

Renombrador Masivo de Archivos en Python. Renombrador con Regex, Secuencia y Filtro

Este script utiliza el módulo `os` para interactuar con el sistema de archivos y el módulo `argparse` para manejar argumentos de línea de comandos, lo que lo hace flexible y profesional.

1. **Renombrado Secuencial** (`--sequence`): Añade un contador numérico.
2. **Filtrado por Extensión** (`--ext`): Procesa solo archivos con una extensión específica.
3. **Reemplazo con Expresiones Regulares** (`--regex`): Permite usar patrones complejos para la búsqueda.

mass_renamer_v2.py

Python

```
import os
```

```
import argparse
```

```
import sys
```

```
import re
```

```
def rename_files(directory, prefix, suffix, replace_old, replace_new, dry_run, sequence, extension, use_regex):  
    """
```

```
    Renombra masivamente archivos en un directorio basándose en los parámetros dados,  
    incluyendo secuenciación, filtrado y expresiones regulares.  
    """
```

```
    if not os.path.isdir(directory):  
        print(f"✗ Error: El directorio '{directory}' no existe.")  
        return
```

```
    print(f"? Analizando directorio: {directory}")  
    print("-" * 30)
```

```
    try:
```

```
        # 1. Aplicar filtro de extensión si está presente  
        all_files = os.listdir(directory)
```

```
        # Filtramos solo archivos (ignorando directorios) y, si se especificó, por extensión
```

```
        files_to_process = []
```

```
        for filename in all_files:
```

```
            full_path = os.path.join(directory, filename)
```

```
            if os.path.isdir(full_path):  
                continue
```

```
            base, ext = os.path.splitext(filename)
```

```
            # Si se especificó una extensión y no coincide, saltar  
            if extension and ext.lower() != extension.lower():  
                continue
```

```
            files_to_process.append(filename)
```

```
        # Ordenar archivos para asegurar una secuencia lógica (útil para el contador)
```

```
        files_to_process.sort()
```

```
        renamed_count = 0
```

```

# 2. Iterar y renombrar
for i, filename in enumerate(files_to_process):
    base_name, ext = os.path.splitext(filename)
    new_base_name = base_name

    # --- OPERACIONES DE RENOMBRADO ---

    # a. Reemplazo (Regex o simple)
    if replace_old and replace_new:
        if use_regex:
            try:
                # Reemplazo usando expresión regular
                new_base_name = re.sub(replace_old, replace_new, new_base_name)
            except re.error as e:
                print(f"✗ Error de Regex para patrón '{replace_old}': {e}. Saltando archivo.")
                continue
        else:
            # Reemplazo simple (string.replace)
            new_base_name = new_base_name.replace(replace_old, replace_new)

    # b. Secuenciación (añadir contador)
    if sequence:
        # Formato de tres dígitos, ej: 001, 002, etc.
        counter = str(i + 1).zfill(3)
        new_base_name = new_base_name + "_" + counter

    # c. Prefijo
    if prefix:
        new_base_name = prefix + new_base_name

    # d. Sufijo (añadido después del contador si existe)
    if suffix:
        new_base_name = new_base_name + suffix

    # --- EJECUTAR CAMBIO ---
    new_filename = new_base_name + ext

    if new_filename == filename:
        continue

    old_path = os.path.join(directory, filename)
    new_path = os.path.join(directory, new_filename)

    print(f"Renombrando: '{filename}' -> '{new_filename}'")
    renamed_count += 1

    if not dry_run:
        os.rename(old_path, new_path)

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

print("-" * 30)

```

```

if dry_run:
    print(f"👉 ¡Simulación completada! Se renombrarían {renamed_count} archivos.")
    print("Para aplicar los cambios, ejecuta el script sin la opción '-d' o '--dry-run'.")
else:
    print(f"👉 ¡Renombrado completado! Se modificaron {renamed_count} archivos en el directorio.")

def main():
    parser = argparse.ArgumentParser(
        description="Renombrador masivo de archivos con soporte para Regex, Secuencia y Filtros.",
        formatter_class=argparse.RawTextHelpFormatter
    )

    parser.add_argument('directory', type=str, help="Ruta del directorio donde se encuentran los archivos.")
    parser.add_argument('-p', '--prefix', type=str, default=None, help="Añade un prefijo al nombre del archivo.")
    parser.add_argument('-s', '--suffix', type=str, default=None, help="Añade un sufijo al nombre del archivo (antes
de la extensión).")
    parser.add_argument('--ext', type=str, default=None, help="Procesa SÓLO archivos con esta extensión (ej:
.jpg o .pdf).")
    parser.add_argument('--seq', '--sequence', action='store_true', help="Añade un contador secuencial (ej: _001,
_002) al nombre base.")

    group_replace = parser.add_argument_group('Opciones de Reemplazo (Simple/Regex)')
    group_replace.add_argument('--replace-old', type=str, default=None, help="Cadena o patrón Regex a
buscar.")
    group_replace.add_argument('--replace-new', type=str, default=None, help="Cadena o reemplazo Regex.")
    group_replace.add_argument('--regex', action='store_true', help="Usa Expresiones Regulares (Regex) para el
reemplazo.")

    parser.add_argument('-d', '--dry-run', action='store_true', help="Modo de prueba. Muestra qué archivos serían
renombrados sin hacer cambios.")

    args = parser.parse_args()

    # Validaciones
    if (args.replace_new and not args.replace_old) or (args.replace_old and not args.replace_new):
        print("❌ Error: Debes especificar tanto --replace-old como --replace-new juntos.")
        sys.exit(1)

    if not (args.prefix or args.suffix or args.replace_old or args.sequence):
        print("❌ Error: Debes especificar al menos un prefijo, sufijo, reemplazo o secuenciación.")
        sys.exit(1)

    rename_files(
        directory=args.directory,
        prefix=args.prefix,
        suffix=args.suffix,
        replace_old=args.replace_old,
        replace_new=args.replace_new,
        dry_run=args.dry_run,
        sequence=args.sequence,
        extension=args.ext,
        use_regex=args.regex
    )

```

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

? Ejemplos de Uso

1. Renombrado Secuencial y Filtro de Extensión

Objetivo: Renombrar solo los archivos .png en la carpeta ./capturas para que se llamen CAPTURA_001.png, CAPTURA_002.png, etc., añadiendo un prefijo y una secuencia.

Bash

Simulación (Dry Run):

```
python mass_renamer_v2.py ./capturas -p "CAPTURA_" --ext ".png" --sequence -d
```

Archivos Originales: imagen_a.png, fondo.png, documento.pdf

Archivos Renombrados (Simulación): CAPTURA_imagen_a_001.png, CAPTURA_fondo_002.png

2. Reemplazo con Expresiones Regulares (Regex)

Objetivo: Eliminar cualquier fecha con formato YYYY-MM-DD (ej: 2024-05-22) del nombre de los archivos. Usaremos un patrón Regex.

- **Regex a buscar (--replace-old):** \d{4}-\d{2}-\d{2}
- **Cadena de reemplazo (--replace-new):** (cadena vacía)

Bash

Simulación (Dry Run):

```
python mass_renamer_v2.py ./reportes --replace-old "\d{4}-\d{2}-\d{2}" --replace-new "" --regex -d
```

Archivos Originales: informe_2024-05-22_final.pdf, nota_2023-11-01.txt

Archivos Renombrados (Simulación): informe_final.pdf, nota.txt

3. Combinación de Operaciones

Objetivo: En la carpeta ./data, reemplazar todas las subcadenas vacio por completo, y a los archivos resultantes añadirles el sufijo _REVISADO.

Bash

Aplicar cambios:

```
python mass_renamer_v2.py ./data --replace-old "vacio" --replace-new "completo" -s "_REVISADO"
```

Archivos Originales: datos_vacio.csv, reporte_vacio.xls

Archivos Renombrados (Aplicado): datos_completo_REVISADO.csv,
reporte_completo_REVISADO.xls
