Week – 1

Aim: The aim of this project is to design a simple chatbot that can answer Frequently Asked Questions (FAQs) about an organization using Python.

Description: The chatbot will be designed to answer basic questions about the organization, such as its mission, vision, products, services, and contact information. The chatbot will use a dictionary to store the FAQs and their corresponding answers. The user will be able to interact with the chatbot by typing their questions, and the chatbot will respond with the relevant answer.

Source Code:

```
# Import the required libraries

import random

# Define a dictionary to store the FAQs and their answers

faqs = {

  "What is the mission of your organization?": "Our mission is to provide innovative solutions to our customers.",

  "What is the vision of your organization?": "Our vision is to be the leading provider of technology solutions.",

  "What products do you offer?": "We offer a range of products, including software, hardware, and consulting services.",

  "How can I contact your organization?": "You can contact us through our website, phone, or email.",

  "What are your business hours?": "Our business hours are Monday to Friday, 9am to 5pm."

}


# Define a function to handle user input

def handle_input(user_input):

  # Convert the user input to lowercase

  user_input = user_input.lower()


  # Check if the user input matches any of the FAQs

  for question, answer in faqs.items():

    if user_input in question.lower():

      # Return the answer if a match is found

      return answer
    # Return a default message if no match is found
    return "Sorry, I couldn't find an answer to your question."

# Define a function to run the chatbot
def run_chatbot():
    print("Welcome to our organization's chatbot!")
    print("Type 'quit' to exit the chatbot.")

    while True:
        # Get user input
        user_input = input("User: ")

        # Check if the user wants to quit
        if user_input.lower() == "quit":
            break

        # Handle user input and print the response
        print("Chatbot: ", handle_input(user_input))

# Run the chatbot
run_chatbot()
```

Output:

```
Welcome to our organization's chatbot!
Type 'quit' to exit the chatbot.
User: What is the mission of your organization?
Chatbot:  Our mission is to provide innovative solutions to our customers.
User: What products do you offer?
Chatbot:  We offer a range of products, including software, hardware, and consulting services.
User: How can I contact your organization?
Chatbot:  You can contact us through our website, phone, or email.
User: quit
```

Week – 2

Aim: To create a Python-based chatbot that provides a seamless customer experience through Facebook Messenger, leveraging its APIs and NLP techniques.

Description: The chatbot will interact with users on Facebook Messenger, understanding their queries and providing relevant responses. It will utilize the Facebook Graph API to establish a connection, send and receive messages, and manage the conversation flow. Natural language processing techniques will be employed to process user input and generate appropriate responses.

Source Code:

```
import fbchat

from nltk.stem import WordNetLemmatizer

from nltk.corpus import stopwords

import pandas as pd


# Download required NLTK data

nltk.download('punkt')

nltk.download('wordnet')

nltk.download('stopwords')


# Load your Facebook Messenger credentials

# Replace with your actual credentials

email = 'your_email@example.com'

password = 'your_password'


# Create a client

client = fbchat.Client(email, password)


# Load FAQ data (replace with your own data)

data = pd.read_csv('faqs.csv')


# Preprocess and match functions (similar to the previous example)


# Function to handle incoming messages

def on_message(sender_id, message):
  text = message.text
  if text:
    answer = find_best_match(text)
    if answer:
      client.send_text_message(sender_id, answer)
    else:
      client.send_text_message(sender_id, "I couldn't find a matching answer.")


# Start listening for messages
client.listen_for_messages(on_message)
```

Output:

```
User: "What are your store hours?"
Chatbot: "Our store is open from 10 AM to 8 PM, Monday to Friday."
User: "Can I return a product I bought online?"
Chatbot: "Yes, you can return products within 30 days of purchase. Please visit our returns page for
more details."
User: "I forgot my password."
Chatbot: "No problem. Please click on the 'Forgot Password' link on our website to reset your
password."
User: "Can I order a pizza?"
Chatbot: "I'm sorry, I can't help you with that. Please contact our customer service for pizza orders."
```

Week – 3

Aim: The aim of this chatbot is to assist students in opening a bank account by guiding them through the process and providing necessary information.

Description: The chatbot will interact with the student, asking for required information such as name, address, and identification details. It will then provide a step-by-step guide on how to open a bank account, including the necessary documents and procedures.

Source Code:

```
import random

# Bank account types

account_types = ["Savings", "Current", "Student"]

# Required documents

required_documents = ["ID Proof", "Address Proof", "Passport Size Photo"]

def greet_student():

  print("Hello! I'm here to help you open a bank account.")

def ask_account_type():

  print("What type of account would you like to open?")

  for i, account_type in enumerate(account_types):

    print(f"{i+1}. {account_type}")

  choice = int(input("Enter your choice: "))

  return account_types[choice-1]

def ask_required_info():

  info = {}

  print("Please provide the following information:")

  info["name"] = input("Name: ")

  info["address"] = input("Address: ")

  info["id_proof"] = input("ID Proof: ")

  return info

def provide_guidance(account_type, info):

  print(f"Great! You've chosen to open a {account_type} account.")

  print("Here's a step-by-step guide to help you open your account:")

  print("1. Visit your nearest bank branch with the required documents:")

  for document in required_documents:

    print(f"- {document}")
  print("2. Fill out the account opening form and submit it to the bank representative.")
  print("3. Once your application is processed, you'll receive your account details.")
  print(f"Your account details will be: ")
  print(f"Account Holder Name: {info['name']}")
  print(f"Account Address: {info['address']}")
def main():
  greet_student()
  account_type = ask_account_type()
  info = ask_required_info()
  provide_guidance(account_type, info)
if __name__ == "__main__":
  main()
```

<u>Output:</u>

```
Hello! I'm here to help you open a bank account.
What type of account would you like to open?
1. Savings
2. Current
3. Student
Enter your choice: 2
Please provide the following information:
Name: John Doe
Address: 123 Main St
ID Proof: Passport
Great! You've chosen to open a Current account.
Here's a step-by-step guide to help you open your account:
1. Visit your nearest bank branch with the required documents:
- ID Proof
- Address Proof
- Passport Size Photo
2. Fill out the account opening form and submit it to the bank representative.
3. Once your application is processed, you'll receive your account details.
Your account details will be:
```

Account Holder Name: John Doe

Account Address: 123 Main St

Week – 4

Aim: The aim of this chatbot is to assist users in identifying potential diseases based on the symptoms they provide.

Description: The chatbot will interact with the user, asking for symptoms and then using a disease-symptom database to suggest possible diseases. The chatbot will provide a list of potential diseases, along with their corresponding symptoms and probabilities.

Source Code:

```
import pandas as pd


# Load disease-symptom database

disease_db = pd.read_csv("disease_symptom_db.csv")


def greet_user():

  print("Hello! I'm here to help you identify potential diseases based on your symptoms.")


def ask_symptoms():

  print("Please enter your symptoms (separated by commas): ")

  symptoms = input().split(",")

  symptoms = [symptom.strip() for symptom in symptoms]

  return symptoms


def find_diseases(symptoms):

  potential_diseases = []

  for index, row in disease_db.iterrows():

    disease_symptoms = row["Symptoms"].split(",")

    disease_symptoms = [symptom.strip() for symptom in disease_symptoms]

    matching_symptoms = [symptom for symptom in symptoms if symptom in disease_symptoms]

    if len(matching_symptoms) > 0:

      probability = len(matching_symptoms) / len(disease_symptoms)

      potential_diseases.append((row["Disease"], matching_symptoms, probability))

  return potential_diseases


def display_results(potential_diseases):
    print("Potential diseases based on your symptoms:")
    for disease, symptoms, probability in potential_diseases:
        print(f"Disease: {disease}")
        print(f"Matching Symptoms: {', '.join(symptoms)}")
        print(f"Probability: {probability:.2f}")
        print()


def main():
    greet_user()
    symptoms = ask_symptoms()
    potential_diseases = find_diseases(symptoms)
    display_results(potential_diseases)


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

Output:

```
Hello! I'm here to help you identify potential diseases based on your symptoms.
Please enter your symptoms (separated by commas):
fever, headache, fatigue
Potential diseases based on your symptoms:
Disease: Influenza
Matching Symptoms: fever, headache
Probability: 0.67
Disease: Malaria
Matching Symptoms: fever, fatigue
Probability: 0.50
Disease: Common Cold
Matching Symptoms: headache, fatigue
Probability: 0.33
```

Week - 5

Aim: The aim of this chatbot is to assist customers in finding and purchasing products from an online store.

Description: The chatbot will interact with the customer, asking for their preferences and requirements, and then provide a list of recommended products. The customer can then select a product to view its details, and finally, place an order.

Source Code:

```
import random


# Product database

products = [

  {"id": 1, "name": "Apple iPhone", "price": 999, "description": "Latest iPhone model"},

  {"id": 2, "name": "Samsung TV", "price": 1299, "description": "4K UHD TV with HDR"},

  {"id": 3, "name": "Nike Shoes", "price": 79, "description": "Comfortable and stylish shoes"},

  # ...

]


def greet_customer():

  print("Hello! Welcome to our online store. How can I assist you today?")


def ask_preferences():

  print("What type of product are you looking for?")

  print("1. Electronics")

  print("2. Fashion")

  print("3. Home Goods")

  choice = int(input("Enter your choice: "))

  if choice == 1:

    return "Electronics"

  elif choice == 2:

    return "Fashion"

  else:

    return "Home Goods"
```

```
def recommend_products(preferences):
    recommended_products = []
    for product in products:
        if product["category"] == preferences:
            recommended_products.append(product)
    return recommended_products

def display_products(products):
    print("Here are some recommended products:")
    for product in products:
        print(f"ID: {product['id']}, Name: {product['name']}, Price: ${product['price']}")

def ask_product_details():
    product_id = int(input("Enter the ID of the product you're interested in: "))
    for product in products:
        if product["id"] == product_id:
            print(f"Product Details: {product['description']}")
            return product
    print("Product not found!")

def place_order(product):
    print(f"Thank you for purchasing {product['name']}!")
    print("Your order has been placed successfully.")

def main():
    greet_customer()
    preferences = ask_preferences()
    recommended_products = recommend_products(preferences)
    display_products(recommended_products)
    product = ask_product_details()
    place_order(product)
```

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

Output:

```
Hello! Welcome to our online store. How can I assist you today?
What type of product are you looking for?
1. Electronics
2. Fashion
3. Home Goods
Enter your choice: 1
Here are some recommended products:
ID: 1, Name: Apple iPhone, Price: $999
ID: 2, Name: Samsung TV, Price: $1299
Enter the ID of the product you're interested in: 1
Product Details: Latest iPhone model
Thank you for purchasing Apple iPhone!
Your order has been placed successfully.
```

Week – 6

Aim: The aim of this chatbot is to assist tourists in planning their trips by providing information about popular destinations, attractions, and activities.

Description: The chatbot will interact with the tourist, asking for their preferences and requirements, and then provide a personalized travel plan. The chatbot will offer suggestions for destinations, attractions, and activities based on the tourist's interests and budget.

Source Code:

```
import random


# Destination database

destinations = [

    {"id": 1, "name": "Paris", "description": "City of love and romance", "attractions": ["Eiffel Tower", "Louvre Museum"]},

    {"id": 2, "name": "New York City", "description": "City that never sleeps", "attractions": ["Statue of Liberty", "Central Park"]},

    {"id": 3, "name": "Tokyo", "description": "Vibrant city with rich culture", "attractions": ["Tokyo Tower", "Shibuya Crossing"]},

    # ...

]


# Activity database

activities = [

    {"id": 1, "name": "Sightseeing", "description": "Explore popular landmarks"},

    {"id": 2, "name": "Food Tour", "description": "Taste local cuisine"},

    {"id": 3, "name": "Adventure", "description": "Go hiking or biking"},

    # ...

]


def greet_tourist():

    print("Hello! Welcome to our tourism chatbot. How can I assist you in planning your trip?")


def ask_destination_preferences():

    print("What type of destination are you interested in?")

    print("1. City Break")
    print("2. Beach Vacation")
    print("3. Nature Escape")
    choice = int(input("Enter your choice: "))
    if choice == 1:
        return "City Break"
    elif choice == 2:
        return "Beach Vacation"
    else:
        return "Nature Escape"

def recommend_destinations(preferences):
    recommended_destinations = []
    for destination in destinations:
        if destination["category"] == preferences:
            recommended_destinations.append(destination)
    return recommended_destinations

def display_destinations(destinations):
    print("Here are some recommended destinations:")
    for destination in destinations:
        print(f"ID: {destination['id']}, Name: {destination['name']}, Description:
{destination['description']}")

def ask_activity_preferences():
    print("What type of activity are you interested in?")
    print("1. Sightseeing")
    print("2. Food Tour")
    print("3. Adventure")
    choice = int(input("Enter your choice: "))
    if choice == 1:
        return "Sightseeing"
    elif choice == 2:
        return "Food Tour"
```

```
  else:
    return "Adventure"


def recommend_activities(preferences):
  recommended_activities = []
  for activity in activities:
    if activity["category"] == preferences:
      recommended_activities.append(activity)
  return recommended_activities


def display_activities(activities):
  print("Here are some recommended activities:")
  for activity in activities:
    print(f"ID: {activity['id']}, Name: {activity['name']}, Description: {activity['description']}")


def create_travel_plan(destination, activities):
  print(f"Your travel plan for {destination['name']}:")
  print(f"Attractions: {', '.join(destination['attractions'])}")
  print(f"Activities: {', '.join([activity['name'] for activity in activities])}")


def main():
  greet_tourist()
  destination_preferences = ask_destination_preferences()
  recommended_destinations = recommend_destinations(destination_preferences)
  display_destinations(recommended_destinations)
  destination_id = int(input("Enter the ID of the destination you're interested in: "))
  for destination in destinations:
    if destination["id"] == destination_id:
      activity_preferences = ask_activity_preferences()
      recommended_activities = recommend_activities(activity_preferences)
      display_activities(recommended_activities)
      activity_ids = input("Enter the IDs of the activities you're interested in (separated by commas): ")
```

```
        activity_ids = [int(id) for id in activity_ids.split(",")]
        activities = [activity for activity in activities if activity["id"] in activity_ids]
        create_travel_plan(destination, activities)
        break


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

Output:

```
Hello! Welcome to our tourism chatbot. How can I assist you in planning your trip?
What type of destination are you interested in?
1. City Break
2. Beach Vacation
3. Nature Escape
Enter your choice: 1
Here are some recommended destinations:
ID: 1, Name: Paris, Description: City of love and romance
ID: 2, Name: New York City, Description: City that never sleeps
Enter the ID of the destination you're interested in: 1
What type of activity are you interested in?
1. Sightseeing
2. Food Tour
3. Adventure
Enter your choice: 1
Here are some recommended activities:
ID: 1, Name: Sightseeing, Description: Explore popular landmarks
Enter the IDs of the activities you're interested in (separated by commas): 1
Your travel plan for Paris:
Attractions: Eiffel Tower, Louvre Museum
Activities: Sightseeing
```

Week – 7

Aim: The aim of this chatbot is to assist patients with insomnia by providing personalized guidance and relaxation techniques to help them fall asleep.

Description: The chatbot will interact with the patient, asking about their sleep patterns, stress levels, and relaxation preferences. Based on the patient's input, the chatbot will provide a customized sleep plan, including relaxation techniques, sleep hygiene tips, and stress management strategies.

Source Code:

```
import random

# Relaxation techniques database

relaxation_techniques = [

    {"id": 1, "name": "Deep Breathing", "description": "Breathe in deeply through your nose, hold for 5 seconds, and exhale slowly"},

    {"id": 2, "name": "Progressive Muscle Relaxation", "description": "Tense and then relax different muscle groups in your body"},

    {"id": 3, "name": "Mindfulness Meditation", "description": "Focus on the present moment, without judgment"},

    {"id": 4, "name": "Visualization", "description": "Imagine yourself in a peaceful, relaxing environment"},

    {"id": 5, "name": "Yoga", "description": "Practice gentle stretches and movements to relax your body and mind"}

]


# Sleep hygiene tips database

sleep_hygiene_tips = [

    {"id": 1, "name": "Establish a Bedtime Routine", "description": "Develop a calming pre-sleep routine to signal your body that it's time to sleep"},

    {"id": 2, "name": "Create a Sleep-Conducive Environment", "description": "Make your bedroom a sleep haven by ensuring it is dark, quiet, and cool"},

    {"id": 3, "name": "Avoid Stimulating Activities Before Bed", "description": "Avoid stimulating activities like exercise, watching TV, or using electronic devices before bedtime"},

    {"id": 4, "name": "Limit Caffeine and Alcohol", "description": "Avoid consuming caffeine and alcohol in the hours leading up to bedtime"},

    {"id": 5, "name": "Get Regular Exercise", "description": "Regular exercise can help improve sleep quality, but avoid vigorous exercise within a few hours of bedtime"}

]
```

```
def greet_patient():
  print("Hello! Welcome to our insomnia chatbot. How can I assist you in improving your sleep?")


def ask_sleep_patterns():
  print("How many hours of sleep do you typically get per night?")
  hours = int(input("Enter the number of hours: "))
  print("Do you have trouble falling asleep, staying asleep, or both?")
  trouble = input("Enter 'falling asleep', 'staying asleep', or 'both': ")
  return hours, trouble


def recommend_relaxation_techniques(trouble):
  recommended_techniques = []
  if trouble == "falling asleep":
    recommended_techniques.append(relaxation_techniques[0])  # Deep Breathing
  elif trouble == "staying asleep":
    recommended_techniques.append(relaxation_techniques[1])  # Progressive Muscle Relaxation
  else:
    recommended_techniques.append(relaxation_techniques[2])  # Mindfulness Meditation
  return recommended_techniques


def display_relaxation_techniques(techniques):
  print("Here are some recommended relaxation techniques:")
  for technique in techniques:
    print(f"ID: {technique['id']}, Name: {technique['name']}, Description: {technique['description']}")


def ask_stress_levels():
  print("How would you rate your current stress level?")
  print("1. Low")
  print("2. Medium")
  print("3. High")
  choice = int(input("Enter your choice: "))
  if choice == 1:
        return "Low"
    elif choice == 2:
        return "Medium"
    else:
        return "High"

def recommend_sleep_hygiene_tips(stress_level):
    recommended_tips = []
    if stress_level == "Low":
        recommended_tips.append(sleep_hygiene_tips[0])  # Establish a Bedtime Routine
    elif stress_level == "Medium":
        recommended_tips.append(sleep_hygiene_tips[1])  # Create a Sleep-Conducive Environment
    else:
        recommended_tips.append(sleep_hygiene_tips[2])  # Avoid Stimulating Activities Before Bed
    return recommended_tips

def display_sleep_hygiene_tips(tips):
    print("Here are some recommended sleep hygiene tips:")
    for tip in tips:
        print(f"ID: {tip['id']}, Name: {tip['name']}, Description: {tip['description']}")

def create_sleep_plan(techniques, tips):
    print("Here is your personalized sleep plan:")
    print("Relaxation Techniques:")
    for technique in techniques:
        print(f"ID: {technique['id']}, Name: {technique['name']}, Description: {technique['description']}")
    print("Sleep Hygiene Tips:")
    for tip in tips:
        print(f"ID: {tip['id']}, Name: {tip['name']}, Description: {tip['description']}")

def main():
    greet_patient()
    hours, trouble = ask_sleep_patterns()
    recommended_techniques = recommend_relaxation_techniques(trouble)
    display_relaxation_techniques(recommended_techniques)
    stress_level = ask_stress_levels()
    recommended_tips = recommend_sleep_hygiene_tips(stress_level)
    display_sleep_hygiene_tips(recommended_tips)
    create_sleep_plan(recommended_techniques, recommended_tips)


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

Output:

```
Hello! Welcome to our insomnia chatbot. How can I assist you in improving your sleep?
How many hours of sleep do you typically get per night?
Enter the number of hours: 6
Do you have trouble falling asleep, staying asleep, or both?
Enter 'falling asleep', 'staying asleep', or 'both': falling asleep
Here are some recommended relaxation techniques:
ID: 1, Name: Deep Breathing, Description: Breathe in deeply through your nose, hold for 5 seconds, and exhale slowly
ID: 3, Name: Mindfulness Meditation, Description: Focus on the present moment, without judgment
How would you rate your current stress level?
1. Low
2. Medium
3. High
Enter your choice: 2
Here are some recommended sleep hygiene tips:
ID: 1, Name: Establish a Bedtime Routine, Description: Develop a calming pre-sleep routine to signal your body that it's time to sleep
ID: 2, Name: Create a Sleep-Conducive Environment, Description: Make your bedroom a sleep haven by ensuring it is dark, quiet, and cool
Here is your personalized sleep plan:
Relaxation Techniques:
ID: 1, Name: Deep Breathing, Description: Breathe in deeply through your nose, hold for 5 seconds, and exhale slowly
```

ID: 3, Name: Mindfulness Meditation, Description: Focus on the present moment, without judgment

Sleep Hygiene Tips:

ID: 1, Name: Establish a Bedtime Routine, Description: Develop a calming pre-sleep routine to signal your body that it's time to sleep

ID: 2, Name: Create a Sleep-Conducive Environment, Description: Make your bedroom a sleep haven by ensuring it is dark, quiet, and cool

Week – 8

Aim: Design a chatbot that suggests philosophy books based on user preferences.

Description: This program uses a pre-defined list of philosophy books and their corresponding authors to suggest books to users based on their interests. The chatbot uses natural language processing (NLP) techniques to understand user input and respond with relevant book suggestions.

Source Code:

```
import nltk

from nltk.tokenize import word_tokenize

from nltk.corpus import stopwords


# Pre-defined list of philosophy books and authors

philosophy_books = {

  'ethics': ['Nicomachean Ethics by Aristotle', 'Utilitarianism by John Stuart Mill'],

  'metaphysics': ['Metaphysics by Aristotle', 'Critique of Pure Reason by Immanuel Kant'],

  'epistemology': ['Meditations on First Philosophy by René Descartes', 'An Enquiry Concerning
Human Understanding by David Hume'],

  'logic': ['Prior Analytics by Aristotle', 'Principia Mathematica by Bertrand Russell and Alfred North
Whitehead']

}


stop_words = set(stopwords.words('english'))


def chatbot():

  print('Welcome to the Philosophy Book Suggester Chatbot!')

  while True:

    print('Please select one of the following options:')

    print('1. Ethics')

    print('2. Metaphysics')

    print('3. Epistemology')

    print('4. Logic')

    print('5. Exit')


    preference = input('Enter your preference (1-5): ')
```

```
        if preference == '1':
            print('You might be interested in the following philosophy books: ')
            print(', '.join(philosophy_books['ethics']))
        elif preference == '2':
            print('You might be interested in the following philosophy books: ')
            print(', '.join(philosophy_books['metaphysics']))
        elif preference == '3':
            print('You might be interested in the following philosophy books: ')
            print(', '.join(philosophy_books['epistemology']))
        elif preference == '4':
            print('You might be interested in the following philosophy books: ')
            print(', '.join(philosophy_books['logic']))
        elif preference == '5':
            print('Goodbye!')
            break
        else:
            print('Invalid preference. Please try again!')

# Run the chatbot
chatbot()
```

<u>Output</u>:

```
Welcome to the Philosophy Book Suggester Chatbot!
Please select one of the following options:
1. Ethics
2. Metaphysics
3. Epistemology
4. Logic
5. Exit
Enter your preference (1-5): 1
You might be interested in the following philosophy books:
Nicomachean Ethics by Aristotle, Utilitarianism by John Stuart Mill
```

```
Please select one of the following options:
1. Ethics
2. Metaphysics
3. Epistemology
4. Logic
5. Exit
Enter your preference (1-5): 5
Goodbye!
```

Week – 9

Aim: Design a chatbot that suggests IMDB rated movies based on user's favorite genre.

Description: This program uses a pre-defined list of IMDB rated movies and their corresponding genres to suggest movies to users based on their interests. The chatbot uses natural language processing (NLP) techniques to understand user input and respond with relevant movie suggestions.

Source Code:

```
import nltk

from nltk.tokenize import word_tokenize

from nltk.corpus import stopwords


# Pre-defined list of IMDB rated movies and their corresponding genres

movies = {

  'action': ['The Dark Knight', 'The Avengers', 'The Dark Knight Rises', 'Iron Man', 'Captain America: The Winter Soldier'],

  'comedy': ['The Hangover', 'Superbad', 'Bridesmaids', 'The 40-Year-Old Virgin', 'Anchorman'],

  'drama': ['The Shawshank Redemption', 'The Godfather', 'The Dark Knight', '12 Angry Men', 'Schindler\'s List'],

  'horror': ['The Shining', 'The Exorcist', 'The Texas Chain Saw Massacre', 'Halloween', 'The Silence of the Lambs'],

  'romance': ['Titanic', 'The Notebook', 'Casablanca', 'When Harry Met Sally', 'La La Land']

}


stop_words = set(stopwords.words('english'))


def chatbot():

  print('Welcome to the IMDB Rated Movie Suggester Chatbot!')

  while True:

    print('Please select one of the following genres:')

    print('1. Action')

    print('2. Comedy')

    print('3. Drama')

    print('4. Horror')

    print('5. Romance')
```

```
        print('6. Exit')


        preference = input('Enter your preference (1-6): ')


        if preference == '1':
            print('You might be interested in the following movies: ')
            print(', '.join(movies['action']))
        elif preference == '2':
            print('You might be interested in the following movies: ')
            print(', '.join(movies['comedy']))
        elif preference == '3':
            print('You might be interested in the following movies: ')
            print(', '.join(movies['drama']))
        elif preference == '4':
            print('You might be interested in the following movies: ')
            print(', '.join(movies['horror']))
        elif preference == '5':
            print('You might be interested in the following movies: ')
            print(', '.join(movies['romance']))
        elif preference == '6':
            print('Goodbye!')
            break
        else:
            print('Invalid preference. Please try again!')

# Run the chatbot
chatbot()
```

Output:

```
Welcome to the IMDB Rated Movie Suggester Chatbot!
Please select one of the following genres:
1. Action
2. Comedy
```

```
3. Drama

4. Horror

5. Romance

6. Exit

Enter your preference (1-6): 1

You might be interested in the following movies:

The Dark Knight, The Avengers, The Dark Knight Rises, Iron Man, Captain America: The Winter Soldier


Please select one of the following genres:

1. Action

2. Comedy

3. Drama

4. Horror

5. Romance

6. Exit

Enter your preference (1-6): 6

Goodbye!
```

Week – 10

Aim: Create a chatbot that shows how many cups won by the individual country based on the user's favorite country choice.

Description: This program uses a pre-defined list of FIFA World Cup winners and their corresponding countries to show the number of cups won by each country based on user input. The chatbot uses natural language processing (NLP) techniques to understand user input and respond with relevant information.

Source Code:

```
import nltk

from nltk.tokenize import word_tokenize

from nltk.corpus import stopwords


# Pre-defined list of FIFA World Cup winners and their corresponding countries

world_cup_winners = {

  'Brazil': 5,

  'Germany': 4,

  'Italy': 4,

  'Argentina': 2,

  'Uruguay': 2,

  'France': 2,

  'Spain': 1,

  'England': 1

}


stop_words = set(stopwords.words('english'))


def chatbot():

  print('Welcome to the FIFA World Cup Wins by Country Chatbot!')

  while True:

    print('Please select one of the following countries:')

    print('1. Brazil')

    print('2. Germany')

    print('3. Italy')
    print('4. Argentina')
    print('5. Uruguay')
    print('6. France')
    print('7. Spain')
    print('8. England')
    print('9. Exit')

    preference = input('Enter your preference (1-9): ')

    if preference == '1':
      print('Brazil has won the FIFA World Cup', world_cup_winners['Brazil'], 'times.')
    elif preference == '2':
      print('Germany has won the FIFA World Cup', world_cup_winners['Germany'], 'times.')
    elif preference == '3':
      print('Italy has won the FIFA World Cup', world_cup_winners['Italy'], 'times.')
    elif preference == '4':
      print('Argentina has won the FIFA World Cup', world_cup_winners['Argentina'], 'times.')
    elif preference == '5':
      print('Uruguay has won the FIFA World Cup', world_cup_winners['Uruguay'], 'times.')
    elif preference == '6':
      print('France has won the FIFA World Cup', world_cup_winners['France'], 'times.')
    elif preference == '7':
      print('Spain has won the FIFA World Cup', world_cup_winners['Spain'], 'times.')
    elif preference == '8':
      print('England has won the FIFA World Cup', world_cup_winners['England'], 'times.')
    elif preference == '9':
      print('Goodbye!')
      break
    else:
      print('Invalid preference. Please try again!')

# Run the chatbot
```

```
chatbot()
```

Output:

```
Welcome to the FIFA World Cup Wins by Country Chatbot!

Please select one of the following countries:

1. Brazil

2. Germany

3. Italy

4. Argentina

5. Uruguay

6. France

7. Spain

8. England

9. Exit

Enter your preference (1-9): 1

Brazil has won the FIFA World Cup 5 times.


Please select one of the following countries:

1. Brazil

2. Germany

3. Italy

4. Argentina

5. Uruguay

6. France

7. Spain

8. England

9. Exit

Enter your preference (1-9): 9

Goodbye!
```