Organizador de archivos: Mueve archivos a carpetas basadas en su extensión (`.pdf`, `.jpg`, etc.).

file_organizer_v3.py

Python

```
import os
import shutil
import argparse
import sys

# Mapeo de extensiones a nombres de carpetas descriptivos
EXTENSION_CATEGORIES = {
    # Documentos
    '.pdf': 'DOCUMENTOS', '.doc': 'DOCUMENTOS', '.docx': 'DOCUMENTOS', '.txt': 'DOCUMENTOS',
    '.odt': 'DOCUMENTOS', '.rtf': 'DOCUMENTOS', '.xls': 'DOCUMENTOS', '.xlsx': 'DOCUMENTOS',
    '.ppt': 'DOCUMENTOS', '.pptx': 'DOCUMENTOS',

    # Imágenes
    '.jpg': 'IMAGENES', '.jpeg': 'IMAGENES', '.png': 'IMAGENES', '.gif': 'IMAGENES',
    '.bmp': 'IMAGENES', '.svg': 'IMAGENES', '.tiff': 'IMAGENES', '.webp': 'IMAGENES',

    # Videos
    '.mp4': 'VIDEOS', '.mov': 'VIDEOS', '.avi': 'VIDEOS', '.mkv': 'VIDEOS', '.wmv': 'VIDEOS',

    # Audio
    '.mp3': 'AUDIO', '.wav': 'AUDIO', '.flac': 'AUDIO', '.aac': 'AUDIO',

    # Comprimidos / Instaladores
    '.zip': 'COMPRIMIDOS', '.rar': 'COMPRIMIDOS', '.7z': 'COMPRIMIDOS', '.iso': 'INSTALADORES',
    '.exe': 'INSTALADORES', '.msi': 'INSTALADORES', '.dmg': 'INSTALADORES',

    # Código
    '.py': 'CODIGO', '.js': 'CODIGO', '.html': 'CODIGO', '.css': 'CODIGO', '.java': 'CODIGO',
}

def get_category(extension):
    """Obtiene la categoría de la carpeta a partir de la extensión."""
    # Convertir a minúsculas para coincidir con las claves del diccionario
    extension = extension.lower()
    return EXTENSION_CATEGORIES.get(extension, 'OTROS')

def handle_duplicate(target_path):
    """
    Añade un sufijo numérico al nombre del archivo si ya existe en la ruta de destino.
    Ej: archivo.pdf -> archivo (1).pdf
    """
    base, ext = os.path.splitext(target_path)
    counter = 1
    new_target_path = target_path

    # Mientras exista un archivo con el nombre actual (incluyendo el contador)
    while os.path.exists(new_target_path):
        # Creamos el nuevo nombre: nombre (contador).ext
        new_target_path = f"{base} ({counter}){ext}"
```

```
            counter += 1

    return new_target_path

def organize_files(directory, dry_run=True):
    """Organiza archivos en un directorio moviéndolos a subcarpetas basadas en su categoría."""
    if not os.path.isdir(directory):
        print(f"❌ Error: El directorio '{directory}' no existe.")
        return

    print(f"? Iniciando organización inteligente en: {directory}")
    print("-" * 50)

    movement_summary = {}

    try:
        for filename in os.listdir(directory):
            full_path = os.path.join(directory, filename)

            # 1. Ignorar elementos no deseados
            if os.path.isdir(full_path) or os.path.islink(full_path) or filename == os.path.basename(sys.argv[0]):
                continue

            # 2. Determinar la categoría y la carpeta de destino
            _, extension = os.path.splitext(filename)
            category = get_category(extension)
            target_folder = os.path.join(directory, category)

            # El destino inicial (sin revisar duplicados)
            initial_target_path = os.path.join(target_folder, filename)

            # Si el archivo ya está en su carpeta correcta (misma ruta), lo ignoramos
            if os.path.dirname(full_path) == target_folder:
                continue

            # 3. Manejar duplicados si el destino ya tiene un archivo con el mismo nombre
            final_target_path = initial_target_path
            if os.path.exists(initial_target_path):
                # Si existe, generamos un nuevo nombre con sufijo numérico
                final_target_path = handle_duplicate(initial_target_path)

            # 4. Crear la carpeta de destino y preparar para el movimiento
            if not os.path.exists(target_folder) and not dry_run:
                os.makedirs(target_folder)

            # Determinar el nombre final para la impresión (puede ser el original o el renombrado)
            final_filename = os.path.basename(final_target_path)

            print(f"Mover: '{filename}' -> /{category}/")
            if final_filename != filename:
                print(f"   Conflictos: renombrado a '{final_filename}'")

            if not dry_run:
                shutil.move(full_path, final_target_path)
```

```
                # 5. Actualizar el resumen
                movement_summary[category] = movement_summary.get(category, 0) + 1

        except Exception as e:
            print(f" Ocurrió un error inesperado: {e}")
            return

        print("-" * 50)

        # Imprimir resumen
        print("? Resumen de la Operación:")
        if movement_summary:
            for cat, count in movement_summary.items():
                print(f"  - {cat}: {count} archivos")
        else:
            print("  - No se encontraron archivos para organizar.")

        if dry_run:
            print("\n✅ ¡Simulación completada!")
            print("Para aplicar los cambios, ejecuta el script sin la opción '-d' o '--dry-run'.")
        else:
            print("✨ ¡Organización completada!")

def main():
    parser = argparse.ArgumentParser(
        description="Organizador masivo de archivos basado en extensión con categorías y manejo de
duplicados.",
        formatter_class=argparse.RawTextHelpFormatter
    )

    parser.add_argument(
        'directory',
        type=str,
        help="Ruta del directorio a organizar (ej: ~/Descargas)."
    )

    parser.add_argument(
        '-d', '--dry-run',
        action='store_true',
        help="Modo de prueba. Muestra qué archivos serían movidos y dónde, sin hacer cambios.
(¡Recomendado!)"
    )

    args = parser.parse_args()

    organize_files(
        directory=args.directory,
        dry_run=args.dry_run
    )

if __name__ == '__main__':
    main()
```

? Demostración de las Mejoras

1. Mapeo de Categorías

Si tienes los siguientes archivos en ./pruebas:

- pelicula.mp4
- musica.flac
- informe.docx

El script creará las carpetas: **VIDEOS**, **AUDIO** y **DOCUMENTOS**, en lugar de MP4, FLAC y DOCX.

2. Manejo de Duplicados

Imagina que ya tienes una carpeta ./Descargas/IMAGENES con el archivo logo.png. En la raíz de Descargas tienes otro archivo llamado logo.png que quieres mover.

Ejecución (Simulación):

Bash

```
python file_organizer_v3.py ./Descargas --dry-run
```

Salida (Relevante):

```
Mover: 'logo.png' -> /IMAGENES/
    Conflictos: renombrado a 'logo (1).png'
```

El nuevo archivo se moverá como logo (1).png, mientras que el archivo original en la subcarpeta IMAGENES permanece intacto, evitando que se sobrescriba.
