an object.\n
        At the start, you'll be given the name of the object you need to
        In each round, I will provide you with an observation. Based on t
        Your action should be formatted like this:\n
        place[part_name, position]\n\n
        Here, part_name is one of the parts from the list, and position i
        When you respond, use this format:\n\nThought: ...\nAction: ... \
"""
user_message = """Your task is to build a 3D object of a airplane. \n
            The list of the parts you need to use is:\n
            airplane_vertical_stabilizer_01, Dimensions (width, depth, he
            wheel_01, Dimensions (width, depth, height): (0.008, 0.017, 0
            wheel_02, Dimensions (width, depth, height): (0.008, 0.017, 0
            wheel_03, Dimensions (width, depth, height): (0.008, 0.017, 0
            wheel_04, Dimensions (width, depth, height): (0.008, 0.017, 0
            wheel_05, Dimensions (width, depth, height): (0.008, 0.017, 0
            wheel_06, Dimensions (width, depth, height): (0.008, 0.017, 0
            wheel_07, Dimensions (width, depth, height): (0.008, 0.017, 0
            wheel_08, Dimensions (width, depth, height): (0.008, 0.017, 0
            wheel_09, Dimensions (width, depth, height): (0.008, 0.017, 0
            wheel_10, Dimensions (width, depth, height): (0.008, 0.017, 0
            wheel_11, Dimensions (width, depth, height): (0.006, 0.014, 0
            wheel_12, Dimensions (width, depth, height): (0.006, 0.014, 0
            wheel_13, Dimensions (width, depth, height): (0.008, 0.017, 0
            wheel_14, Dimensions (width, depth, height): (0.008, 0.017, 0
            airplane_wing_01, Dimensions (width, depth, height): (0.421,
            airplane_wing_02, Dimensions (width, depth, height): (0.421,
            airplane_horizontal_stabilizer_01, Dimensions (width, depth,
            airplane_horizontal_stabilizer_02, Dimensions (width, depth,
            wheel_connector_01, Dimensions (width, depth, height): (0.05,
            wheel_connector_02, Dimensions (width, depth, height): (0.05,
            wheel_connector_03, Dimensions (width, depth, height): (0.016
            wheel_cover_01, Dimensions (width, depth, height): (0.004, 0.
            wheel_cover_02, Dimensions (width, depth, height): (0.004, 0.
            airplane_body_01, Dimensions (width, depth, height): (0.088,
            airplane_window_01, Dimensions (width, depth, height): (0.087
            airplane_engine_fan_01, Dimensions (width, depth, height): (0
            airplane_engine_cowling_01, Dimensions (width, depth, height)
            airplane_engine_pylon_01, Dimensions (width, depth, height):
            airplane_engine_pylon_02, Dimensions (width, depth, height):
            airplane_engine_fan_02, Dimensions (width, depth, height): (0
            airplane_engine_cowling_02, Dimensions (width, depth, height)
"""

# Run the model with context
response = run_model_with_context(client, initial_context, user_message)
```

```
# Print the response
if response:
    print("AI Response:", response)
```

```
AI Response: Thought: The best way to start building the airplane would be
to start with the main body as it would give a foundation to build upon.

Action: place[airplane_body_01, (0,0,0)]
```

- I found that GPT can understand this quite complicated task and give the action in the correct form.
- Also, the action GPT took is very reasonble. It placed the center piece of the airplane at the center