Scenario 1: You are designing an automated AI system that generates **visual news summaries**.
It should:

- Take a news topic (e.g., “AI in education”)
- Generate a short 100–200 word script
- Produce a relevant image
- Generate narration and create a 30-second news-style video

Question:

How would you adapt the text generation model and image prompts to produce more realistic, factual content instead of fantasy stories?
How can you evaluate the accuracy of AI-generated news content?

Scenario 2: “Cultural Story Revival Project”

Scenario:
You are working on a heritage preservation project that uses AI to retell **traditional folk tales**.
Your system should:

- Accept a title or theme (e.g., “Panchatantra wisdom”)
- Generate a story in that style
- Create matching artwork and narration
- Produce a short video that can be shared online

Question:

How would you fine-tune the story generation and image prompts to reflect **specific cultural contexts** or story styles?
How could multilingual text-to-speech improve accessibility?

Solution:

```
from transformers import pipeline

from diffusers import StableDiffusionPipeline

from gtts import gTTS
```

```
from moviepy.editor import ImageClip, AudioFileClip, concatenate_videoclips

import torch


generator = pipeline("text-generation", model="gpt2")

prompt = “Once upon a time madhura nagar lord krishna was living he was very nougty and clever”


story = generator(prompt, max_length=300,
num_return_sequences=1)[0]['generated_text']

print("Generated Story:\n", story)

# Generate an image based on the story

from IPython.display import display

pipe =
StableDiffusionPipeline.from_pretrained("runwayml/stable-diffusion-v1-5")

pipe = pipe.to("cuda" if torch.cuda.is_available() else "cpu")

image_prompt = story[:200]  # use first part of story as prompt

image = pipe(image_prompt).images[0]

image.save("generated_image.png")

display(image)

#Audio generation

from IPython.display import Audio

speech = gTTS(text=story[:400], lang='en')

speech.save("story_audio.mp3")

Audio("story_audio.mp3")

#Video generator
```

```
from moviepy.editor import *

from IPython.display import Video, display

# Create an image clip (duration matches audio)

audio_clip = AudioFileClip("story_audio.mp3")

image_clip =
ImageClip("generated_image.png").set_duration(audio_clip.duration)

image_clip = image_clip.set_audio(audio_clip)

final_clip = image_clip.fx(vfx.fadein, 1).fx(vfx.fadeout, 1)

final_clip = final_clip.resize(height=720)  # resize for standard video

output_path = "story_video.mp4"

final_clip.write_videofile(output_path, fps=24, codec="libx264",
audio_codec="aac")

print("Video generated successfully!")

display(Video(output_path, embed=True))
```