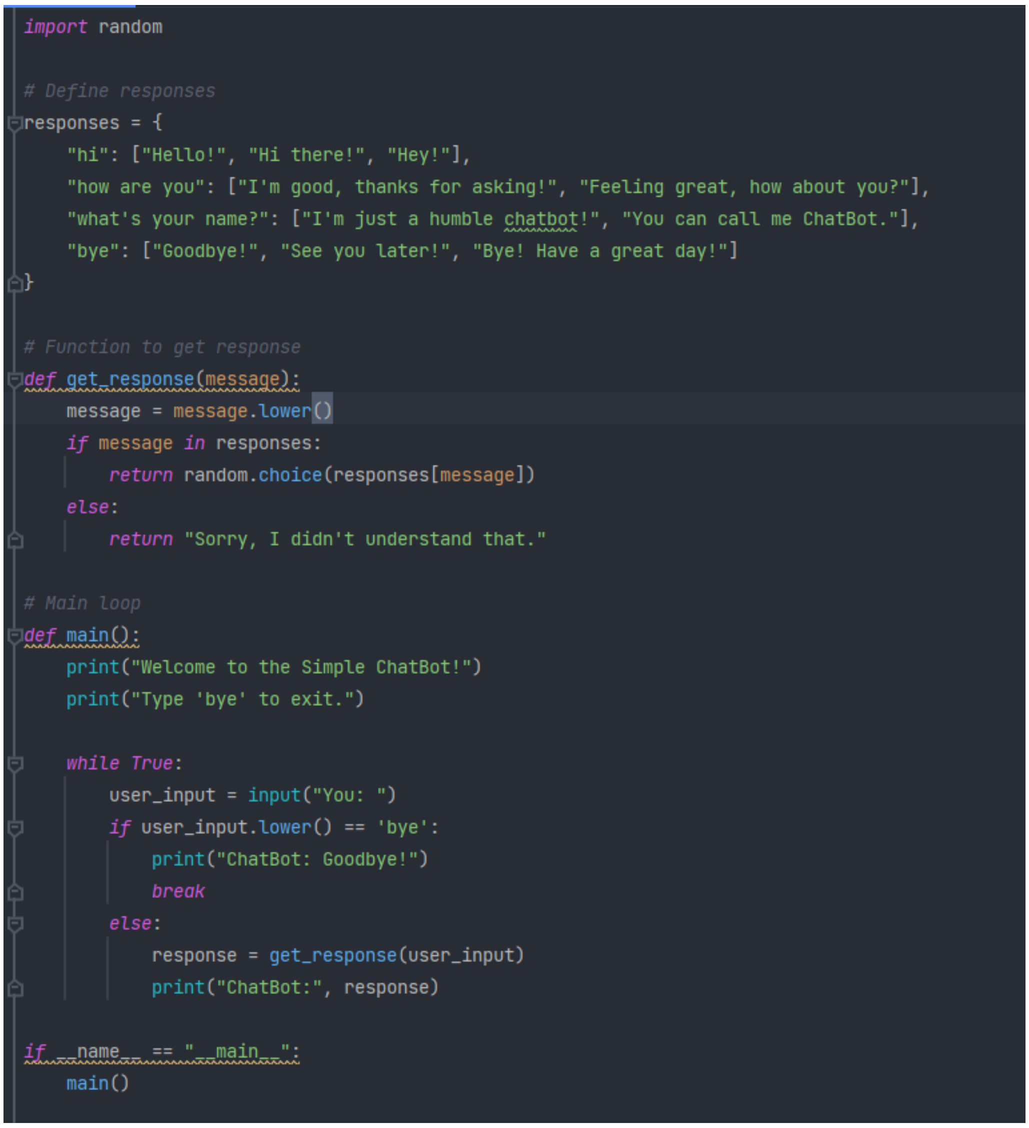
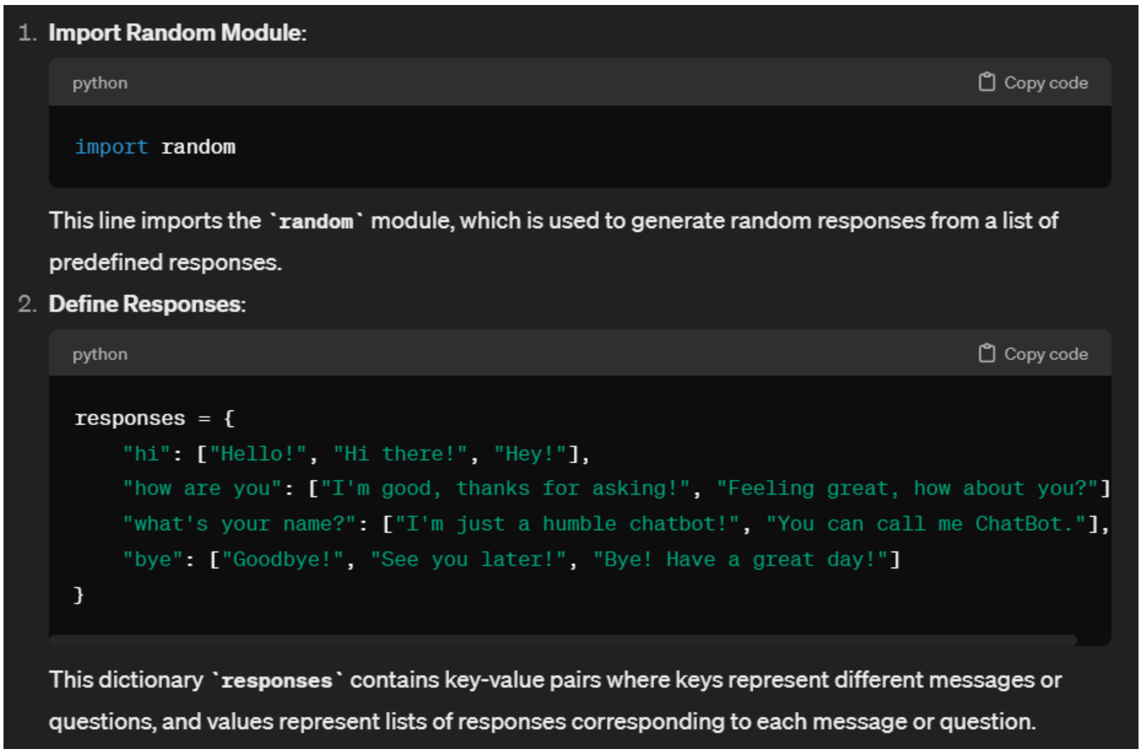
Chatbot Application (Rule-Based)



```python
import random

# Define responses
responses = {
    "hi": ["Hello!", "Hi there!", "Hey!"],
    "how are you": ["I'm good, thanks for asking!", "Feeling great, how about you?"],
    "what's your name?": ["I'm just a humble chatbot!", "You can call me ChatBot."],
    "bye": ["Goodbye!", "See you later!", "Bye! Have a great day!"]
}

# Function to get response
def get_response(message):
    message = message.lower()
    if message in responses:
        return random.choice(responses[message])
    else:
        return "Sorry, I didn't understand that."

# Main loop
def main():
    print("Welcome to the Simple ChatBot!")
    print("Type 'bye' to exit.")

    while True:
        user_input = input("You: ")
        if user_input.lower() == 'bye':
            print("ChatBot: Goodbye!")
            break
        else:
            response = get_response(user_input)
            print("ChatBot:", response)

if __name__ == "__main__":
    main()
```

Let's break down the code step by step:



1. **Import Random Module:**

python Copy code

```python
import random
```

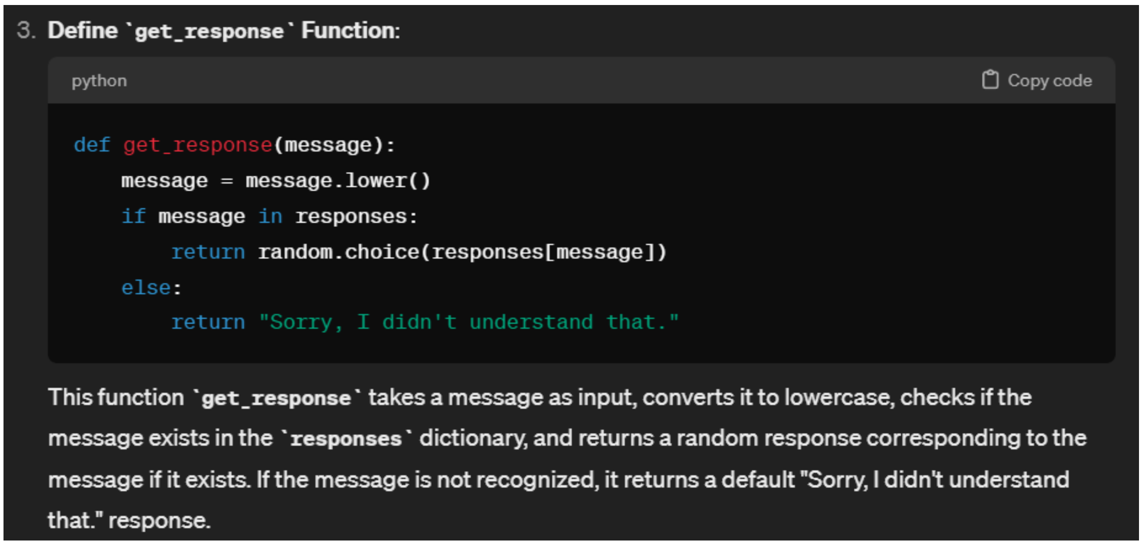
This line imports the `random` module, which is used to generate random responses from a list of predefined responses.

2. **Define Responses:**

python Copy code

```python
responses = {
    "hi": ["Hello!", "Hi there!", "Hey!"],
    "how are you": ["I'm good, thanks for asking!", "Feeling great, how about you?"]
    "what's your name?": ["I'm just a humble chatbot!", "You can call me ChatBot."],
    "bye": ["Goodbye!", "See you later!", "Bye! Have a great day!"]
}
```

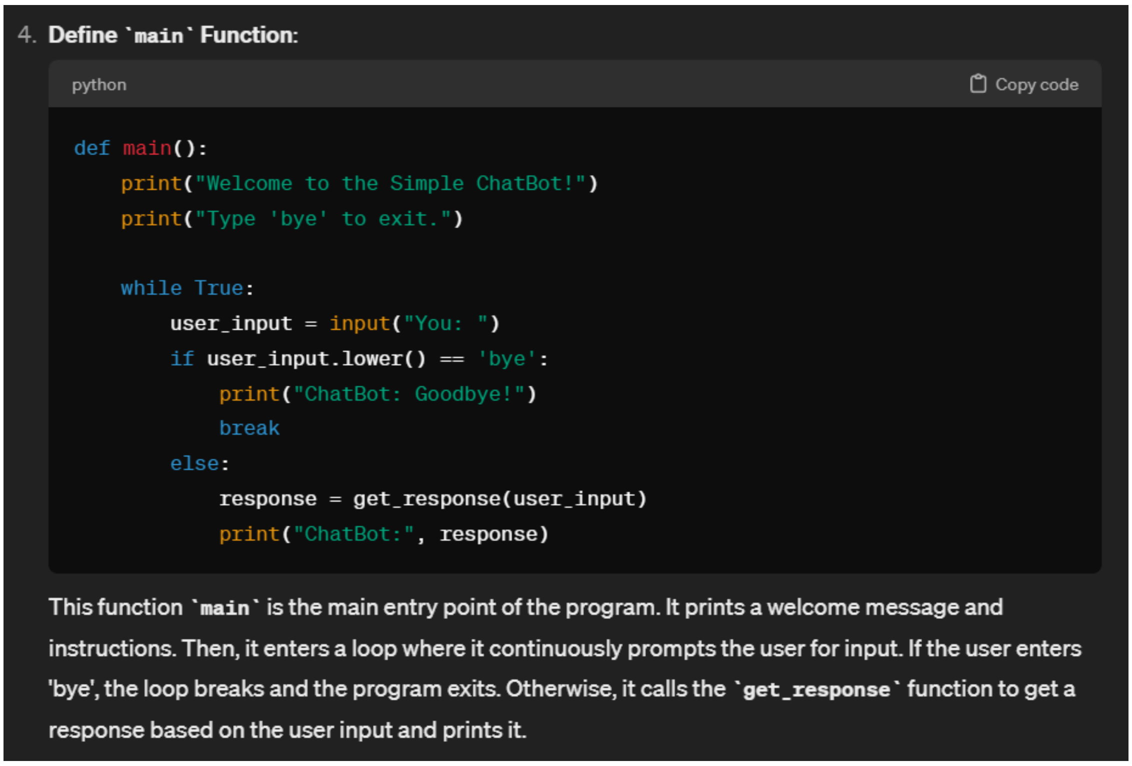
This dictionary `responses` contains key-value pairs where keys represent different messages or questions, and values represent lists of responses corresponding to each message or question.



3. **Define `get_response` Function:**

```python
def get_response(message):
    message = message.lower()
    if message in responses:
        return random.choice(responses[message])
    else:
        return "Sorry, I didn't understand that."
```

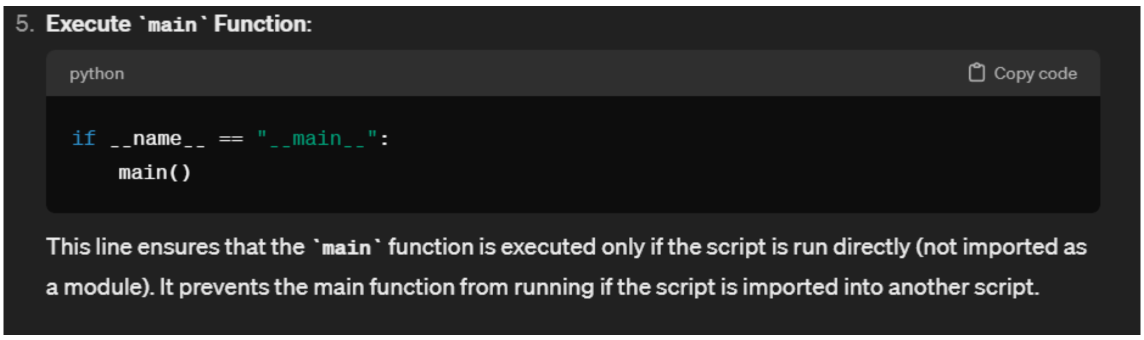
This function `get_response` takes a message as input, converts it to lowercase, checks if the message exists in the `responses` dictionary, and returns a random response corresponding to the message if it exists. If the message is not recognized, it returns a default "Sorry, I didn't understand that." response.



4. **Define `main` Function:**

```python
def main():
    print("Welcome to the Simple ChatBot!")
    print("Type 'bye' to exit.")

    while True:
        user_input = input("You: ")
        if user_input.lower() == 'bye':
            print("ChatBot: Goodbye!")
            break
        else:
            response = get_response(user_input)
            print("ChatBot:", response)
```

This function `main` is the main entry point of the program. It prints a welcome message and instructions. Then, it enters a loop where it continuously prompts the user for input. If the user enters 'bye', the loop breaks and the program exits. Otherwise, it calls the `get_response` function to get a response based on the user input and prints it.



5. **Execute `main` Function:**

```python
if __name__ == "__main__":
    main()
```

This line ensures that the `main` function is executed only if the script is run directly (not imported as a module). It prevents the main function from running if the script is imported into another script.

Assignment to develop a similar chatbot application with some modifications:

Assignment: Develop a Simple ChatBot

Objective: To create a simple chatbot application using Python, focusing on basic programming concepts like dictionaries, functions, loops, and user input/output.

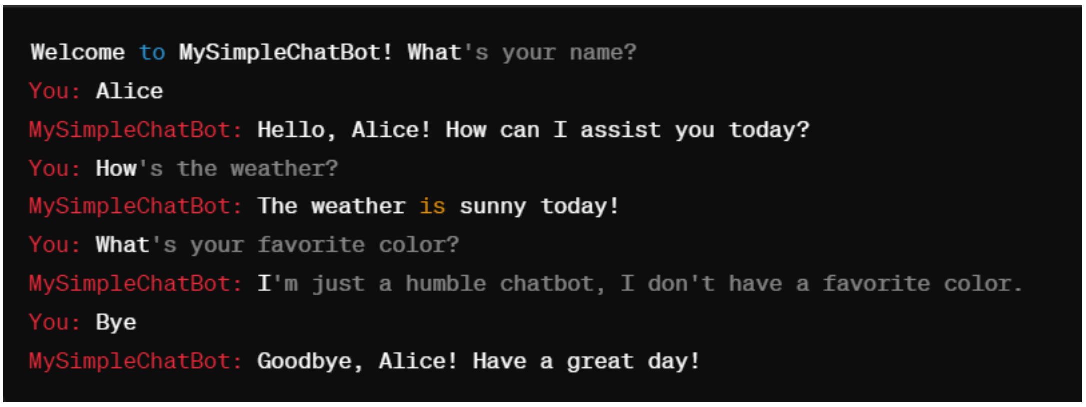
Instructions:

1. **Functionality Requirements:**
   - The chatbot should greet the user upon startup and ask for their name.
   - The chatbot should respond to at least four different messages or questions from the user with predefined responses. These responses should be stored in a dictionary.
   - Add at least one new type of message or question and its corresponding responses to the chatbot. Ensure the responses are appropriate to the message/question.
   - The chatbot should handle unrecognized messages gracefully by responding with a default message.
   - The chatbot should allow the user to exit the conversation by typing 'bye'. Upon exiting, the chatbot should bid farewell to the user.
2. **Technical Requirements:**
   - Use functions to organize your code logically. For example, separate the response retrieval logic into a function.
   - Implement error handling to handle unexpected user inputs gracefully.
   - Ensure your code is well-documented with comments explaining the purpose of each function and significant blocks of code.
3. **Additional Features (Optional):**
   - Add a feature where the chatbot remembers the user's name and uses it in responses.
   - Implement a feature to handle multiple variations of the same message/question.
   - Integrate basic natural language processing techniques to improve the chatbot's understanding of user inputs.

Submission Guidelines:

- Submit a Python script (.py file) containing your chatbot implementation.
- Ensure your code follows proper coding conventions and is well-structured.
- Include a brief explanation of any additional features or modifications you have implemented beyond the basic requirements.

Sample Output:



```
Welcome to MySimpleChatBot! What's your name?
You: Alice
MySimpleChatBot: Hello, Alice! How can I assist you today?
You: How's the weather?
MySimpleChatBot: The weather is sunny today!
You: What's your favorite color?
MySimpleChatBot: I'm just a humble chatbot, I don't have a favorite color.
You: Bye
MySimpleChatBot: Goodbye, Alice! Have a great day!
```

Note: Feel free to be creative and add more features to enhance your chatbot's functionality beyond the basic requirements. Good luck and have fun coding!