State-of-the-Art Jarvis Implementation in Python

In this presentation exploring the development of a cutting-edge Jarvis implementation in Python. This presentation will delve into the core functionalities, integration of key technologies, and the future of human-AI collaboration.

The Rise of Virtual Assistants

Convenience

Virtual assistants provide hands-free control over various devices and tasks, streamlining daily routines.

Efficiency

They automate tasks, freeing up time for more important activities, increasing productivity.

Personalized Experiences

They learn user preferences and offer customized recommendations for an enhanced user experience.

Accessibility

Virtual assistants make technology more accessible for individuals with disabilities, enabling them to interact with devices seamlessly.

Jarvis' Core Functionalities

Information Retrieval

Jarvis can access and retrieve information from various sources, including web searches and databases.

Task Management

Jarvis can manage tasks, set reminders, and schedule appointments, ensuring efficiency and organization.

Entertainment

Jarvis can play music, videos, and podcasts, offering entertainment and relaxation.

Smart Home Control

Jarvis can control smart home devices, adjusting lighting, temperature, and appliances.

Integrating Speech Recognition

Speech-to-Text

Jarvis converts spoken words into text, allowing for voice-based commands and interactions.

Error Handling

Jarvis handles noisy environments and unclear speech, ensuring accurate transcription.

Harnessing the Power of Text-to-Speech

1

Synthesized Voice

Jarvis generates human-like speech, providing clear and understandable responses.

2

Voice Customization

Jarvis allows users to personalize the voice, adjusting the tone, pitch, and accent.

Seamless Wikipedia Integration

1. **Knowledge Base Access**

   Jarvis connects to Wikipedia's vast database, providing access to a wealth of information.

2. **Query Processing**

   Jarvis intelligently analyzes user queries and retrieves relevant information from Wikipedia.

3. **Information Presentation**

   Jarvis presents information in a concise and understandable manner, using text and multimedia elements.

Datetime and System Interaction

1

Time Management

Jarvis helps users manage their time, set reminders, and schedule appointments efficiently.

2

System Control

Jarvis can control various system functions, such as opening applications and websites.

3

Information Retrieval

Jarvis can access and retrieve information about current time, date, and other system details.

Importing Libraries

```
import pyttsx3
import speech_recognition as sr
import datetime
import wikipedia
import sys
```

Importing

- **pyttsx3:** A text-to-speech conversion library that allows the assistant to speak responses.
- **speech recognition:** A library that enables the assistant to listen to and recognize spoken commands.
- **datetime:** A module to work with dates and times, used here to get the current time.
- **Wikipedia:** A library to fetch summaries from Wikipedia based on user queries.
- **sys:** A module that provides access to system-specific parameters and functions, used here to exit the program.

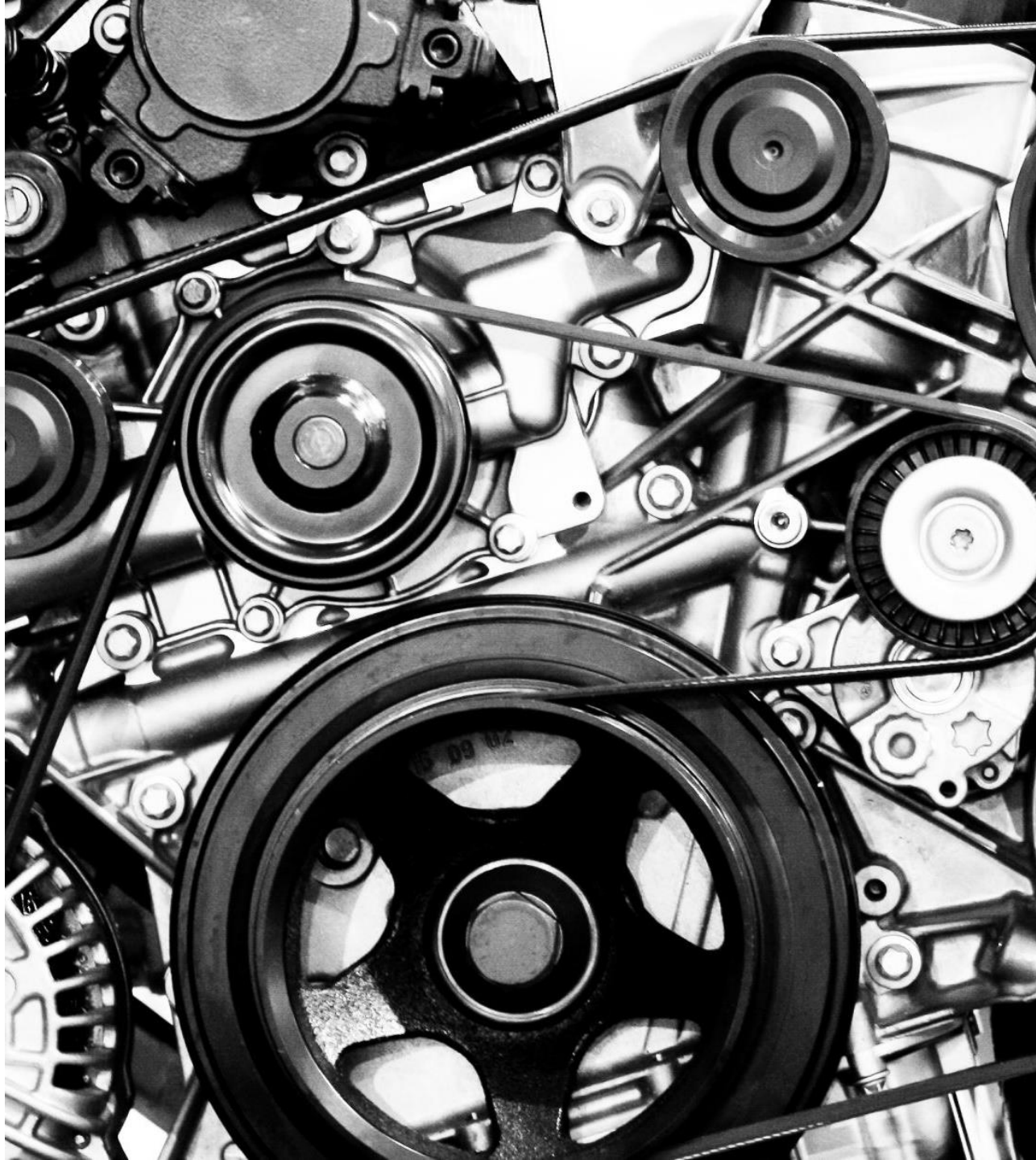
Creating Engine Object

```
engine = pyttsx3.init()
```

Initializing:

`engine = pyttsx3.init():`

Initializes the text-to-speech engine.



```
def speak(text):
    engine.say(text)
    engine.runAndWait()
```

Function for converting text to speech

Speak Function

speak(text): Takes a string text as input and uses the text-to-speech engine to speak it out loud.

Listening Function;

```python
def listen():
    recognizer = sr.Recognizer()
    with sr.Microphone() as source:
        print("Listening...")
        recognizer.adjust_for_ambient_noise(source)
        audio = recognizer.listen(source)

    try:
        print("Recognizing...")
        query = recognizer.recognize_google(audio, language='en-in')
        print(f"User said: {query}")
        return query.lower()
    except:
        print("Sorry, I didn't catch that. Please try again.")
        return None
```

Listener function in Jarvis

- Initializes a ***speech recognizer*** and listens for audio input from the microphone.
- ***Adjusts for ambient noise*** to *improve recognition accuracy*.
- Attempts to recognize the spoken audio using Google’s speech recognition service.
- Returns the recognized text in ***lowercase*** or None if recognition *fails*.

Function for time;

```
def tell_time():
    time = datetime.datetime.now().strftime("%H:%M:%S")
    speak(f"The current time is {time}")
```

Tell_time function

- **tell_time():**Gets the current time and speaks it out loud in a formatted string.

```
def search_wikipedia(query):
    query = query.replace("wikipedia", "")
    result = wikipedia.summary(query, sentences=1)
    speak(result)
```

Jarvis Search Wikipedia Function

Wikipedia function

- **search_wikipedia(query):**Takes a query string, removes the word **"wikipedia"** from it, and fetches a summary from Wikipedia.
- Speaks the summary out loud.

Function to Quit the program

```
def close_program():
    speak("Goodbye!")
    sys.exit()
```

close_program():

close_program():

Speaks a goodbye message and exits the program using sys.exit().

Main function with a loop and if-else condition:

```python
def main():
    speak("Hello, I am Jarvis. How can I assist you today?")

    while True:
        query = listen()

        if query is None:
            continue

        # Check if the user asked for the time
        if 'time' in query:
            tell_time()

        # Check if the user asked to search Wikipedia
        elif 'wikipedia' in query:
            search_wikipedia(query)


        # If the user says 'quit' or 'exit', close the program
        elif 'quit' in query or 'exit' in query:
            close_program()

        else:
            speak("Sorry, I didn't understand that.")

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

Main loop:

- ***main():***Greets the user and enters an infinite loop to continuously listen for commands.
- Calls the ***listen()*** function to get user input.
- Checks the recognized query for specific keywords: If the query contains "time", it calls ***tell_time().***
- If the query contains "wikipedia", it calls search_wikipedia(query).
- If the query contains "quit" or "exit", it calls ***close_program().***
- If the query does not match any known commands, it informs the user that it didn't understand.

Execution Code :

Python file name : ***jarvis.py***

Command for executing code : ***python jarvis.py***

Conclusion: The Future of Human-AI Collaboration

As AI continues to advance, the collaboration between humans and AI will become increasingly vital. Jarvis represents a step toward a future where AI seamlessly complements human abilities, creating a more efficient and enriching world.