**VIVA (Virtual Interactive Voice Assistant)**

Developed a Python-based voice assistant using SpeechRecognition, pyttsx3, and pywhatkit. Implemented voice commands for Google search, YouTube playback, time announcements, and application control (Chrome, etc.). Integrated text-to-speech for real-time responses. Enhanced accuracy with ambient noise reduction and error handling.

Python script is a simple voice assistant named **VIVA (Virtual Interactive Voice Assistant)** that can recognize voice commands and execute tasks like:

* Opening Chrome
* Searching Google
* Telling the time
* Playing videos on YouTube
* Opening YouTube
* Introducing itself
* Exiting on command

**Libraries Used and Their Purpose:**

1. **speech\_recognition (sr):**

 Provides speech recognition capabilities.

 Converts spoken language into text using online services (Google, etc.).

 Functions used:

* **sr.Recognizer()** → Creates a recognizer instance to process audio.
* **recognizer.adjust\_for\_ambient\_noise(source, duration=0.5)** → Adjusts to background noise.
* **recognizer.listen(source, timeout=5, phrase\_time\_limit=5)** → Listens for input from the microphone.
* **recognizer.recognize\_google(recordedaudio, language='en\_US')** → Converts audio to text using Google’s API.

One of the most popular Python libraries for recognizing speech. It provides support for several engines and APIs, such as Google Web Speech API, Microsoft Bing Voice Recognition, and IBM Speech to Text. It's known for its ease of use and flexibility, making it a great starting point for beginners and experienced developers alike.

**Key Components of SpeechRecognition**

1. **Recognizer Class (sr.Recognizer())**
   * This is the core of the library, used to convert speech into text.
2. **Microphone Class (sr.Microphone())**
   * Captures real-time audio input from the user's microphone.
3. **AudioFile Class (sr.AudioFile())**
   * Reads pre-recorded audio files for speech recognition.
4. **Recognition Methods**
   * Supports different speech recognition services like:
     + Google Web Speech API (recognizer.recognize\_google())
     + Sphinx (Offline) (recognizer.recognize\_sphinx())
     + IBM Watson, Microsoft Bing, etc.
5. **Background noise can affect accuracy** (can be mitigated with adjust\_for\_ambient\_noise).

**Key Functions in SpeechRecognition**

| **Function** | **Description** |
| --- | --- |
| recognizer.adjust\_for\_ambient\_noise(source, duration=1) | Adjusts to background noise before capturing speech. |
| recognizer.listen(source, timeout=5, phrase\_time\_limit=5) | Listens for speech with optional time limits |
| recognizer.recognize\_google(audio, language='en-US') | Converts audio to text using Google's API. |
| recognizer.recognize\_sphinx(audio) | Offline speech recognition using CMU Sphinx. |
| recognizer.record(source, duration=5) | Records audio for a fixed duration. |
| recognizer.energy\_threshold | Controls sensitivity to background noise. |

### 2. ****pyttsx3****

* Handles text-to-speech (TTS) conversion.
* Works offline and supports multiple voices.
* Functions used:
  + pyttsx3.init() → Initializes the speech engine.
  + engine.setProperty('voice', voices[0].id) → Selects a voice.
  + engine.say(message) → Converts text to speech.
  + engine.runAndWait() → Processes the speech output.

## ****Key Features of pyttsx3****

**•Works Offline** – No internet required.  
**•Supports Multiple Voices** – Male and female voices available.  
**•Adjustable Speech Rate** – Slow down or speed up speech.  
**•** **Volume Control** – Increase or decrease volume.  
**•Cross-Platform Support** – Works on Windows, macOS, and Linux.

**Key Functions in pyttsx3**

| **Function** | **Description** |
| --- | --- |
| pyttsx3.init() | Initializes the TTS engine. |
| engine.say("Text") | Converts text to speech. |
| engine.runAndWait() | Processes the speech queue. |
| engine.getProperty('rate') | Gets the speaking speed (default: 200 wpm). |
| engine.setProperty('rate', 150) | Adjusts speech rate (lower = slower). |
| engine.getProperty('volume') | Gets the volume level (0.0 to 1.0). |
| engine.setProperty('volume', 0.8) | Adjusts volume level. |
| engine.getProperty('voices') | Retrieves available voices. |
| engine.setProperty('voice', voices[1].id) | Changes the voice. |
| engine.save\_to\_file("text", "file.mp3") | Saves speech to an audio file. |

### 3. ****datetime****

* Retrieves the current time and date.
* Function used:
  + datetime.datetime.now().strftime('%I:%M %p') → Gets the current time in HH:MM AM/PM format.

### 4. ****subprocess****

* Launches external applications.
* Function used:
  + subprocess.Popen(r"C:\Program Files (x86)\Google\Chrome\Application\chrome.exe", shell=True) → Opens Chrome.

The subprocess module in Python is used to **execute external commands and programs** from within a Python script. It allows interaction with the system shell and can be used to start applications, run shell commands, and capture command output.

**Key Functions in subprocess**

| **Function** | **Description** |
| --- | --- |
| subprocess.run(command, capture\_output=True, text=True) | Runs a command and captures output. |
| subprocess.Popen(command) | Starts a command **without waiting** for it to complete. |
| subprocess.call(command) | Runs a command and **waits** for it to finish. |
| subprocess.check\_output(command) | Runs a command and **returns its output**. |
| subprocess.PIPE | Captures standard input/output from a process. |
| subprocess.communicate(input="text") | Sends input to a process and retrieves output. |

**5. pywhatkit**

**•** Provides additional automation features.

**•** Functions used:

* pywhatkit.search(query) → Performs a Google search.
* pywhatkit.playonyt(query) → Plays a YouTube video based on a search query.

**Key Functions in pywhatkit**

| **Function** | **Description** |
| --- | --- |
| sendwhatmsg("+1234567890", "Hello!", 14, 30) | Sends a WhatsApp message at a scheduled time. |
| sendwhatmsg\_instantly("+1234567890", "Hi!") | Sends a WhatsApp message instantly. |
| search("Python programming") | Opens Google search for the given query. |
| playonyt("Despacito") | Plays a YouTube video. |
| text\_to\_handwriting("Hello!", "output.png") | Converts text to a handwritten image. |
| send\_mail(sender, password, receiver, subject, message) | Sends an email. |
| info("Python", lines=3) | Fetches a Wikipedia summary. |

### 6. ****webbrowser****

* Opens web pages in the default browser.
* Function used:
  + webbrowser.open('https://www.youtube.com') → Opens YouTube.

In Python, webbrowser module is a convenient web browser controller. It provides a high-level interface that allows displaying Web-based documents to users.

webbrowser can also be used as a CLI tool. It accepts a URL as the argument with the following optional parameters: -n opens the URL in a new browser window, if possible, and -t opens the URL in a new browser tab.

### 7. ****sys****

* Provides system-specific functions.
* Function used:
  + sys.exit() → Exits the program when the user says "exit" or "quit."

 Allows **command-line argument handling**.

 Useful for **exiting scripts** in case of errors.

 Provides **system information** like Python version & OS..

**Key Functions in sys**

| **Function** | **Description** |
| --- | --- |
| sys.argv | List of command-line arguments. |
| sys.exit() | Exits the program. |
| sys.version | Returns the Python version. |
| sys.platform | Identifies the OS (Windows, Linux, macOS). |
| sys.path | Lists module search paths. |
| sys.stdout | Redirects print output. |
| sys.exc\_info() | Returns exception details. |

**CODE EXPLAINATION:**

### ****1. Initialize Text-to-Speech (TTS) Engine****

engine = pyttsx3.init()

voices = engine.getProperty('voices')

engine.setProperty('voice', voices[0].id)

* This initializes pyttsx3 for text-to-speech and sets the voice.

### ****2. Initialize Speech Recognizer****

recognizer = sr.Recognizer()

* Creates an instance of the speech recognizer.

### ****3. Function: speak()****

def speak(message):

"""Speaks the provided message."""

engine.say(message)

engine.runAndWait()

* Converts text to speech and speaks it out loud.

### ****4. Function: cmd() - Main Logic****

def cmd():

"""Listens for a command and executes it."""

* **Step 1: Listen for a command**

with sr.Microphone() as source:

print("Clearing background noises... Please wait.")

recognizer.adjust\_for\_ambient\_noise(source, duration=0.5)

print('Listening...')

* Adjusts for background noise and starts listening.
* **Step 2: Recognize speech**

recordedaudio = recognizer.listen(source, timeout=5, phrase\_time\_limit=5)

text = recognizer.recognize\_google(recordedaudio, language='en\_US').lower()

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

* Converts spoken audio into text.
* **Step 3: Handle errors**

except sr.UnknownValueError:

print("Sorry, I didn't catch that. Could you repeat?")

speak("Sorry, I didn't catch that. Could you repeat?")

return

* Handles cases where speech is unclear or unrecognized.

### ****5. Execute Commands Based on Recognized Speech****

* **Open Chrome**

if 'chrome' in text:

speak('Opening Chrome...')

subprocess.Popen(r"C:\Program Files (x86)\Google\Chrome\Application\chrome.exe", shell=True)

* **Google Search**

elif 'search' in text:

query = text.replace('search', '').strip()

if query:

speak(f'Searching {query} on Google...')

pywhatkit.search(query)

* **Tell Time**

elif 'time' in text:

current\_time = datetime.datetime.now().strftime('%I:%M %p')

print(f'The time is {current\_time}')

speak(f'The time is {current\_time}')

* **Play YouTube Video**

elif 'play' in text:

query = text.replace('play', '').strip()

if query:

speak(f'Playing {query} on YouTube...')

pywhatkit.playonyt(query)

* **Open YouTube**

elif 'youtube' in text:

speak('Opening YouTube...')

webbrowser.open('https://www.youtube.com')

* **Introduce Itself**

elif "your name" in text:

speak("Myself Viva.")

* **Exit the Program**

elif 'exit' in text or 'quit' in text:

speak('Goodbye!')

sys.exit()

* **Handle Unrecognized Commands**

else:

speak("I didn't understand that command. Please try again.")

### ****6. Run the Assistant****

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

speak("Hello! I am Virtual Interactive Voice Assistant. Simply VIVA. How can I assist you today?")

while True:

cmd()

* Starts the assistant and keeps it running in a loop.