Chat Bot in python - Simple tutorial work

Intro to computer science (University of Azad Jammu & Kashmir)

Chatbot implemented in Python

message_probability function:

- This function calculates the probability of a response based on the number of recognized words in the user's input.
- It takes the user's message, a list of recognized words, and optional parameters for single_response (a boolean indicating if it's a single-response scenario) and required_words (a list of words that must be present in the user's message).
- The function counts the number of recognized words in the user's message and calculates a percentage based on the total number of recognized words.
- It also checks if all the required words are present in the user's message.
- The final probability is returned as an integer percentage.

2. **check_all_messages function:**
    - This function checks all predefined responses and selects the one with the highest probability.
    - It uses the message_probability function to calculate the probability for each predefined response.
    - The responses and their corresponding recognized words are defined in the function.
    - The function returns the best-matching response based on the calculated probabilities.
3. **Predefined Responses:**
    - Greetings: Responds to greetings such as "hello," "hi," "hey," etc.
    - Farewell: Responds to farewell messages like "bye" or "goodbye."
    - How are you: Responds to queries about the bot's well-being.
    - Thank you: Responds to expressions of gratitude.
    - Specific phrases: Responds to specific phrases like "I love code palace."
4. **Longer Responses:**
    - **There are longer responses stored in an external module (long_responses). These responses are triggered by specific keywords like "give advice," "what you eat," and "what you know about AI."**
5. **get_response function:**

- This function takes user input, splits it into words, converts them to lowercase, and then passes them to the check_all_messages function.
- It returns the selected response**.**

6. **Testing:**
    - The script enters a loop where the user can input messages, and the bot responds based on the predefined responses and probabilities.
7. **Overall Flow:**
    - The chatbot evaluates user input, calculates the probability of each predefined response, and selects the one with the highest probability. If no response meets the criteria (probability greater than 1%), it returns a default unknown response.

Code:

```
import re

import long_responses as long


def message_probability(user_message, recognised_words, 
single_response=False, required_words=[]):

    message_certainty = 0

    has_required_words = True


    # Counts how many words are present in each predefined 
message

    for word in user_message:

        if word in recognised_words:

            message_certainty += 1


    # Calculates the percent of recognised words in a user 
message

    percentage = float(message_certainty) / 
float(len(recognised_words))
```

```
    # Checks that the required words are in the string
    for word in required_words:
        if word not in user_message:
            has_required_words = False
            break

    # Must either have the required words, or be a single
response
    if has_required_words or single_response:
        return int(percentage * 100)
    else:
        return 0


def check_all_messages(message):
    highest_prob_list = {}


    # Simplifies response creation / adds it to the dict
    def response(bot_response, list_of_words,
single_response=False, required_words=[]):
        nonlocal highest_prob_list
        highest_prob_list[bot_response] =
message_probability(message, list_of_words, single_response,
required_words)



    # Responses -------------------------------------------------
-------------------------------------------------
```

```
    response('Hello!', ['hello', 'hi', 'hey', 'sup', 'heyo'],
single_response=True)

    response('See you!', ['bye', 'goodbye'],
single_response=True)

    response('I\'m doing fine, and you?', ['how', 'are', 'you',
'doing'], required_words=['how'])

    response('You\'re welcome!', ['thank', 'thanks'],
single_response=True)

    response('Thank you!', ['i', 'love', 'code', 'palace'],
required_words=['code', 'palace'])



    # Longer responses

    response(long.R_ADVICE, ['give', 'advice'],
required_words=['advice'])

    response(long.R_EATING, ['what', 'you', 'eat'],
required_words=['you', 'eat'])

    response(long.R_AI, ['what', 'you', 'know','about','AI'],
required_words=['what ', 'AI'])



    best_match = max(highest_prob_list,
key=highest_prob_list.get)

    # print(highest_prob_list)

    # print(f'Best match = {best_match} | Score:
{highest_prob_list[best_match]}')



    return long.unknown() if highest_prob_list[best_match] < 1
else best_match



# Used to get the response

def get_response(user_input):
```

```
    split_message = re.split(r'\s+|[,;?!.-]\s*',
user_input.lower())

    response = check_all_messages(split_message)

    return response


# Testing the response system

while True:

    print('Bot: ' + get_response(input('You: ')))
```

Long response.py

```
import random

R_EATING = "I don't like eating anything because I'm a bot
obviously!"

R_ADVICE = "If I were you, I would go to the internet and type
exactly what you wrote there!"

R_AI="AI stands for artificial intelligence, which is the
ability of machines or software to perform tasks that normally
require human intelligence, such as reasoning, learning, or
problem-solving."


def unknown():

    response = ["Could you please re-phrase that? ",

                "...",

                "Sounds about right.",

                "What does that mean?"

                ][

        random.randrange(4)]

    return response
```