CODE ALPHA – INTERN PROJECTS

1. Basic Chatbot

* Create a text-based chatbot that can have conversations with users. You can use natural language processing libraries like NLTK to make your chatbot more conversational.

STEP 1 : Install PYTHON (3.10/64 bit )

STEP 2: Make Virtual Environment, (to develop project individually )

In Command Prompt –

1. python -m venv env ( creating a folder env i.e virtual environment )

2. C:\Users\Lenovo\env\Scripts\activate (activating the VENV )

3. pip2 install nltk1 (installs that particular module)

STEP 3 : DOWNLOAD NLTK data for tokenization

1. In python IDLE create a file chatbot.py
2. Type import nltk ; nltk.download(‘punkt\_tab’)3 ;
3. # the nltk.download line and
4. type

import nltk

#nltk.download('punkt\_tab')4

from nltk.tokenize import word\_tokenize

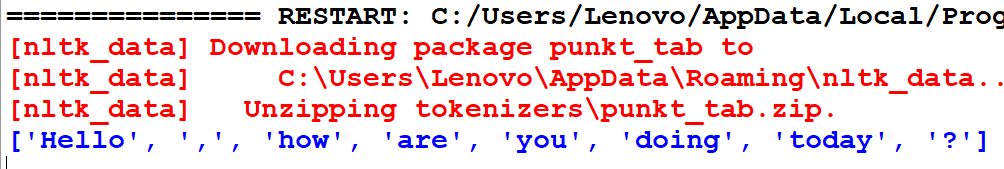
text = "Hello, how are you doing today?"

tokens = word\_tokenize(text)

print(tokens)

(to test for working )

**OUTPUT:**



STEP 4 : Function to handle responses and give your code:

import nltk

from nltk.tokenize import word\_tokenize

import random

import time

# Download necessary NLTK resources (if not already done)

# nltk.download('punkt\_tab')

# Function to process and tokenize input

def process\_input(text):

tokens=word\_tokenize(text.lower()) #Convert input to lowercase for case-insensitive matching

return tokens

#Simple response function with using tokenization

def respond():

print(" HELLO ! THIS IS A SIMPLE CHATBOX\n\n\t Instructions: \n")

print("1.IT CAN RESPOND TO GREETINGS\n"

"2.GIVE MOTIVATIONAL QUOTES\n"

"3.PROVIDE CURRENT TIME\n"

"4.MOVIE RECOMMENDATIONS\n"

"5.PROVIDE HELP\n"

"6.A DEFAULT RESPONSE TO UNKNOWN QUERIES\n"

"7.TO EXIT GIVE 1 AS YOUR INPUT \n")

while True:

user\_input=input("You: ")

tokens=process\_input(user\_input)

# Exit the CHATBOT when user gives 1 as input

if '1' in tokens:

print("Exiting!")

break

#Greetings Response

elif 'hello' in tokens or 'hi' in tokens or 'hey' in tokens:

print("Hey there!")

elif 'bye' in tokens or 'good bye' in tokens:

print("TAKE CARE ! SEE YOU LATER")

elif 'sup' in tokens or 'how is it going' in tokens:

print("I'm fine, how about you!")

elif 'fine' in tokens or 'good' in tokens:

print("ha ! Good to know ")

#Motivstional response

elif 'sad' in tokens or 'bad' in tokens or 'motivation' in tokens:

quotes=["DONT WORRY","IT WILL HAPPEN GOOD","IT WILL CHANGE","BELEIVE YOURSELF"]

print(random.choice(quotes))

#time response

elif 'what time' in tokens or 'time' in tokens:

print("The current time is: ", time.strftime("%H:%M:%S"))

#Help response

elif 'help' in tokens or 'need help' in tokens:

print("I'm here to help!")

#movie response

elif 'movie' in tokens or 'suggest' in tokens:

m=["BABY JOHN - HINDI","G.O.A.T - TAMIL","KALKI 2898 AD - TELUGU","MANJUMMEL BOYS - MALAYALAM","K.G.F - KANNADA"]

print(random.choice(m))

# Default response if none matches

else:

print("Sorry! I'm not programmed for your query")

# Start the conversation

respond()

**Glossary –**

1. **nltk -**NLTK (Natural Language Toolkit) is a Python library for working with human language data (text). It provides tools for tasks like:

* Tokenization (splitting text into words or sentences)
* Stemming and lemmatization (reducing words to their root forms)
* Part-of-speech tagging
* Parsing and syntax analysis
* Building machine learning models for text

It's widely used for learning and experimenting with natural language processing.

1. **PIP -** Pip is a package manager for Python that lets you install, update, and manage Python libraries and dependencies from the Python Package Index (PyPI)
2. punkt is the tokenizer designed for general text, like the kind used in your chatbot. It splits text into words and sentences and is ideal for natural language processing tasks.

punkt\_tab, on the other hand, is designed for tabular data and isn't necessary for typical text-based chatbot projects.

1. nltk is the overall library.

tokenization is a specific functionality within nltk for splitting text into smaller components.

Word\_tokenization is the function

1. It is in loop because till I give “1” it will be a conversational chatbot.
2. Have used if and elifs condition.
3. Have provided instructions in the start
4. Hangman Game

* Design a text-based Hangman game. The program selects a random word, and the player guesses one letter at a time to uncover the word. You can set a limit on the number of incorrect guesses allowed.

STEP 1 : Install PYTHON (3.10/64 bit )

STEP 2 : No need of module installation as its already built – in

import random1

choices=['raspberry', 'swift', 'python', 'kotlin', 'java']

word=random.choice(choices)

print("WELCOME TO THE HANGMAN GAME\nLET'S START!!!")

print("NUMBER OF LETTERS: ", len(word))2

chances=8

guessed=[]

correct=[]

while chances>0:

letter=input("\nGuess a letter: ")

if letter in guessed:

print("You already guessed that letter! Try again.")

continue

guessed.append(letter)

if letter in word:

position=[i + 1 for i, char in enumerate(word)3 if char == letter]

print("Good! The letter'"+letter+"'is in the word at position: "+str(position))

correct.append(letter)

else:

print("Sorry, the letter'"+letter+"'is not in the word")

chances=chances-1

print("Remaining chances: "+str(chances))

if all4(char in correct for char in word):

print("\nCongratulations! You guessed the word: "+ word)

break

if not all(char in correct for char in word):

print("\nGame Over! The man is HANGED!")

print("The word was: "+word)

**GLOSSARY :**

1. **Random – random.choice(choices)**

Imports the random module, which allows us to use the random.choice() function to select a random word from the given choices and stores in word variable.

1. **Len(word) –** Analyses the length of word.
2. **enumerate()**

Returns both the index and value of items in an iterable. Used to find positions of guessed letters in the word.

1. **all() Function:**

Checks if all elements in an iterable are True. Used here to verify if all letters in the word are guessed.

1. It keeps guessing till chances become 0 and have used while loops.
2. **Task Automation using Python Scripts**

* Identify a repetitive task in your workflow and create Python scripts to automate it. This could include tasks like file organization, data cleaning, or system maintenance.

STEP 1 : Install PYTHON (3.10/64 bit )

STEP 2: No need of installing any module as we will use a built-in one.

import os1

import time2

print("TASK AUTOMATION USING PYTHON SCRIPTS\n")

print("\tRENAMING THE FOLDER FILES\n")

print("1. The files in the folder starting with QP will change as Question Paper")

print("2. The files in the folder which have spaces will change to underscore")

print("3. It checks for every 5 minutes")

#ask the user to input the folder path

folder\_path=input("Enter the folder path to automate: ")

#list all files in the folder

while True:

files=os.listdir(folder\_path)3

for a in files:

try:

#replace spaces with underscores

new\_name=a.replace(" ","\_")

#replace QP with 'Question Paper

new\_name=a.replace("QP","Question Paper")5

old\_path=os.path.join(folder\_path,a)4

new\_path=os.path.join(folder\_path,new\_name)

#rename the file

os.rename(old\_path,new\_path)

print("Renamed: ",a,"AS: ",new\_name)

except Exception as e:

print("Error renaming",a,":",e)

time.sleep(20)7

**Glossary** :

1. **os Module**

Provides a way to interact with the operating system. Used here for listing files in a folder (os.listdir()) and renaming files (os.rename()).

1. **time Module**

Used to manage time-related tasks. In this code, time.sleep() is used to pause the program for 20 seconds between iterations, simulating the 5-minute check interval mentioned in the comment.

1. **os.listdir()**

Lists all files and directories in the specified folder. Here, it retrieves the files to be renamed.

1. **os.path.join()**

Joins folder paths and filenames into a complete file path, ensuring the correct format for different operating systems.

1. **String Methods:**

replace(): Replaces a specified substring with another.

Used here to replace spaces with underscores and "QP" with "Question Paper" in the file names.

1. Using a infinite while loop, to make sure it works every 20 seconds.
2. **time.sleep(20)**

Pauses the program for 20 seconds before the next iteration, making it wait before checking the folder again.