**TITLE:** CodTech IT Solutions Internship - Task Documentation: “RULE BASED CHAT BOAT” Using Conditional Statements and Pattern Matching

**INTERN INFORMATION:**

**Name:** SAI TEJA BOMMA

**ID:** ICOD7167

**INTRODUCTION**

Chatbots have become increasingly prevalent in today's digital landscape, serving as virtual assistants capable of engaging in conversations with users through text or speech interfaces. These intelligent programs are designed to understand natural language inputs and provide appropriate responses, mimicking human-like interactions. The primary purpose of chatbots is to streamline communication and automate tasks across various domains, including customer service, information retrieval, and entertainment

**Purpose of Chatbots:**

Chatbots serve several purposes, including:

**Customer Support**: Chatbots assist customers by answering frequently asked questions, troubleshooting issues, and providing guidance, thereby reducing the workload on human customer support agents and improving response times.

**Information Retrieval**: Chatbots retrieve and present relevant information to users based on their queries, such as weather forecasts, news updates, or product details, offering a convenient and efficient way to access information.

**Task Automation**: Chatbots automate routine tasks, such as appointment scheduling, reservations, or order processing, by interacting with users in a conversational manner, enhancing efficiency and reducing manual effort.

**Engagement and Entertainment**: Chatbots engage users through interactive conversations, games, quizzes, or storytelling, providing entertainment and enhancing user experience in various applications

**Rule-Based Chatbots :**

Rule-based chatbots, also known as scripted or deterministic chatbots, operate on predefined rules and responses. These chatbots do not rely on machine learning or artificial intelligence techniques for understanding language; instead, they use if-else statements or pattern matching techniques to identify user inputs and provide predefined responses.

**IMPLEMENTATION**

**Define Predefined Rules and Responses :**

* Create a dictionary where keys represent user queries or patterns, and values represent corresponding responses**.**

**Process User Input:**

* Get user input and convert it to lowercase for case-insensitive matching.
* Tokenize the user input to identify individual words or tokens.
* Remove stopwords (common words that do not carry much meaning) if necessary.

**Match User Input with Predefined Rules**:

* Iterate through the predefined rules and check if any of the rule keys match the user input.
* Use if-else statements or pattern matching techniques to identify matches.
* If a match is found, retrieve the corresponding response from the rules dictionary.

**Provide Appropriate Response:**

* If a match is found, provide the corresponding response to the user.
* If no match is found, provide a default response indicating that the chatbot didn't understand the input.

**Interaction Loop:**

* Continuously prompt the user for input and provide responses until the user decides to end the conversation (e.g., by typing "bye").

**CODE EXPLAINATION**

The code defines a simple chatbot that responds to user inputs based on predefined rules stored in a dictionary called rules. Each rule consists of a pattern or keyword mapped to a corresponding response.

1. **Define Predefined Rules and Responses:**

Create a dictionary where keys represent user queries or patterns, and values represent corresponding responses.

**For example:**

rules = {

"hi": "Hello! How can I assist you today?",

"how are you": "I'm just a chatbot, but thanks for asking!",

"bye": "Goodbye! Have a great day!",

}

1. **Processing User Input:**

The simple\_chatbot function takes user input as its parameter. It converts the input to lowercase to ensure case-insensitive matching

.**For example:**

user\_input = input("You: ").lower()

tokens = user\_input.split()

# Assuming stopwords are defined in a list called "stopwords"

filtered\_tokens = [token for token in tokens if token not in stopwords]

1. **Matching User Input with Rules:**

The function iterates through the rules dictionary and checks if any pattern or keyword from the rules matches the user input. If a match is found, it returns the corresponding response.

.**For example:**

matched\_response = None

for pattern, response in rules.items():

if pattern in filtered\_tokens:

matched\_response = response

break

# Use if-else statements for matching

if matched\_response:

print("Bot:", matched\_response)

else:

print("Bot: I'm sorry, I didn't understand that.")

1. **Providing Responses:**

If a match is found, the function returns the associated response. If no match is found, it returns a default response indicating that it didn't understand the input.

.**For example:**

# If a match is found, provide the corresponding response

if matched\_response:

print("Bot:", matched\_response)

else:

print("Bot: I'm sorry, I didn't understand that.")

1. **Interaction Loop:**

# Main loop for interacting with the user

print("Welcome to the Simple Chatbot!")

print("You can start chatting. Type 'bye' to exit.")

while True:

user\_input = input("You: ").lower()

if user\_input == "bye":

print("Bot: Goodbye! Have a great day!")

break

else:

# Process user input, match with rules, and provide response

# (code for processing user input, matching, and providing response goes here)

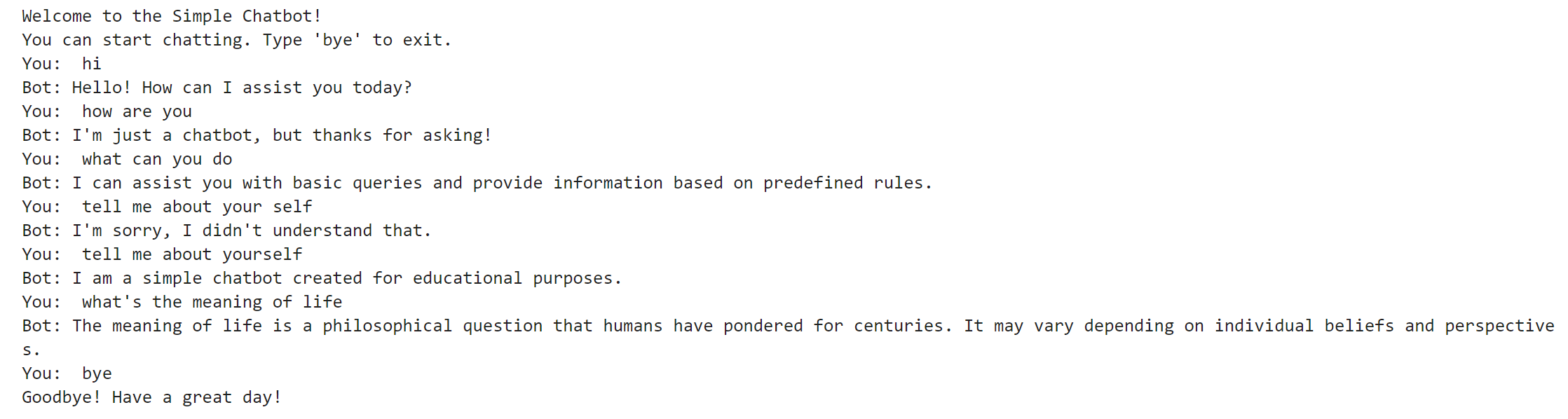
**USAGE**

* **Handling Frequently Asked Questions (FAQs):** Rule-based chatbots are well-suited for addressing common queries and providing standard responses. They can efficiently handle FAQs related to products, services, policies, or procedures, freeing up human resources from repetitive tasks.
* **Automating Routine Tasks:** Rule-based chatbots excel at automating routine tasks that follow predefined workflows or decision trees. For example, they can assist with appointment scheduling, order processing, reservation bookings, and form submissions, streamlining processes and improving efficiency.
* **Providing Consistent Responses**: Rule-based chatbots ensure consistency in responses by adhering to predefined rules and guidelines. They offer reliable and standardized interactions with users, which is particularly important in customer service and support scenarios.
* **Navigating Informational Queries:** Rule-based chatbots can navigate through structured information and provide relevant responses based on user queries. They are effective at retrieving data from knowledge bases, databases, or FAQs to address informational requests accurately.

**CONCLUSION**

This simple chatbot demonstrates the use of predefined rules and responses to provide basic conversational interactions. By matching user inputs with predefined patterns or keywords, the chatbot identifies user queries and provides appropriate responses. While this approach is limited in its capabilities compared to more advanced natural language processing techniques, it offers a foundational understanding of conversation flow and demonstrates how simple if-else statements or pattern matching techniques can be used to create chatbots.

**OUTPUT**

****

