KALYANI GOVERNMENT ENGINEERING COLLEGE

CA2 ASSIGNMENT ON

Speech processing using PYTHON

NAME-INDRAKSHI CHATTERJEE

ROLL NO-10201622079 REG NO-221020110378

YEAR-FOURTH SEMESTER-SEVENTH

SUBJECT- ARTIFICIAL INTELLIGENCE

SUBJECT CODE- OE-EE-701A

DEPARTMENT- ELECTRICAL ENGINEERING

SESSION-2024-2025

INTRODUCTION

Speech is the most natural way for humans to communicate, and enabling machines to understand and respond to speech has been one of the primary goals of Artificial Intelligence (AI). **Speech processing** is the technology that allows computers to listen, interpret, and generate spoken language. It involves tasks such as **speech recognition** (converting spoken words into text), **speech synthesis** (converting text into spoken words), and **language understanding** (interpreting the meaning of speech).

With the advancement of machine learning and natural language processing (NLP), speech processing has become a fundamental part of everyday technology. Examples include **virtual assistants** like Siri, Alexa, and Google Assistant, **voice-controlled smart devices**, **customer service chatbots**, and **accessibility tools** for the visually impaired.

Python, being a versatile programming language with a wide ecosystem of libraries, provides powerful tools for speech processing. Libraries such as **SpeechRecognition**, **pyttsx3**, and **pyaudio** allow developers to capture voice input, analyze it, and respond with synthesized speech. Additionally, Python can integrate with services like **Wikipedia API**, **math libraries**, and **date-time utilities**, enabling the creation of intelligent, voice-driven assistants that can answer questions, perform calculations, and provide real-time information.

This project demonstrates how speech processing can be implemented using Python to build a **basic AI voice assistant**. The assistant continuously listens to the user, processes their queries, provides answers in both text and voice, and exits when commanded.

ALGORITHM

Step 1: Import Libraries

- Import speech_recognition for speech-to-text.
- Import pyttsx3 for text-to-speech.
- Optionally, import wikipedia, datetime, or math for answering questions.

Step 2: Initialize Components

- Create a recognizer object to listen to microphone input.
- Initialize the text-to-speech engine.

Step 3: Capture Speech

- Use the microphone to listen to the user's speech.
- Convert the audio into text using Google Speech Recognition API.

Step 4: Process Input

- Check if the query is a simple question (e.g., time, date, math, greetings).
- If it's a knowledge-based query, fetch results from Wikipedia.

Step 5: Generate Response

- Convert the text-based answer into speech using pyttsx3.
- Print the answer as text for reference.

Step 6: Exit Condition

- If the user says "bye", stop the program.

PROBLEM STATEMENT:

Write a Python code to create a voice-based AI assistant that continuously listens to user speech, understands questions, answers simple/general knowledge/math queries directly, fetches detailed information from Wikipedia only when required, and always replies in both text and speech without repeating or getting stuck.

Python code:

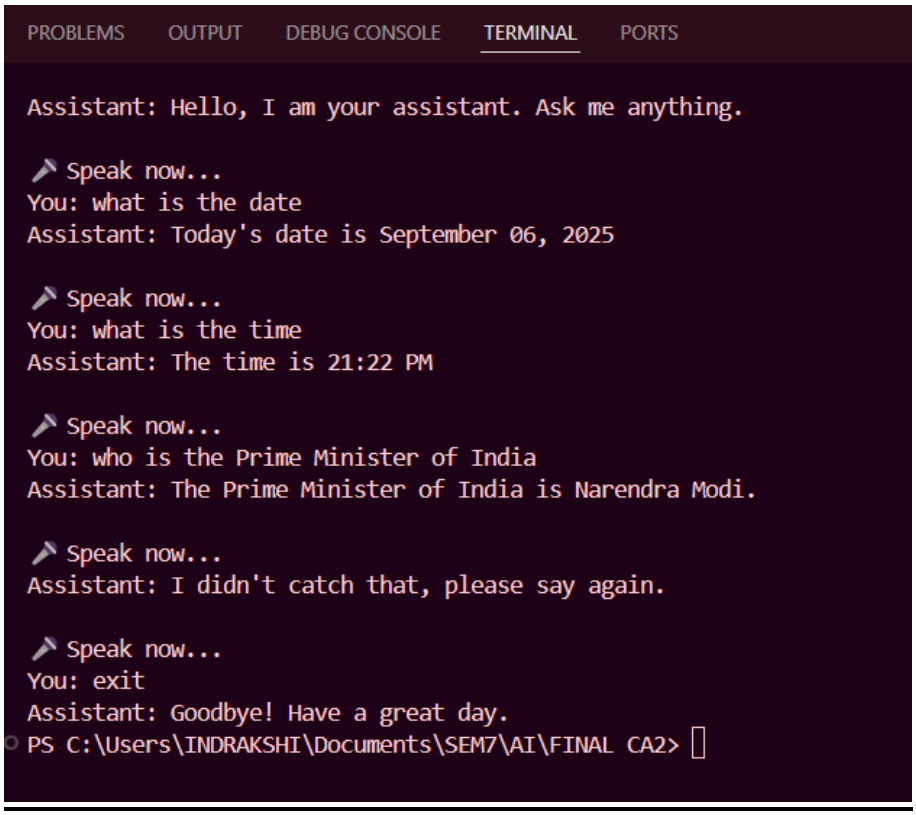
```
import speech_recognition as sr
import pyttsx3
import wikipedia
import re
from sympy import sympify
from datetime import datetime


recognizer = sr.Recognizer()


def speak(text):
    """Force speech each time"""
    print("Assistant:", text)
    try:
        engine = pyttsx3.init()  # reinitialize every call
        voices = engine.getProperty("voices")
        if voices:  # pick first available voice
            engine.setProperty("voice", voices[0].id)
        engine.setProperty("rate", 170)
        engine.say(text)
        engine.runAndWait()
        engine.stop()  # flush engine
    except Exception as e:
        print("[TTS ERROR]", e)
```

```
def simple_answer(question):
    q = question.lower().strip()

    # Date and Time
    if "time" in q:
        return f"The time is {datetime.now().strftime('%H:%M %p')}"
    if "date" in q:
        return f"Today's date is {datetime.now().strftime('%B %d, %Y')}"

    # Predefined
    short_answers = {
        "what is the capital of india": "The capital of India is New Delhi.",
        "who is the prime minister of india": "The Prime Minister of India is Narendra Modi.",
        "what color is the sky": "The sky is usually blue during the day and dark at night.",
        "who are you": "I am your AI voice assistant.",
        "how are you": "I am doing great, thank you!",
        "what is your name": "You can call me your assistant."
    }
    if q in short_answers:
        return short_answers[q]

    # Math (sympy)
    try:
        expr = sympify(q)
        return f"The answer is {expr.evalf()}"
    except:
        pass
```

```
    # Math regex
    match = re.search(r'(\d+)\s*(\+|plus|\-|minus|\*|x|times|/|divided by)\s*(\d+)', q)
    if match:
        a, op, b = match.groups()
        a, b = float(a), float(b)
        if op in ["+", "plus"]:
            return f"The answer is {a+b}"
        elif op in ["-", "minus"]:
            return f"The answer is {a-b}"
        elif op in ["*", "x", "times"]:
            return f"The answer is {a*b}"
        elif op in ["/", "divided by"]:
            return f"The answer is {a/b}"

    # Wikipedia fallback
    try:
        return wikipedia.summary(q, sentences=2)
    except:
        return "Sorry, I don't know that yet."

def listen():
    with sr.Microphone() as source:
        recognizer.adjust_for_ambient_noise(source, duration=0.5)
        print("\n🎤 Speak now...")
        audio = recognizer.listen(source)
    try:
        text = recognizer.recognize_google(audio)
```

```
        print("You:", text)
        return text.lower()
    except:
        return None

def main():
    speak("Hello, I am your assistant. Ask me anything.")
    while True:
        query = listen()

        if not query:
            speak("I didn't catch that, please say again.")
            continue

        if any(word in query for word in ["bye", "exit", "quit"]):
            speak("Goodbye! Have a great day.")
            break

        answer = simple_answer(query)
        speak(answer)

if __name__ == "__main__":
    main()
```

OUTPUT:



RESULT:

The Python code take user input through microphone and replies in the form of speech as well as text to answer the user's questions.

CONCLUSION

Speech processing using Python demonstrates how computers can interact with humans in a natural, intuitive way. By combining **speech-to-text**, **text-to-speech**, and **knowledge-based querying**, it is possible to build an AI assistant capable of understanding questions, retrieving information, and responding conversationally.

The project shows that with only a few Python libraries, one can implement key features of modern intelligent systems, such as:

- **Voice recognition** to capture human speech.
- **Natural language understanding** to interpret user queries.
- **Text-to-speech synthesis** to respond in a human-like manner.
- **Integration with external knowledge bases** (Wikipedia, math functions, date & time) to answer a wide variety of questions.

The results highlight how Python serves as an excellent platform for prototyping AI-driven speech applications. While the system built here is simple compared to commercial products like Alexa or Siri, it represents the **core principles** behind such assistants. Future improvements could involve adding **contextual memory**, **sentiment analysis**, or integration with **IoT devices** for practical real-world use.

In conclusion, speech processing bridges the gap between humans and machines, moving technology closer to natural communication. Python makes this process accessible, efficient, and scalable for both students and professionals aiming to explore the field of **AI-driven voice technologies**.