E-BOOK AUTOMAÇÃO EM PYTHON

Baixando e editando um vídeo

INÍCIO

O QUE É O PROGRAMA?

O programa desenvolvido é uma aplicação em Python que utiliza as bibliotecas MoviePy e PyTube para baixar vídeos do YouTube e adicionar legendas a eles. Esse programa é útil para aqueles que desejam assistir a vídeos do YouTube com legendas embutidas, seja por necessidade de acessibilidade ou para aprender um novo idioma.

O conceito principal explorado neste programa é a automação de tarefas de download de vídeo e adição de legendas. A automação de tarefas é uma prática comum na programação, onde rotinas repetitivas são automatizadas por meio de scripts ou programas. Neste caso, a automação é aplicada ao processo de baixar um vídeo do YouTube, procurar e baixar a legenda correspondente (se disponível) e finalmente combinar o vídeo e a legenda em um único arquivo de vídeo.

Além disso, o programa utiliza APIs fornecidas pelas bibliotecas PyTube e MoviePy para interagir com o YouTube e manipular arquivos de vídeo, respectivamente. Isso demonstra a importância de utilizar bibliotecas e APIs em vez de reinventar a roda, aproveitando o trabalho de desenvolvedores especializados para alcançar os objetivos de programação de forma eficiente e eficaz.

PARTE 1

IMPORTAÇÃO DE BIBLIOTECAS:

```
// put your import os
from pytube import YouTube
from moviepy.editor import VideoFileClip
from moviepy.editor import TextClip
code here
```

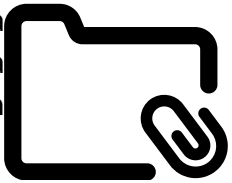

- os: é usado para interagir com o sistema operacional, manipulando caminhos de arquivos e pastas.
- pytube.YouTube: é a classe fornecida pelo PyTube para interagir com vídeos do YouTube.
- moviepy.editor.VideoFileClip: é usada para carregar vídeos e realizar operações de edição de vídeo.
- moviepy.editor.TextClip: é usada para criar um clip de texto que será usado como legenda.

PARTE 2

IFUNÇÃO DOWNLOAD_VIDEO(URL, OUTPUT_PATH)

```
def download_video(url, output_path):
    try:
        yt = YouTube(url)
        video = yt.streams.filter(file_extension='mp4', res='360p').first()
        video.download(output_path)
        return video.default_filename
    except Exception as e:
        print("Erro ao baixar o vídeo:", e)
        return None
```

Esta função baixa o vídeo do YouTube com a URL fornecida para o caminho de saída especificado. Ele tenta baixar o vídeo em formato MP4 com resolução de 360p.



PARTE 3

DOWNLOAD_SUBTITLE(YOUTUBE_URL, OUTPUT_PATH):

```
def download_subtitle(youtube_url, output_path):
    try:
        yt = YouTube(youtube_url)
        caption = yt.captions.get_by_language_code('pt')
        if caption:
            caption_str = caption.generate_srt_captions()
            with open(os.path.join(output_path, 'subtitle.srt'), 'w', encoding='utf-8') as file:
                file.write(caption_str)
            return True
        else:
            print("Não foi encontrada legenda em português para este vídeo.")
            return False
    except Exception as e:
        print("Erro ao baixar legenda:", e)
        return False
```

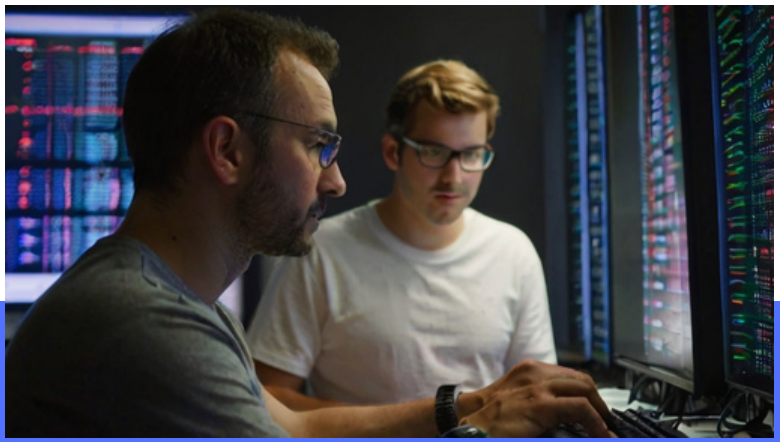

Esta função baixa a legenda em português (se disponível) para o vídeo do YouTube. Se a legenda for encontrada, ela é salva em um arquivo SRT.

PARTE 4

ADD_SUBTITLE(VIDEO_PATH, SUBTITLE_PATH, OUTPUT_PATH)

```
def add_subtitle(video_path, subtitle_path, output_path):
    try:
        video = VideoFileClip(video_path)
        subtitle = TextClip(subtitle_path, fontsize=24, color='white')
        .set_position(('center', 'bottom')).set_duration(video.duration)
        video = video.set_subclip(0, video.duration)
        video = video.set_audio(video.audio)
        video = video.set_subtitles(subtitle)
        video.write_videofile(os.path.join(output_path,
        'video_with_subtitle.mp4'), codec='libx264', audio_codec='aac')
        return True
    except Exception as e:
        print("Erro ao adicionar legenda ao vídeo:", e)
        return False
```


Esta função carrega o vídeo baixado e o arquivo de legenda, cria um clip de texto a partir da legenda e adiciona esse clip ao vídeo. O vídeo resultante com a legenda é salvo no caminho de saída especificado.



PARTE 5

FUNÇÃO MAIN()

```python
def main():
    url = input("Insira a URL do vídeo do YouTube: ")
    output_folder = input("Insira o caminho para a pasta de saída: ")

    video_filename = download_video(url, output_folder)
    if video_filename:
        subtitle_downloaded = download_subtitle(url, output_folder)
        if subtitle_downloaded:
            subtitle_path = os.path.join(output_folder, 'subtitle.srt')
            video_path = os.path.join(output_folder, video_filename)
            output_video_path = os.path.join(output_folder, 'video_with_subtitle.mp4')
            add_subtitle(video_path, subtitle_path, output_folder)
            print("Legenda adicionada com sucesso!")
            print("O vídeo com legenda está salvo em:", output_video_path)
            os.remove(video_path)
            os.remove(subtitle_path)
        else:
            print("Não foi possível adicionar legenda ao vídeo.")
            os.remove(os.path.join(output_folder, video_filename))
    else:
        print("Não foi possível baixar o vídeo.")
```


Esta função principal controla o fluxo do programa. Ela solicita a URL do vídeo do usuário e o caminho da pasta de saída. Em seguida, chama as funções para baixar o vídeo, baixar a legenda e adicionar a legenda ao vídeo. Se a operação for bem-sucedida, imprime uma mensagem com o caminho do vídeo resultante. Se ocorrer um erro, imprime uma mensagem correspondente.

PARTE 6

EXECUÇÃO DO PROGRAMA:

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


Esta linha garante que o programa seja executado somente se for executado como um script principal. Isso impede que o código seja executado se o arquivo for importado como um módulo em outro script.

CONCLUSÃO

Este projeto proporcionou uma oportunidade de explorar a automação de tarefas em Python, utilizando bibliotecas como PyTube e MoviePy para baixar vídeos do YouTube e adicionar legendas a eles. Ao longo do desenvolvimento, aprendemos a importância de utilizar APIs e bibliotecas de terceiros para facilitar o desenvolvimento de soluções eficazes e eficientes.

A automação de tarefas é uma habilidade valiosa em diversas áreas, incluindo produção de conteúdo, análise de dados, gerenciamento de sistemas e muito mais. Neste projeto específico, vimos como a automação pode ser aplicada para simplificar o processo de assistir a vídeos do YouTube com legendas incorporadas, melhorando a acessibilidade e facilitando a aprendizagem de novos idiomas.

Além disso, ao utilizar bibliotecas Python como PyTube e MoviePy, compreendemos a importância de aproveitar o vasto ecossistema de bibliotecas disponíveis na comunidade Python. Isso não apenas acelera o desenvolvimento de soluções, mas também nos permite concentrar nossos esforços na lógica específica do problema que estamos resolvendo.

Em resumo, este projeto exemplifica como a automação de tarefas pode ser aplicada de forma prática e útil, destacando a importância de utilizar bibliotecas e APIs para alcançar nossos objetivos de programação de maneira eficiente. Essas habilidades são essenciais para qualquer desenvolvedor que busque criar soluções eficazes e automatizar processos repetitivos em suas áreas de atuação.