# AIML252EXP9-Integrate a speech recognition ML model to allow voice input for Arithmetic calculations

import sounddevice as sd

import soundfile as sf

import speech\_recognition as sr

import pyttsx3

from word2number import w2n

import sympy as sp

import math

engine = pyttsx3.init()

def speak(text):

engine.say(text)

engine.runAndWait()

def record\_audio(filename='input.wav', duration=5, fs=44100):

print("Recording...")

recording = sd.rec(int(duration \* fs), samplerate=fs, channels=1)

sd.wait()

sf.write(filename, recording, fs)

print("Recording saved!")

def get\_audio():

record\_audio()

r = sr.Recognizer()

with sr.AudioFile('input.wav') as source:

audio = r.record(source)

try:

text = r.recognize\_google(audio)

print(f"You said: {text}")

return text

except sr.UnknownValueError:

speak("Sorry, I didn't catch that.")

return None

except sr.RequestError:

speak("Speech service is down.")

return None

def parse\_expression(speech\_input):

replacements = {

'plus': '+',

'minus': '-',

'times': '\*',

'multiplied by': '\*',

'divided by': '/',

'over': '/',

'into': '\*',

'mod': '%',

'modulus': '%',

'power': '\*\*',

'to the power of': '\*\*',

'square root of': 'sqrt',

'log of': 'log',

'sin': 'sin()',

'cos': 'cos()',

'tan': 'tan()',

'pi': 'pi',

'e': 'E',

'x': '\*' # Handle 'x' as multiplication

}

speech\_input = speech\_input.lower()

for word, symbol in replacements.items():

speech\_input = speech\_input.replace(word, symbol)

for func in ['sin', 'cos', 'tan', 'sqrt', 'log']:

speech\_input = speech\_input.replace(f'{func} ', f'{func}(')

speech\_input = speech\_input.replace('sin(', 'sin()').replace('cos(', 'cos()').replace('tan(', 'tan()')

# Add closing parentheses for each trigonometric function

for func in ['sin', 'cos', 'tan', 'sqrt', 'log']:

speech\_input = speech\_input.replace(f'{func}(', f'{func}(') + ')'

tokens = speech\_input.split()

converted\_tokens = []

for token in tokens:

try:

converted = str(w2n.word\_to\_num(token)) # Converts word to number

converted\_tokens.append(converted)

except:

converted\_tokens.append(token) # Keeps the token as is if it's not a number word

parsed\_expr = ' '.join(converted\_tokens) # Joins the tokens back into a single string

print(f"Parsed Expression: {parsed\_expr}") # Debug print

return parsed\_expr

def main():

speak("Please say your scientific arithmetic expression.")

speech\_input = get\_audio()

if speech\_input:

expression = parse\_expression(speech\_input.lower())

try:

# Attempt to evaluate the expression

result = sp.sympify(expression)

speak(f"The result is {result}")

print(f"Result: {result}") # Print result to the console

except Exception as e:

print(f" Error while evaluating: {e}")

speak("Sorry, I couldn't calculate that.")

if \_\_name\_\_ == "\_\_main\_\_":

main()

OUTPUT:

