```python
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()
```