1. Prerequisites

Before starting, ensure you have the following:

- Windows or Linux system
- Python 3.11.9 installed
- A terminal or command prompt to execute commands
- Stable internet connection (to download required packages)
- Audio file (a `.wav` file with multiple speakers talking)

Checking Python Version and Installation

To check if Python is installed, run:

```
python --version
```

If Python is not installed or a version other than 3.11.x is installed, uninstall the existing version and install Python 3.11.9.

Windows:

1. Uninstall previous Python versions:
   - Open **Control Panel > Programs and Features**.
   - Find **Python**, right-click, and select **Uninstall**.
2. Download and install Python 3.11.9 from Python's official website.
3. Ensure Python is added to the system PATH during installation.

Linux:

1. Uninstall previous Python versions:

   ```
   sudo apt remove python3 && sudo apt autoremove
   ```

2. Install Python 3.11.9:

   ```
   sudo apt update && sudo apt install python3.11
   ```

Verify installation:

```
python --version
```

2. Setting Up the Virtual Environment

A virtual environment helps keep dependencies isolated.

Windows:

```
cd path\to\your\project-folder
python -m venv venv
venv\Scripts\activate
```

Linux:

```
cd /path/to/your/project-folder
python -m venv venv
source venv/bin/activate
```

Your terminal should now show `(venv)`, indicating the virtual environment is active.

3. Installing Required Dependencies

Run the following command inside the virtual environment:

```
pip install -r requirements.txt
```

If `requirements.txt` is missing, install packages manually:

```
pip install pydub pyannote.audio openai-whisper librosa matplotlib ffmpeg
python-dotenv
```

Installing FFmpeg

FFmpeg is required for audio processing.

Windows:

1. Download FFmpeg from https://ffmpeg.org/download.html
2. Add `ffmpeg` to the system PATH.

Linux:

```
sudo apt update && sudo apt install ffmpeg
```

Verify installation:

```
ffmpeg -version
```

4. Project Workflow - Running the Project

Now that everything is set up, run the project using:

```
python complete_file.py <audio_file>
```

Expected Output

The script will process the given audio file and generate a transcript as:

```
<audio_file_name>_transcript.txt
```

5. Understanding the Code

The project consists of a single script: `complete_file.py`.

How It Works:

1. **Loads environment variables**: Reads `HF_TOKEN` from a `.env` file.
2. **Validates input file**: Ensures the audio file exists.
3. **Converts audio**: Transforms the file into a 16kHz mono WAV format.
4. **Runs speaker diarization**: Identifies speakers using `pyannote.audio`.
5. **Transcribes speech**: Uses Whisper to transcribe each speaker's segment.
6. **Saves results**: Outputs a transcript to `<audio_file_name>_transcript.txt`.
7. **Cleans up temporary files**: Deletes temporary WAV files.

6. Setting Up the .env File

The Hugging Face API token should not be hardcoded. Instead, create a `.env` file.

1. Create a `.env` file in your project folder:

```
touch .env
```

2. Open `.env` and add your Hugging Face token:

```
HF_TOKEN=your_huggingface_token_here
```

Obtaining a Hugging Face Token

1. Go to Hugging Face.
2. Log in or create an account.
3. Navigate to **Settings > Access Tokens**.
4. Generate a new token and copy it.
5. Paste the token in the `.env` file.

After setting up the `.env` file, the script will automatically load it.

7. Final Output

After execution, the transcript will be saved as:

```
<audio_file_name>_transcript.txt
```

This file contains the transcribed speech with speaker labels.

8. Cleanup

The script automatically removes temporary files (`temp.wav` and `segment.wav`).