Install Libraries

```
# The exclamation mark (!) is used to run shell commands directly from Jupyter Notebook or Google Colab.
# This command installs the yt_dlp library, which is used for converting video to audio in Python.
!pip install yt_dlp
```

```
Collecting yt_dlp
  Downloading yt_dlp-2025.3.31-py3-none-any.whl.metadata (172 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 172.2/172.2 kB 3.6 MB/s eta 0:00:00
Downloading yt_dlp-2025.3.31-py3-none-any.whl (3.2 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 3.2/3.2 MB 47.3 MB/s eta 0:00:00
Installing collected packages: yt_dlp
Successfully installed yt_dlp-2025.3.31
```

```
# The exclamation mark (!) is used to run shell commands directly from Jupyter Notebook or Google Colab.
# This command installs the SpeechRecognition library, which is used for converting speech to text in Python.
!pip install speechrecognition
```

```
Collecting speechrecognition
  Downloading speechrecognition-3.14.2-py3-none-any.whl.metadata (30 kB)
Requirement already satisfied: typing-extensions in /usr/local/lib/python3.11/dist-packages (from speechrecognition) (4.13.0)
Downloading speechrecognition-3.14.2-py3-none-any.whl (32.9 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 32.9/32.9 MB 18.8 MB/s eta 0:00:00
Installing collected packages: speechrecognition
Successfully installed speechrecognition-3.14.2
```

```
# The exclamation mark (!) is used to run shell commands directly from Jupyter Notebook or Google Colab.
# This command installs the pydub library, which is used for audio processing tasks like converting, slicing, and merging audio file
!pip install pydub
```

```
Collecting pydub
  Downloading pydub-0.25.1-py2.py3-none-any.whl.metadata (1.4 kB)
Downloading pydub-0.25.1-py2.py3-none-any.whl (32 kB)
Installing collected packages: pydub
Successfully installed pydub-0.25.1
```

Generating Transcript from Audio

```
!pip install youtube-transcript-api
```

```
Collecting youtube-transcript-api
  Downloading youtube_transcript_api-1.0.3-py3-none-any.whl.metadata (23 kB)
Requirement already satisfied: defusedxml<0.8.0,>=0.7.1 in /usr/local/lib/python3.11/dist-packages (from youtube-transcript-api) (0
Requirement already satisfied: requests in /usr/local/lib/python3.11/dist-packages (from youtube-transcript-api) (2.32.3)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.11/dist-packages (from requests->youtube-transcrip
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/dist-packages (from requests->youtube-transcript-api) (3.1
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.11/dist-packages (from requests->youtube-transcript-api
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.11/dist-packages (from requests->youtube-transcript-api
Downloading youtube_transcript_api-1.0.3-py3-none-any.whl (2.2 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2.2/2.2 MB 26.6 MB/s eta 0:00:00
Installing collected packages: youtube-transcript-api
Successfully installed youtube-transcript-api-1.0.3
```

```
!pip install pytube
```

```
Collecting pytube
  Downloading pytube-15.0.0-py3-none-any.whl.metadata (5.0 kB)
Downloading pytube-15.0.0-py3-none-any.whl (57 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 57.6/57.6 kB 2.6 MB/s eta 0:00:00
Installing collected packages: pytube
Successfully installed pytube-15.0.0
```

```
import re
import urllib.parse
import requests
from youtube_transcript_api import YouTubeTranscriptApi
from pytube import YouTube
import speech_recognition as sr
from pydub import AudioSegment
import os
import yt_dlp
def extract_video_id(video_url):
    """
    Extracts the YouTube video ID from various URL formats.
    """
```

```python
    parsed_url = urllib.parse.urlparse(video_url)
    query_params = urllib.parse.parse_qs(parsed_url.query)

    if "v" in query_params:
        return query_params["v"][0]

    match = re.search(r"(youtu\.be/|youtube\.com/embed/|youtube\.com/shorts/)([\w-]+)", video_url)
    if match:
        return match.group(2)

    return None

def download_audio(video_url):
    """
    Downloads the audio using yt-dlp with cookies and returns the file path.
    """
    try:
        ydl_opts = {
            'format': 'bestaudio/best',
            'outtmpl': 'audio.%(ext)s',
            'cookiefile': '/content/cookies (2).txt',  # Use the exported cookies
            'postprocessors': [{
                'key': 'FFmpegExtractAudio',
                'preferredcodec': 'mp3',
                'preferredquality': '192',
            }],
        }
        with yt_dlp.YoutubeDL(ydl_opts) as ydl:
            info = ydl.extract_info(video_url, download=True)
            return "audio.mp3"
    except Exception as e:
        return f"Error downloading audio: {str(e)}"

def convert_audio_to_wav(audio_file):
    """
    Converts the downloaded MP3 audio to WAV format using pydub.
    """
    wav_file = "audio.wav"
    try:
        AudioSegment.from_mp3(audio_file).export(wav_file, format="wav")
        return wav_file
    except Exception as e:
        return f"Error converting to WAV: {str(e)}"

def transcribe_audio(audio_path, chunk_length=30):
    """
    Splits audio into smaller chunks and transcribes each chunk separately.
    Args:
        audio_path (str): Path to the audio file.
        chunk_length (int): Length of each chunk in seconds (default: 30).
    Returns:
        list: List of dictionaries containing transcribed text and timestamps.
    """
    recognizer = sr.Recognizer()
    audio = AudioSegment.from_wav(audio_path)
    total_duration = len(audio) / 1000  # Convert to seconds
    transcribed_segments = []

    print("Transcribing audio in chunks...")

    # Split and transcribe audio in chunks
    for start in range(0, int(total_duration), chunk_length):
        end = min(start + chunk_length, int(total_duration))
        chunk = audio[start * 1000:end * 1000]  # Extract chunk in milliseconds
        chunk.export("chunk.wav", format="wav")  # Save chunk temporarily

        with sr.AudioFile("chunk.wav") as source:
            try:
                audio_data = recognizer.record(source)
                text = recognizer.recognize_google(audio_data)
                transcribed_segments.append({
                    "start": start,
                    "end": end,
                    "text": text
                })
            except sr.UnknownValueError:
                transcribed_segments.append({
                    "start": start,
                    "end": end,
                    "text": "[Unintelligible]"
                })
            except sr.RequestError as e:
                return f"Error with the speech recognition service: {str(e)}"

    os.remove("chunk.wav")  # Clean up temporary chunk file
    return transcribed_segments

def get_transcript_unlisted(video_url):
    """
    Tries to fetch the transcript using youtube_transcript_api first,
    then falls back to downloading and transcribing audio if necessary.
    """
    video_id = extract_video_id(video_url)
    if not video_id:
        return "Invalid YouTube URL."

    # Try to fetch transcript using youtube_transcript_api
    try:
        transcript = YouTubeTranscriptApi.get_transcript(video_id)
        # Add 'end' time to each segment
        for segment in transcript:
            segment["end"] = segment["start"] + segment["duration"]
        return transcript  # Return transcript with timestamps
    except:
        print("Transcript not available via API, attempting audio transcription...")

    # Download and transcribe audio if no transcript is available
    audio_file = download_audio(video_url)
    if "Error" in audio_file:
        return audio_file

    wav_file = convert_audio_to_wav(audio_file)
    if "Error" in wav_file:
        return wav_file

    transcription = transcribe_audio(wav_file)

    # Cleanup temporary files
    os.remove(audio_file)
    os.remove(wav_file)

    return transcription

def save_transcript_to_file(transcript, filename="transcript.txt"):
    """
    Saves the transcript to a text file.
    Args:
        transcript (list or str): The transcript to save.
        filename (str): The name of the output file.
    """
    with open(filename, "w", encoding="utf-8") as file:
        if isinstance(transcript, list):
            for segment in transcript:
                file.write(f"{segment['start']} - {segment['end']}: {segment['text']}\n")
        else:
            file.write(transcript)
    print(f"Transcript saved to {filename}")

# Example usage
if __name__ == "__main__":
    video_url = input("Enter the YouTube video URL: ")
    transcript = get_transcript_unlisted(video_url)

    if isinstance(transcript, list):
        print("\nTranscript with Timestamps:")
        for segment in transcript:
            print(f"{segment['start']} - {segment['end']}: {segment['text']}")
    else:
        print("\nTranscript:\n", transcript)

    # Save transcript to a text file
    save_transcript_to_file(transcript, "transcript.txt")
```

```
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
<ipython-input-6-1e4221868400> in <cell line: 0>()
    152 # Example usage
    153 if __name__ == "__main__":
--> 154     video_url = input("Enter the YouTube video URL: ")
    155     transcript = get_transcript_unlisted(video_url)
    156

                                  1 frames
/usr/local/lib/python3.11/dist-packages/ipykernel/kernelbase.py in _input_request(self, prompt, ident, parent, password)
   1217             except KeyboardInterrupt:
   1218                 # re-raise KeyboardInterrupt, to truncate traceback
-> 1219                 raise KeyboardInterrupt("Interrupted by user") from None
   1220             except Exception:
   1221                 self.log.warning("Invalid Message:", exc_info=True)

KeyboardInterrupt: Interrupted by user
```

```python
import re
import urllib.parse
import requests
from youtube_transcript_api import YouTubeTranscriptApi
from pytube import YouTube
import speech_recognition as sr
from pydub import AudioSegment
import os
import yt_dlp

def extract_video_id(video_url):
    parsed_url = urllib.parse.urlparse(video_url)
    query_params = urllib.parse.parse_qs(parsed_url.query)

    if "v" in query_params:
        return query_params["v"][0]

    match = re.search(r"(youtu\.be/|youtube\.com/embed/|youtube\.com/shorts/)([\w-]+)", video_url)
    if match:
        return match.group(2)

    return None

def download_audio(video_url):
    try:
        ydl_opts = {
            'format': 'bestaudio/best',
            'outtmpl': 'audio.%(ext)s',
            'postprocessors': [{
                'key': 'FFmpegExtractAudio',
                'preferredcodec': 'mp3',
                'preferredquality': '192',
            }],
        }
        with yt_dlp.YoutubeDL(ydl_opts) as ydl:
            info = ydl.extract_info(video_url, download=True)
            return "audio.mp3"
    except Exception as e:
        return f"Error: {str(e)}"

def convert_audio_to_wav(audio_file):
    if "Error" in audio_file:
        return audio_file
    wav_file = "audio.wav"
    try:
        AudioSegment.from_mp3(audio_file).export(wav_file, format="wav")
        return wav_file
    except Exception as e:
        return f"Error converting to WAV: {str(e)}"

def transcribe_audio(audio_path, chunk_length=30):
    if "Error" in audio_path:
        return audio_path
    recognizer = sr.Recognizer()
    audio = AudioSegment.from_wav(audio_path)
    total_duration = len(audio) / 1000
    transcribed_segments = []
    formatted_transcript = {}

    for start in range(0, int(total_duration), chunk_length):
        end = min(start + chunk_length, int(total_duration))
        chunk = audio[start * 1000:end * 1000]
        chunk.export("chunk.wav", format="wav")

        with sr.AudioFile("chunk.wav") as source:
            try:
                audio_data = recognizer.record(source)
                text = recognizer.recognize_google(audio_data)
            except sr.UnknownValueError:
                text = "[Unintelligible]"
            except sr.RequestError as e:
                return f"Error with the speech recognition service: {str(e)}"

        formatted_transcript[f"{start}-{end}"] = text

    os.remove("chunk.wav")
    return formatted_transcript

def get_transcript_unlisted(video_url):
    video_id = extract_video_id(video_url)
    if not video_id:
        return "Invalid YouTube URL."

    try:
        transcript = YouTubeTranscriptApi.get_transcript(video_id)
        formatted_transcript = {}
        for segment in transcript:
            start = int(segment["start"] // 30) * 30
            end = start + 30
            if f"{start}-{end}" not in formatted_transcript:
                formatted_transcript[f"{start}-{end}"] = ""
            formatted_transcript[f"{start}-{end}"] += " " + segment["text"]
        return formatted_transcript
    except:
        print("Transcript not available via API, attempting audio transcription...")

    audio_file = download_audio(video_url)
    if "Error" in audio_file:
        return audio_file

    wav_file = convert_audio_to_wav(audio_file)
    if "Error" in wav_file:
        return wav_file

    transcription = transcribe_audio(wav_file)
    os.remove(audio_file)
    os.remove(wav_file)

    return transcription

def save_transcript_to_file(transcript, filename="transcript.txt"):
    if isinstance(transcript, str):
        print("Error:", transcript)
        return

    with open(filename, "w", encoding="utf-8") as file:
        for time_range, text in transcript.items():
            file.write(f"{time_range}: {text}\n")
    print(f"Transcript saved to {filename}")

if __name__ == "__main__":
    video_url = input("Enter the YouTube video URL: ")
    transcript = get_transcript_unlisted(video_url)

    if isinstance(transcript, dict):
        print("\nFormatted Transcript:")
        for time_range, text in transcript.items():
            print(f"{time_range}: {text}")
    else:
        print("\nError:\n", transcript)

    save_transcript_to_file(transcript, "transcript.txt")
```

```
Enter the YouTube video URL: https://youtu.be/UXoUwBW5nhs?si=KmBXMJwUfW2vIVlt

Formatted Transcript:
0-30:  most people think that being attractive is about looking good and while your looks do matter to some extent they are not the
30-60:  hit subscribe because I bring you videos like this every single week and let's learn about my top six most powerful ways to
60-90:  already there and investing in your appearance is literally in your control most of the times and it's not just because look
90-120:  elevated my confidence like nothing else this is something called the enclothed cognition effect in Psychology every time y
120-150:  automatic positive selft talk repeating over and over again in your head and what happens when you repeat something over a
150-180:  your life and when you have trust in yourself you automatically stand out and become memorable and thereby you become some
180-210:  person in addition to looking interesting how do you do that think back to when you were a kid and try to remember your fr
210-240:  we don't want to stand out we want to blend in we want to feel relatable but developing an interesting personality and att
```

```
240-270:  trained to identify applicants who have cool personalities and they do this by figuring out if an applicant is better thar
270-300:  one or two Niche things that truly interest you and then invest enough time enough energy into it to become better than th
300-330:  dishes coffee it's coffee for me honestly street photography or even knowing every IPL player's stats the key is to dive c
330-360:  responses automatically set you apart people will listen because you're presenting something different from what they norm
360-390:  ability to be interesting but then when you actually go to have a conversation with someone you find yourself stuck in a l
390-420:  making them feel like they just had the best conversation of their life let's break it down now imagine that you're at a w
420-450:  something they can contribute towards so how exactly do you get to this stage firstly remember that conversations are not
450-480:  Maggie tasted at 18,000 ft why is Mountain food so much better now they're thinking about their own travel experiences or
480-510:  so if somebody says they love Cricket don't just stop at oh cool so do I follow it up with if you could watch A Match live
510-540:  conversation awkward silences will be a part of you know your vocabulary for example you can have a story about the time y
540-570:  sleeve you don't have to use it every time but whenever there's an awkward silence whenever you don't know how to steer a
570-600:  reacting keeping the vibe alive think about it no one remembers the exact words somebody said but they always remember how
600-630:  practice this the better you will get at it soon you will go from that person was nice to wow I really enjoyed talking to
630-660:  world around you it's impossible to be engaging to be memorable to be attractive when your entire knowledge base is limite
660-690:  build the blocks of your personality and all you need to do is just keep doing more things that you find interesting you n
690-720:  and we learned how to make pizza and pasta from scratch and anytime somebody talks to us about Italy or anytime somebody t
720-750:  right when you expose yourself to new experiences you naturally become the kind of person who always has something to shar
750-780:  about celebrities it is about actively seeking out new experiences new ideas new cultures and you can do this simply by jo
780-810:  to talk about now one very crucial part of being interesting is being mysterious having that air of oh I want to know more
810-840:  you for granted it's human nature we value what feels rare and intentional not what is always there so like they say right
840-870:  that my time is important and so am I and when you prioritize yourself people see you as confident they see you as indeper
870-900:  quality over quantity you will notice that the world starts respecting this and valuing you a lot [Music] more now one fac
900-930:  memorable think about genzi influencers right so many of them are memorable you can probably list five 10 different influe
930-960:  earned every action you take how you show up in the world what you say how you say it how you treat people all of these th
960-990:  respectable the first is discipline somebody who sticks to their word someone who shows up for themselves regardless of hc
990-1020:  person who actually finishes their part without being chased down that's the person that everybody respects maybe everybo
1020-1050:  when you respect yourself other people naturally follow through on the same energy the second trait that I have seen in
1050-1080:  Blake Lively and Justin baldoni drama on the flip side when somebody is kind even if they're not the funniest or the sma
1080-1110:  like good celebrity PR honestly admittedly some people are inherently better at doing this but if you have the right knc
1110-1140:  to and you're going to spend 30 minutes a day for the next month learning more about that thing and doing or practicing
Transcript saved to transcript.txt
```

```
pip install -U yt-dlp

```

```
Requirement already satisfied: yt-dlp in /usr/local/lib/python3.11/dist-packages (2025.2.19)
```

```
Start coding or generate with AI.
```