Aim: To build a Cognitive text based application to understand context for Insurance Help.

Theory:

Objective:
The objective of this experiment is to develop a basic cognitive text-based application that can assist users with insurance-related queries. The application will be able to understand user input, identify keywords, and provide relevant information or responses. The primary goal is to demonstrate the basic principles of building a text-based chatbot for insurance help.

Theoretical Background:
1. **Natural Language Processing (NLP):** NLP is a field of artificial intelligence that focuses on the interaction between computers and human language. It includes techniques for text analysis, sentiment analysis, and language understanding, which are essential for building chatbots.

2. **Keyword-Based Text Analysis:** Keyword-based text analysis involves searching for specific words or phrases in a text to extract relevant information. In this experiment, we will use keyword matching to identify user queries related to insurance.

Libraries Used:
- **spaCy:** We will use the spaCy library for advanced text processing, including tokenization, lemmatization, and stop word removal.

- **nltk** - NLTK (Natural Language Toolkit) is a popular Python library for natural language processing, offering a wide range of tools and resources for tasks such as text analysis, tokenization, and part-of-speech tagging.

Steps:
1. Data Preparation: We will define a set of sample customer queries and their corresponding insurance responses. These queries will be used to train and test our chatbot.

2. Text Preprocessing: We will preprocess the text by tokenizing, lemmatizing, and removing stop words and punctuation using spaCy.

3. Keyword Matching: We will implement keyword matching to identify relevant queries based on the presence of specific keywords.

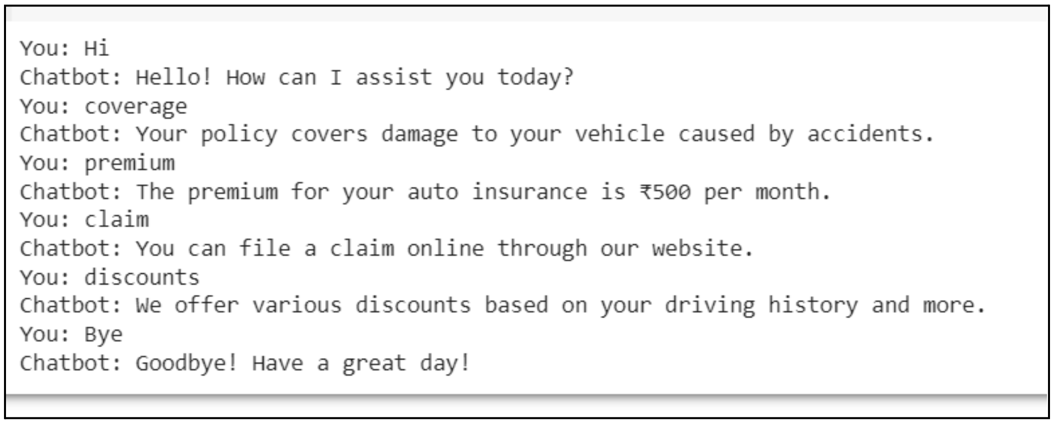
4. User Interaction: Users will interact with the chatbot by entering queries, and the chatbot will respond based on the identified keywords and similarity scores.

Expected Outcome:
The experiment is expected to result in a basic insurance assistance chatbot that can respond to user queries by matching them with predefined keywords and responses. It will demonstrate the fundamental principles of building a cognitive text-based application for insurance help.

Code:
```
import nltk
import spacy

# Load spaCy model for advanced text processing
nlp = spacy.load("en_core_web_sm")

# Sample customer queries and responses with keywords
queries_and_responses = [
     ("policy coverage", "Your policy covers damage to your vehicle caused
by accidents."),
    ("file a claim", "You can file a claim online through our website."),
    ("premium for auto insurance", "The premium for your auto insurance is
₹500 per month."),
     ("rental car expenses", "Yes, rental car expenses are covered under
your policy."),
        ("update contact information", "You can update your contact
information in your online account."),
     ("grace period for premium payment", "The grace period for premium
payment is 30 days."),
    ("add a new driver to my policy", "Yes, you can add a new driver to your policy. Contact our support for details."),
    ("cancel my policy", "To cancel your policy, please contact our customer service department."),
    ("discounts", "We offer various discounts based on your driving history and more."),
]

# Default responses for various scenarios
default_responses = {
    "greeting": "Hello! How can I assist you today?",
    "farewell": "Goodbye! Have a great day!",
    "default": "I'm sorry, I didn't understand your question. Please ask something else.",
}

# Function to classify user queries
def classify_query(user_query):
    user_query = user_query.lower()  # Convert user query to lowercase for case-insensitive matching

    # Check for greetings and farewells
    if any(greeting in user_query for greeting in ["hi", "hello"]):
        return "greeting"
    elif any(farewell in user_query for farewell in ["bye", "goodbye"]):
        return "farewell"

    # Check for keyword matches in queries
    for keywords, response in queries_and_responses:
        for keyword in keywords.split():
            if keyword in user_query:
                return response

    return "default"  # No match found

# Interactive user interface
while True:
    user_query = input("You: ")

    # Classify the user's query
    query_type = classify_query(user_query)

    # Handle different query types
    if query_type == "greeting":
        print("Chatbot:", default_responses["greeting"])
    elif query_type == "farewell":
        print("Chatbot:", default_responses["farewell"])
        break  # End the chat on a farewell
    elif query_type != "default":
        print("Chatbot:", query_type)
    else:
        response = default_responses["default"]
        print("Chatbot:", response)
```

Code Explanation -
- The code defines a chatbot that responds to user input based on the presence of keywords from a predefined set of customer queries.
- It uses the spaCy library for natural language processing to preprocess and analyze user queries and responses.
- The chatbot classifies user queries into different categories: greetings, farewells, keyword-based responses, or a default response.
- Responses are provided according to the detected category, with specific answers for greetings, farewells, and keyword matches, while handling unknown queries with a default response.
- The chatbot continues to interact with the user until a farewell keyword is detected, at which point it ends the conversation.

Output:



```
You: Hi
Chatbot: Hello! How can I assist you today?
You: coverage
Chatbot: Your policy covers damage to your vehicle caused by accidents.
You: premium
Chatbot: The premium for your auto insurance is ₹500 per month.
You: claim
Chatbot: You can file a claim online through our website.
You: discounts
Chatbot: We offer various discounts based on your driving history and more.
You: Bye
Chatbot: Goodbye! Have a great day!
```

Conclusion:

In this experiment, we successfully developed a basic cognitive text-based application for insurance help, showcasing the principles of keyword-based text analysis and response generation.