```
 # Meet V-Bot: your friend

import nltk
import warnings
warnings.filterwarnings("ignore")
# nltk.download() # for downloading packages
#import tensorflow as tf
import numpy as np
import random
import string # to process standard python strings

f=open('nlp python answer finals.txt','r',errors = 'ignore')
m=open('modules pythons.txt','r',errors = 'ignore')
checkpoint = "./chatbot_weights.ckpt"
#session = tf.InteractiveSession()
#session.run(tf.global_variables_initializer())
#saver = tf.train.Saver()
#saver.restore(session, checkpoint)

raw=f.read()
rawone=m.read()
raw=raw.lower()# converts to lowercase
rawone=rawone.lower()# converts to lowercase
nltk.download('punkt') # first-time use only
nltk.download('wordnet') # first-time use only
sent_tokens = nltk.sent_tokenize(raw)# converts to list of sentences
word_tokens = nltk.word_tokenize(raw)# converts to list of words
sent_tokensone = nltk.sent_tokenize(rawone)# converts to list of sentences
word_tokensone = nltk.word_tokenize(rawone)# converts to list of words


sent_tokens[:2]
sent_tokensone[:2]

word_tokens[:5]
word_tokensone[:5]

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 string.punctuation)
def LemNormalize(text):
    return LemTokens(nltk.word_tokenize(text.lower().translate(remove_punct_dict)))

Introduce_Ans = ["My name is V-Bot.","My name is V-Bot you can called me v.","Im V-Bot :) ","My name is V-Bot. and my nickname is v and i am happy to solve your queries :) "]
GREETING_INPUTS = ("hello", "hi","hiii","hii","hiiii","hiiii", "greetings", "sup", "what's up","hey",)
GREETING_RESPONSES = ["hi", "hey", "hii there", "hi there", "hello", "I am glad! You are talking to me"]
Basic_Q = ("what is python ?","what is python","what is python?","what is python.")
Basic_Ans = "Python is a high-level, interpreted, interactive and object-oriented scripting programming language python is designed to be highly readable It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages."
Basic_Om = ("what is module","what is module.","what is module ","what is module ?","what is module?","what is module in python","what is module in python.","what is module in python?","what is module in python ?")
Basic_AnsM = ["Consider a module to be the same as a code library.","A file containing a set of functions you want to include in your application.","A module can define functions, classes and variables. A module can also include runnable code. Grouping related code into a module makes the code easier to understand and use."]


# Checking for greetings
def greeting(sentence):
    """If user's input is a greeting, return a greeting response"""
    for word in sentence.split():
        if word.lower() in GREETING_INPUTS:
            return random.choice(GREETING_RESPONSES)

# Checking for Basic_Q
def basic(sentence):
    for word in Basic_Q:
        if sentence.lower() == word:
            return Basic_Ans

# Checking for Basic_QM
def basicM(sentence):
    """If user's input is a greeting, return a greeting response"""
    for word in Basic_Om:
        if sentence.lower() == word:
            return random.choice(Basic_AnsM)

# Checking for Introduce
def IntroduceMe(sentence):
    return random.choice(Introduce_Ans)


from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity


# Generating response
def response(user_response):
    robo_response=''
    sent_tokens.append(user_response)
    TfidfVec = TfidfVectorizer(tokenizer=LemNormalize, stop_words='english')
    tfidf = TfidfVec.fit_transform(sent_tokens)
    vals = cosine_similarity(tfidf[-1], tfidf)
    idx=vals.argsort()[0][-2]
    flat = vals.flatten()
    flat.sort()
    req_tfidf = flat[-2]
    if(req_tfidf==0):
        robo_response=robo_response+"I am sorry! I don't understand you"
        return robo_response
    else:
        robo_response = robo_response+sent_tokens[idx]
        return robo_response

# Generating response
def responseone(user_response):
    robo_response=''
    sent_tokensone.append(user_response)
    TfidfVec = TfidfVectorizer(tokenizer=LemNormalize, stop_words='english')
    tfidf = TfidfVec.fit_transform(sent_tokensone)
    vals = cosine_similarity(tfidf[-1], tfidf)
    idx=vals.argsort()[0][-2]
    flat = vals.flatten()
    flat.sort()
    req_tfidf = flat[-2]
    if(req_tfidf==0):
        robo_response=robo_response+"I am sorry! I don't understand you"
        return robo_response
    else:
        robo_response = robo_response+sent_tokensone[idx]
        return robo_response


def chat(user_response):
```

```
def chat(user_response):
    user_response=user_response.lower()
    keyword = " module "
    keywordone = " module"
    keywordsecond = "module "

    if(user_response!='bye'):
        if(user_response=='thanks' or user_response=='thank you' ):
            flag=False
            #print("V-Bot: You are welcome..")
            return "You are welcome.."
        elif(basicM(user_response)!=None):
            return basicM(user_response)
        else:
            if(user_response.find(keyword) != -1 or user_response.find(keywordone) != -1 or user_response.find(keywordsecond) != -1):
                #print("V-Bot: ",end="")
                #print(responseone(user_response))
                return responseone(user_response)
                sent_tokensone.remove(user_response)
            elif(greeting(user_response)!=None):
                #print("V-Bot: "+greeting(user_response))
                return greeting(user_response)
            elif(user_response.find("your name") != -1 or user_response.find(" your name") != -1 or user_response.find("your name ") != -1 or user_response.find(" your name ") != -1):
                return IntroduceMe(user_response)
            elif(basic(user_response)!=None):
                return basic(user_response)
            else:
                #print("V-Bot: ",end="")
                #print(response(user_response))
                return response(user_response)
                sent_tokens.remove(user_response)

    else:
        flag=False
        #print("V-Bot: Bye! take care..")
        return "Bye! take care.."
```