```
#pip install whisper openai diffusers pydub
```

```
#from google.colab import drive
#drive.mount('/content/drive')
```

```
#pip install moviepy speechrecognition transformers pillow
```

```
import moviepy.editor as mp

def extract_audio(video_path, audio_path):
    video = mp.VideoFileClip(video_path)
    audio = video.audio
    audio.write_audiofile(audio_path)
    print(f"Audio extracted and saved to {audio_path}")

# Example usage
video_path = '/content/drive/MyDrive/What is Generative AI _ Introduction to Generative AI _ Generative AI Explained _ Simplilearn.m
audio_path = 'extracted_audio.wav'
extract_audio(video_path, audio_path)
```

```
MoviePy - Writing audio in extracted_audio.wav
                                                                                 MoviePy - Done.
Audio extracted and saved to extracted_audio.wav
```

```
import whisper

def audio_to_text(audio_path, text_path):
    # Load the Whisper model
    model = whisper.load_model("base")

    # Transcribe the audio
    result = model.transcribe(audio_path)
    text = result["text"]

    # Save the transcribed text to a file
    with open(text_path, 'w') as file:
        file.write(text)

    print(f"Text transcribed and saved to {text_path}")
    return text

# Example usage
audio_path = 'extracted_audio.wav'
text_path = 'transcribed_text.txt'
text = audio_to_text(audio_path, text_path)
```

```
100%|████████████████████████████████████████| 139M/139M [00:01<00:00, 122MiB/s]
WARNING:py.warnings:/usr/local/lib/python3.10/dist-packages/whisper/transcribe.py:115: UserWarning: FP16 is not supported on CPU
  warnings.warn("FP16 is not supported on CPU; using FP32 instead")

Text transcribed and saved to transcribed_text.txt
```

```
# Open and read the contents of the transcribed text file
with open(text_path, 'r') as file:
    transcribed_text = file.read()

# Print the transcribed text
print("Transcribed Text:")
print(transcribed_text)
```

```
Transcribed Text:
 The term generative AI has emers seemingly out of nowhere in recent months. With a notable surge in interest according to Googl
```

```
#pip install --upgrade whisper
```

```
#pip install openai-whisper
```

```
#pip install openai==0.28
```

```
#pip install transformers
```

```
from transformers import BartForConditionalGeneration, BartTokenizer

def summarize_text(text, summary_path):
    # Load pre-trained BART model and tokenizer
    model_name = "facebook/bart-large-cnn"
    tokenizer = BartTokenizer.from_pretrained(model_name)
    model = BartForConditionalGeneration.from_pretrained(model_name)

    # Encode the input text
    inputs = tokenizer.encode("summarize: " + text, return_tensors="pt", max_length=1024, truncation=True)

    # Generate summary
    summary_ids = model.generate(inputs, max_length=150, min_length=40, length_penalty=2.0, num_beams=4, early_stopping=True)
    summary = tokenizer.decode(summary_ids[0], skip_special_tokens=True)

    # Save the summary to a file
    with open(summary_path, 'w') as file:
        file.write(summary)

    print(f"Summarized text saved to {summary_path}")
    return summary

# Example usage
text_path = 'transcribed_text.txt'
summary_path = 'summarized_text.txt'

# Read the transcribed text from the file
with open(text_path, 'r') as file:
    transcribed_text = file.read()

# Summarize the transcribed text
summarized_text = summarize_text(transcribed_text, summary_path)

# Print the summarized text
print("Summarized Text:")
print(summarized_text)
```

```
WARNING:py.warnings:/usr/local/lib/python3.10/dist-packages/huggingface_hub/utils/_token.py:89: UserWarning:
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your settings tab (https://huggingface.co/settings/tokens), set it
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to access public models or datasets.
  warnings.warn(

vocab.json: 100%                          899k/899k [00:00<00:00, 1.32MB/s]
merges.txt: 100%                          456k/456k [00:00<00:00, 2.61MB/s]
tokenizer.json: 100%                      1.36M/1.36M [00:00<00:00, 7.52MB/s]
config.json: 100%                         1.58k/1.58k [00:00<00:00, 66.0kB/s]
model.safetensors: 100%                   1.63G/1.63G [00:25<00:00, 58.5MB/s]
generation_config.json: 100%              363/363 [00:00<00:00, 12.2kB/s]
Summarized text saved to summarized_text.txt
Summarized Text:
Generative AI is a form of artificial intelligence, possesses the capability of to generate a wide range of contact including te
```

```
#pip install transformers diffusers torch torchvision torchaudio
```

Show hidden output

```
from transformers import BartForConditionalGeneration, BartTokenizer
from diffusers import StableDiffusionPipeline
import torch
from PIL import Image
```

```
# Function to summarize text
def summarize_text(text, summary_path):
    model_name = "facebook/bart-large-cnn"
    tokenizer = BartTokenizer.from_pretrained(model_name)
    model = BartForConditionalGeneration.from_pretrained(model_name)

    inputs = tokenizer.encode("summarize: " + text, return_tensors="pt", max_length
    summary_ids = model.generate(inputs, max_length=150, min_length=40, length_pena
    summary = tokenizer.decode(summary_ids[0], skip_special_tokens=True)

    with open(summary_path, 'w') as file:
        file.write(summary)

    print(f"Summarized text saved to {summary_path}")
    return summary

# Function to create an image banner based on summarized text
def create_image_banner(summary_text, image_path):
    model_id = "CompVis/stable-diffusion-v1-4"  # or use another model if available
    device = "cuda" if torch.cuda.is_available() else "cpu"

    pipe = StableDiffusionPipeline.from_pretrained(model_id)
    pipe = pipe.to(device)

    with torch.no_grad():
        image = pipe(summary_text).images[0]

    image.save(image_path)
    print(f"Banner image created and saved to {image_path}")

# Example usage
text_path = 'transcribed_text.txt'
summary_path = 'summarized_text.txt'
image_path = 'banner_image.png'

# Read the transcribed text from the file
with open(text_path, 'r') as file:
    transcribed_text = file.read()

# Summarize the transcribed text
summarized_text = summarize_text(transcribed_text, summary_path)

# Print the summarized text
print("Summarized Text:")
print(summarized_text)

# Create an image banner based on the summarized text
create_image_banner(summarized_text, image_path)
```

```
Summarized text saved to summarized_text.txt
Summarized Text:
Generative AI is a form of artificial intelligence, possesses the capability of to generate a wide range of contact including te
```

```
model_index.json: 100%                                    541/541 [00:00<00:00, 33.3kB/s]
Fetching 16 files: 100%                                   16/16 [01:10<00:00,  4.84s/it]
model.safetensors: 100%                                   1.22G/1.22G [00:29<00:00, 62.0MB/s]
tokenizer/merges.txt: 100%                                525k/525k [00:00<00:00, 992kB/s]
scheduler/scheduler_config.json: 100%                     313/313 [00:00<00:00, 1.60kB/s]
```

```python
from IPython.display import Image as IPImage, display

# Display the generated image
display(IPImage(filename="banner_image.png"))
```