Importing Libraries: I start by bringing in the necessary tools for my code. "**openai**" helps me access the GPT3 language model.

From "IPython.display," I import "**Markdown**" and "**display**" functions to display **formatted text**.

Setting API Key: Next, I set my API key for Openai. This key allows me to make requests to the Openai API and use the language model.

Initializing Prompt Techniques: I create a dictionary "**prompt\_techniques**." It's like a container to store different ways of interacting with the chatbot. For this dictionary, I've add various prompting techniques, such as:

**• Role play • Chained • Linked • Tree of thought**

**• Instructional • Add Examples • Style • Temperature,  
o Open ended o Instruction o Multiple Choice o Fill in the blank o Binary**

**o Ordering o Prediction o Explanation o Opinion o Scenario o Comparative.**

These **techniques** help make the chatbot more **interesting** and **interactive**.

Providing Context: I set a variable called "context" to a string that holds information about **Postpartum, Anxiety, and Stress**.

This helps the chatbot understand the topic of our conversation.

Defining the Function "generate\_answer": I define a function called "generate\_answer" that takes three inputs: "**question**" (the user's input/question), "**prompt\_technique**" (a specific way to talk to the chatbot), and "**context**" (the topic of conversation).

"generate\_answer" function:

I create a "**prompt**" using the "prompt\_technique", the user's question, and the context. This "prompt" will be used to ask the text-davinci-003 model for an answer.

I send the "**prompt**" to the GPT model using the "openai.Completion.create()" function, and it gives me a "response" based on the input.

From the "**response**," I extract the generated answer. I only take the first choice, and I remove any extra spaces around it.

Finally, the function returns the generated answer.

Chatbot-Loop:

It start by greeting the user with the message "**Hello, I am your Executive Coach. How can I help you?** Type 'exit' to end."

IT enter an infinite loop using "**while True**," which means It'll keep talking to the user until they decide to stop.

It take the user's input as text from the command line using the "**input()**" function.

If the user enters '**exit**' (**case-insensitive**), It say goodbye and break out of the loop, ending the conversation.

It try to find a matching "Input-text " from the "prompt\_techniques" dictionary based on the user's input. If It find one, It use that technique to respond. But if there's no matching technique, It use the Openai model directly to answer the user.

Then, it show the generated response to the user using the "**display**" function, making sure it's formatted nicely using **Markdown**.