import subprocess

import time

import os

import psutil

import pyautogui

import tkinter as tk

from tkinter import filedialog, Listbox, Button, Entry, Label, Checkbutton, messagebox

import openpyxl

import tkinter.simpledialog as simpledialog

import sys

from moviepy import VideoFileClip  # Assurez-vous que moviepy est installé

vlc\_path = r"C:\Program Files\VideoLAN\VLC\cvlc.exe"

if not os.path.exists(vlc\_path):

    print("VLC introuvable a l'emplacement indiquer :" , vlc\_path)

else :

    print("VLC trouvé !")

# Import de python-pptx pour compter le nombre de diapositives dans un PowerPoint

try:

    from pptx import Presentation

except ImportError:

    Presentation = None

    print(

        "La bibliothèque python-pptx n'est pas installée. Pour un décompte précis des slides PowerPoint, installez-la via 'pip install python-pptx'.")

def fermer\_applications():

    """Ferme les applications Word, Excel, PowerPoint, VLC et MPC-HC si elles sont ouvertes."""

    for proc in psutil.process\_iter(['pid', 'name']):

        try:

            if proc.info['name'] and proc.info['name'].lower() in ['winword.exe', 'excel.exe', 'powerpnt.exe', 'vlc.exe', 'mpc-hc.exe', 'wmplayer.exe']:

                proc.kill()

                proc.wait()

        except (psutil.NoSuchProcess, psutil.AccessDenied, psutil.ZombieProcess):

            pass

def forcer\_fermeture\_powerpoint():

    """

    Force la fermeture de PowerPoint en appelant taskkill /F /IM powerpnt.exe /T

    de manière répétée jusqu'à ce qu'aucun processus PowerPoint ne soit détecté.

    """

    print("Forçage de la fermeture de PowerPoint...")

    for i in range(10):

        os.system("taskkill /F /IM powerpnt.exe /T")

        time.sleep(1)

        # Vérifier s'il reste un processus PowerPoint

        ppt\_running = any(

            proc.info['name'] and proc.info['name'].lower() == 'powerpnt.exe'

            for proc in psutil.process\_iter(['name'])

        )

        if not ppt\_running:

            print("PowerPoint a été fermé.")

            break

    else:

        print("Impossible de fermer PowerPoint complètement.")

def ouvrir\_fichier\_pendant\_duree(chemin\_fichier, delai\_diapo=5, feuilles\_a\_defiler=None, duree\_affichage=30):

    """

    Ouvre le fichier et simule son affichage en fonction de son type.

    Pour PowerPoint (.pptx) :

      - Le nombre de diapositives est déterminé (via python-pptx).

      - La présentation démarre en mode plein écran.

      - Chaque diapo est affichée pendant 'delai\_diapo' secondes.

      - À la fin du défilement, la touche ESC est envoyée pour quitter, puis on force la fermeture de PowerPoint.

    Pour Excel (.xlsx) :

      - Le classeur est ouvert.

      - Si une sélection de feuilles (indices 0-based) a été configurée (feuilles\_a\_defiler non None), seules ces feuilles sont défilées.

      - Sinon, toutes les feuilles sont parcourues.

      - Chaque feuille est affichée pendant 'delai\_diapo' secondes.

      - Une fois toutes les feuilles affichées, Excel est fermé.

    Pour les vidéos (.mp4, .avi, .mov, .mkv) :

      - Le fichier est lu avec VLC en mode plein écran avec l'option --play-and-exit.

      - Le script attend la fin de la lecture (i.e. la fermeture de VLC) avant de passer au document suivant.

    Pour les autres types de fichiers :

      - Le fichier est ouvert et laissé affiché pendant 'duree\_affichage' secondes.

    """

    try:

        print(f"Ouverture du fichier : {chemin\_fichier}")

        ext = os.path.splitext(chemin\_fichier)[1].lower()

        if ext == '.pptx':

            subprocess.Popen('start "" "{}"'.format(chemin\_fichier), shell=True)

            time.sleep(5)

            if Presentation:

                prs = Presentation(chemin\_fichier)

                nb\_slides = len(prs.slides)

            else:

                nb\_slides = 10

            pyautogui.press('f5')

            time.sleep(2)

            for \_ in range(nb\_slides - 1):

                pyautogui.press('right')

                time.sleep(delai\_diapo)

            time.sleep(3)  # Délai unique pour la dernière diapositive

            pyautogui.press('esc')

            time.sleep(1)

            pyautogui.hotkey('alt', 'f4')

            time.sleep(1)

            forcer\_fermeture\_powerpoint()

        elif ext == '.xlsx':

            # Ouvrir le classeur Excel

            subprocess.Popen('start "" "{}"'.format(chemin\_fichier), shell=True)

            time.sleep(5)  # Attendre l'ouverture

            wb = openpyxl.load\_workbook(chemin\_fichier)

            total\_feuilles = len(wb.sheetnames)

            wb.close()

            if feuilles\_a\_defiler is not None:

                # Défilement sur la sélection de feuilles

                for \_ in range(total\_feuilles - 1):

                    pyautogui.hotkey('ctrl', 'pgup')

                    time.sleep(0.3)

                current\_sheet = 0

                for target in feuilles\_a\_defiler:

                    diff = target - current\_sheet

                    if diff > 0:

                        for \_ in range(diff):

                            pyautogui.hotkey('ctrl', 'pgdn')

                            time.sleep(0.3)

                    elif diff < 0:

                        for \_ in range(-diff):

                            pyautogui.hotkey('ctrl', 'pgup')

                            time.sleep(0.3)

                    current\_sheet = target

                    time.sleep(delai\_diapo)

            else:

                # Défilement automatique de toutes les feuilles

                for \_ in range(total\_feuilles):

                    pyautogui.hotkey('ctrl', 'pgup')

                    time.sleep(0.3)

                for \_ in range(total\_feuilles - 1):

                    pyautogui.hotkey('ctrl', 'pgdn')

                    time.sleep(delai\_diapo)

                time.sleep(delai\_diapo)

            # Fermer Excel correctement

            pyautogui.hotkey('alt', 'f4')

        elif ext == '.pdf':

            print(f"Affichage du PDF : {chemin\_fichier}")

            subprocess.Popen('start "" "{}"'.format(chemin\_fichier), shell=True)

            time.sleep(duree\_affichage)

            pyautogui.hotkey('alt', 'f4')

            time.sleep(1)  # Wait for closure

            # New handling for image files

        elif ext in ['.jpg', '.jpeg', '.png', '.gif', '.bmp']:

            print(f"Affichage de l'image : {chemin\_fichier}")

            subprocess.Popen('start "" "{}"'.format(chemin\_fichier), shell=True)

            time.sleep(duree\_affichage)

            pyautogui.hotkey('alt', 'f4')

            time.sleep(1)  # Wait for closure

        elif ext in ['.mp4', '.avi', '.mov', '.mkv']:

            if VideoFileClip is None:

                print("Erreur: MoviePy n'est pas installé. Installez-le via 'pip install moviepy'")

                return

            print(f"Lecture de la vidéo : {chemin\_fichier}")

            if  os.path.exists(chemin\_fichier):

                print("Vidéo trouvé a l'emplacement indiquer :", chemin\_fichier)

                try:

                    clip = VideoFileClip(chemin\_fichier)

                    clip\_duree = clip.duration

                    print(f"Durée de la vidéo : {clip\_duree} secondes")

                    duree\_affichage = clip\_duree

                    os.startfile(chemin\_fichier)

                    time.sleep(duree\_affichage+3)

                    fermer\_applications()

                except Exception as e:

                    print(f"Erreur lors de la lecture vidéo: {e}")

                finally:

                    if 'clip' in locals():

                        clip.close()

        else:

            # Pour les autres types de fichiers, utiliser la durée d'affichage fixe

            subprocess.Popen('start "" "{}"'.format(chemin\_fichier), shell=True)

            time.sleep(duree\_affichage)

            fermer\_applications()

    except FileNotFoundError:

        print(f"Erreur : Le fichier {chemin\_fichier} est introuvable.")

    except Exception as e:

        print(f"Erreur : {e}")

def ajouter\_fichiers(fichiers, listbox\_principal, listbox\_excel):

    """Ouvre une boîte de dialogue pour ajouter des fichiers et met à jour les listes."""

    nouveaux\_fichiers = filedialog.askopenfilenames(title="Ajouter des fichiers",

                                                    filetypes=[("Tous les fichiers", "\*.\*")])

    for fichier in nouveaux\_fichiers:

        if fichier not in fichiers:

            fichiers.append(fichier)

            listbox\_principal.insert(tk.END, fichier)

            if fichier.lower().endswith('.xlsx'):

                listbox\_excel.insert(tk.END, fichier)

def configurer\_feuilles(excel\_listbox, excel\_sheet\_config):

    """

    Pour le fichier Excel sélectionné dans la liste, ouvre une invite pour choisir

    les numéros de feuilles à défiler (en 1-based).

    La configuration est stockée dans le dictionnaire excel\_sheet\_config.

    """

    selection = excel\_listbox.curselection()

    if not selection:

        messagebox.showerror("Erreur", "Veuillez sélectionner un fichier Excel dans la liste.")

        return

    index = selection[0]

    file\_excel = excel\_listbox.get(index)

    try:

        wb = openpyxl.load\_workbook(file\_excel)

        sheet\_names = wb.sheetnames

        wb.close()

    except Exception as e:

        messagebox.showerror("Erreur", f"Erreur lors de l'ouverture du fichier Excel:\n{e}")

        return

    prompt = "Feuilles disponibles dans {}:\n".format(os.path.basename(file\_excel))

    prompt += "\n".join(f"{i + 1}: {name}" for i, name in enumerate(sheet\_names))

    prompt += "\n\nEntrez les numéros des feuilles à défiler (séparés par des virgules) :"

    response = simpledialog.askstring("Sélection des feuilles", prompt)

    if response:

        try:

            indices = [int(x.strip()) - 1 for x in response.split(",")]

            valid\_indices = [i for i in indices if 0 <= i < len(sheet\_names)]

            if valid\_indices:

                excel\_sheet\_config[file\_excel] = valid\_indices

            else:

                messagebox.showerror("Erreur", "Aucun indice valide n'a été entré.")

        except Exception as e:

            messagebox.showerror("Erreur", f"Erreur lors de la conversion des indices:\n{e}")

def configurer\_delai\_fichier(listbox\_principal, delai\_config):

    """

    Pour le fichier sélectionné dans la liste principale, ouvre une invite pour choisir

    le délai entre diapos (en secondes).

    La configuration est stockée dans le dictionnaire delai\_config sous forme de tuple (duree\_affichage, delai\_diapo).

    Pour PowerPoint et Excel, la valeur de 'duree\_affichage' est ignorée.

    """

    selection = listbox\_principal.curselection()

    if not selection:

        messagebox.showerror("Erreur", "Veuillez sélectionner un fichier pour configurer ses délais.")

        return

    index = selection[0]

    fichier = listbox\_principal.get(index)

    default\_delai = 5

    if fichier in delai\_config:

        \_, default\_delai = delai\_config[fichier]

    delai = simpledialog.askinteger("Délai entre diapos",

                                    f"Délai entre diapos (en secondes) pour {os.path.basename(fichier)} :",

                                    initialvalue=default\_delai)

    if delai is not None:

        delai\_config[fichier] = (30, delai)

def move\_up(fichiers, listbox):

    """Déplace vers le haut l'élément sélectionné dans la liste."""

    selection = listbox.curselection()

    if not selection:

        messagebox.showerror("Erreur", "Veuillez sélectionner un fichier à déplacer.")

        return

    index = selection[0]

    if index == 0:

        return

    fichiers[index], fichiers[index - 1] = fichiers[index - 1], fichiers[index]

    listbox.delete(0, tk.END)

    for file in fichiers:

        listbox.insert(tk.END, file)

    listbox.selection\_set(index - 1)

def move\_down(fichiers, listbox):

    """Déplace vers le bas l'élément sélectionné dans la liste."""

    selection = listbox.curselection()

    if not selection:

        messagebox.showerror("Erreur", "Veuillez sélectionner un fichier à déplacer.")

        return

    index = selection[0]

    if index == len(fichiers) - 1:

        return

    fichiers[index], fichiers[index + 1] = fichiers[index + 1], fichiers[index]

    listbox.delete(0, tk.END)

    for file in fichiers:

        listbox.insert(tk.END, file)

    listbox.selection\_set(index + 1)

def quit():

    sys.exit()

def main():

    root = tk.Tk()

    root.title("Sélection de fichiers")

    root.attributes('-fullscreen', True)

    root.bind('<Escape>', lambda e: root.attributes('-fullscreen', False))

    fichiers = []  # Liste de tous les fichiers sélectionnés

    excel\_sheet\_config = {}  # Dictionnaire : fichier Excel -> liste d'indices des feuilles

    delai\_config = {}  # Dictionnaire : fichier -> (duree\_affichage, delai\_diapo)

    Label(root, text="Fichiers sélectionnés :").pack(pady=5)

    listbox\_principal = Listbox(root, selectmode=tk.SINGLE, width=100, height=8)

    listbox\_principal.pack(padx=10, pady=5, fill=tk.BOTH, expand=False)

    bouton\_ajouter = Button(root, text="Ajouter des fichiers",

                            command=lambda: ajouter\_fichiers(fichiers, listbox\_principal, listbox\_excel))

    bouton\_ajouter.pack(pady=5)

    frame\_reorder = tk.Frame(root)

    frame\_reorder.pack(pady=5)

    bouton\_monter = Button(frame\_reorder, text="Monter", command=lambda: move\_up(fichiers, listbox\_principal))

    bouton\_monter.pack(side=tk.LEFT, padx=5)

    bouton\_descendre = Button(frame\_reorder, text="Descendre", command=lambda: move\_down(fichiers, listbox\_principal))

    bouton\_descendre.pack(side=tk.LEFT, padx=5)

    bouton\_config\_delai = Button(root, text="Configurer délais (fichier)",

                                 command=lambda: configurer\_delai\_fichier(listbox\_principal, delai\_config))

    bouton\_config\_delai.pack(pady=5)

    Label(root, text="Valeur globale par défaut :").pack(pady=5)

    Label(root, text="Délai entre diapos/feuilles (en secondes) :").pack(pady=5)

    delai\_entry = Entry(root)

    delai\_entry.pack(pady=5)

    delai\_entry.insert(0, "5")

    frame\_excel = tk.LabelFrame(root, text="Configuration des feuilles pour Excel", font=("Arial", 14))

    frame\_excel.pack(padx=10, pady=10, fill=tk.BOTH, expand=True)

    var\_config\_excel = tk.BooleanVar()

    check\_config = Checkbutton(frame\_excel,

                               text="Activer configuration manuelle des feuilles",

                               variable=var\_config\_excel,

                               font=("Arial", 16))

    check\_config.pack(pady=5)

    Label(frame\_excel, text="Fichiers Excel disponibles :", font=("Arial", 12)).pack(pady=5)

    listbox\_excel = Listbox(frame\_excel, selectmode=tk.SINGLE, width=80, height=4)

    listbox\_excel.pack(padx=10, pady=5, fill=tk.BOTH, expand=False)

    bouton\_configurer = Button(frame\_excel,

                               text="Configurer feuilles",

                               command=lambda: configurer\_feuilles(listbox\_excel, excel\_sheet\_config),

                               font=("Arial", 12))

    bouton\_configurer.pack(pady=5)

    bouton\_terminer = Button(root, text="Terminer", state=tk.NORMAL, command=root.quit , fg="Green" , bg= "Light Green", width=40)

    bouton\_terminer.pack(pady=5)

    button\_destroy = Button(root, text="quitter", command=quit, fg="red", bg='light grey', width=20,

                                 activebackground="red", font=("taille", 15))

    button\_destroy.place(relx=1.0, rely=1.0, anchor='se')

    root.mainloop()

    if not fichiers:

        print("Aucun fichier sélectionné.")

        return

    global\_duree = 30

    try:

        global\_delai = int(delai\_entry.get())

    except ValueError:

        print("Veuillez entrer une valeur numérique pour le délai entre diapos.")

        return

    fermer\_applications()

    print("Démarrage de la boucle de traitement. Appuyez sur Ctrl+C pour interrompre.")

    try:

        while True:

            for fichier in fichiers:

                # Récupérer les configurations individuelles

                file\_delai = global\_delai

                file\_duree = global\_duree

                if fichier in delai\_config:

                    file\_duree, file\_delai = delai\_config[fichier]

                if fichier.lower().endswith('.pptx'):

                    ouvrir\_fichier\_pendant\_duree(fichier, delai\_diapo=file\_delai)

                elif fichier.lower().endswith('.xlsx'):

                    feuilles = excel\_sheet\_config.get(fichier) if var\_config\_excel.get() else None

                    ouvrir\_fichier\_pendant\_duree(fichier, delai\_diapo=file\_delai, feuilles\_a\_defiler=feuilles)

                else:

                    # Pour les vidéos et autres types de fichiers, la fonction se charge de la temporisation

                    ouvrir\_fichier\_pendant\_duree(fichier, duree\_affichage=file\_duree, delai\_diapo=file\_delai)

                time.sleep(2)

    except KeyboardInterrupt:

        print("\nInterruption détectée. Fermeture de toutes les fenêtres.")

        fermer\_applications()

        sys.exit(0)

if \_\_name\_\_ == "\_\_main\_\_":

    main()