```python
import re
import random
import requests
import nltk
from nltk.corpus import wordnet
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity

nltk.download("punkt")
nltk.download("wordnet")

gk_data = {
    "who is the president of india": "Droupadi Murmu",
    "what is the capital of india": "New Delhi",
    "what is the largest ocean": "The Pacific Ocean",
    "what is the fastest land animal": "The cheetah",
}

GREET_INPUTS = ("hello", "hi", "greetings", "sup", "what's up", "hey")
GREET_RESPONSES = ["hi", "hey", "hello", "hello there", "I am glad to talk to
you!"]

lemmer = nltk.stem.WordNetLemmatizer()

def LemTokens(tokens):
    return [lemmer.lemmatize(token) for token in tokens]

remove_punct_dict = dict((ord(punct), None) for punct in re.escape(".,;!?"))
def LemNormalize(text):
    return
LemTokens(nltk.word_tokenize(text.lower().translate(remove_punct_dict)))

def greet(sentence):
    for word in sentence.split():
        if word.lower() in GREET_INPUTS:
            return random.choice(GREET_RESPONSES)

def solve_math(expression):
    try:
        result = eval(expression)
        return f"The answer is {result}."
    except:
        return "I couldn't understand the math problem. Please try again."
```

```python
# Fetch weather data
def get_weather(city):
    api_key = "c1b4c10a6ca74a476e23fe6ee62ad4af"
    base_url = "http://api.openweathermap.org/data/2.5/weather"
    try:
        params = {"q": city, "appid": api_key, "units": "metric"}
        response = requests.get(base_url, params=params)
        data = response.json()
        if data["cod"] == 200:
            weather = data["weather"][0]["description"]
            temp = data["main"]["temp"]
            return f"The weather in {city} is currently {weather} with a 
temperature of {temp}°C."
        else:
            return "Sorry, I couldn't find the weather for that location."
    except:
        return "I'm having trouble fetching the weather data. Please try again 
later."


# General knowledge function
def general_knowledge(question):
    question = question.lower()
    if question in gk_data:
        return gk_data[question]
    else:
        return "I don't know the answer to that. Can you teach me?"


# Main chatbot loop
print("BOT: My name is Stark. I can chat, solve math problems, provide weather 
updates, and answer general knowledge questions. Type 'Bye' to exit.")

while True:
    user_response = input("You: ").lower()

    if user_response == "bye":
        print("BOT: Goodbye! Take care <3")
        break

    elif user_response.startswith("solve"):
        # Handle arithmetic problems
        math_problem = user_response.replace("solve", "").strip()
        print("BOT:", solve_math(math_problem))
    elif user_response.startswith("weather"):
        # Handle weather requests
        city = user_response.replace("weather", "").strip()
        print("BOT:", get_weather(city))

    elif greet(user_response):
        # Handle greetings
        print("BOT:", greet(user_response))

    elif user_response in ["thanks", "thank you"]:
        # Handle gratitude
        print("BOT: You're welcome!")

    else:
        # Handle general knowledge or fallback
        gk_response = general_knowledge(user_response)
        if gk_response == "I don't know the answer to that. Can you teach me?":
            print("BOT:", gk_response)
        else:
            print("BOT:", gk_response)
```