CLASE 16

Guia de Estudio: Modulos Utiles

pathlib | os | sys | datetime

Contenido:

- 1. pathlib - Manejo de rutas moderno
- 2. os y os.path - Referencia clasica
- 3. sys - Argumentos y sistema
- 4. datetime - Fechas y tiempos
- 5. Cheat Sheet - Referencia rapida
- 6. Proyecto ejemplo - Organizador de archivos
- 7. Ejercicios adicionales (8 ejercicios)
- 8. Soluciones completas

Estos modulos son parte de la libreria estandar de Python. No necesitas instalar nada adicional.

1. pathlib - Manejo de rutas moderno

pathlib es el modulo recomendado desde Python 3.4 para trabajar con rutas de archivos y directorios. Proporciona una interfaz orientada a objetos que es mas legible y segura que os.path.

1.1 Crear rutas (Path)

```
from pathlib import Path
# Crear rutas
archivo = Path("datos/reporte.txt")
carpeta = Path("datos/reportes")
# Ruta actual
actual = Path.cwd()
print(actual)  # C:\Users\Ana\proyecto
# Carpeta home
home = Path.home()
print(home)  # C:\Users\Ana
# Unir rutas con /
ruta = Path("proyecto") / "datos" / "archivo.txt"
print(ruta)  # proyecto/datos/archivo.txt
```

1.2 Partes de una ruta

```
archivo = Path("documentos/proyecto/reporte_final.txt")
archivo.name      # "reporte_final.txt"  (nombre completo)
archivo.stem      # "reporte_final"      (sin extension)
archivo.suffix    # ".txt"               (extension)
archivo.parent    # documentos/proyecto  (carpeta padre)
```

1.3 Verificar existencia y crear

```
ruta = Path("datos/reportes")
ruta.exists()     # True o False
ruta.is_file()    # True si es archivo
ruta.is_dir()     # True si es directorio
# Crear carpeta
ruta.mkdir()                              # Crea la carpeta
ruta.mkdir(exist_ok=True)                 # Sin error si ya existe
ruta.mkdir(parents=True, exist_ok=True)   # Crea intermedias
```

> Tip: Siempre usa parents=True y exist_ok=True juntos para evitar errores.

1.4 Listar y buscar archivos

```
carpeta = Path(".")
# Listar todo
for item in carpeta.iterdir():
    print(item)
# Buscar por patron (glob)
for f in carpeta.glob("*.txt"):       # Solo .txt
    print(f)
for f in carpeta.glob("**/*.py"):     # .py en subcarpetas
    print(f)
# Patrones utiles:
```

```
# *.txt       -> todos los .txt
# data_*.csv  -> data_01.csv, data_02.csv
# **/*.json   -> .json en TODAS las subcarpetas
```

1.5 Leer y escribir archivos

```
archivo = Path("notas.txt")
# Escribir (crea el archivo si no existe)
archivo.write_text("Hola mundo!")
# Leer
contenido = archivo.read_text()
print(contenido)  # Hola mundo!
# Borrar
archivo.unlink()                  # Borra el archivo
archivo.unlink(missing_ok=True)   # Sin error si no existe
```

Nota: write_text() SOBRESCRIBE el contenido. Para append, lee primero y luego escribe todo.

2. os y os.path - Referencia clasica

os.path es la forma clasica de manejar rutas en Python. Lo veras en codigo existente, tutoriales antiguos y Stack Overflow. No necesitas memorizarlo, pero si reconocerlo.

2.1 os.path - Funciones principales

```
import os
# Crear rutas
ruta = os.path.join("datos", "reportes", "2024")
# Verificar existencia
os.path.exists("archivo.txt")
os.path.isfile("archivo.txt")
os.path.isdir("carpeta")
# Obtener partes
os.path.basename("datos/reporte.txt")   # "reporte.txt"
os.path.dirname("datos/reporte.txt")    # "datos"
os.path.splitext("reporte.txt")         # ("reporte", ".txt")
```

2.2 os - Funciones utiles

```
import os
os.listdir(".")                    # Lista contenido (strings)
os.mkdir("carpeta")                # Crear carpeta
os.makedirs("a/b/c", exist_ok=True)  # Crear anidadas
os.rename("viejo.txt", "nuevo.txt")   # Renombrar/mover
os.remove("archivo.txt")          # Eliminar archivo
os.rmdir("carpeta_vacia")         # Eliminar carpeta vacia
os.getcwd()                        # Carpeta actual
```

2.3 Tabla comparativa: pathlib vs os

Operacion	pathlib (moderno)	os (clasico)
Crear ruta	`Path("a") / "b"`	`os.path.join("a", "b")`
Existe?	`Path(r).exists()`	`os.path.exists(r)`
Es archivo?	`Path(r).is_file()`	`os.path.isfile(r)`
Es carpeta?	`Path(r).is_dir()`	`os.path.isdir(r)`
Nombre	`Path(r).name`	`os.path.basename(r)`
Carpeta padre	`Path(r).parent`	`os.path.dirname(r)`
Extension	`Path(r).suffix`	`os.path.splitext(r)[1]`
Listar	`Path(r).iterdir()`	`os.listdir(r)`
Crear carpeta	`Path(r).mkdir(...)`	`os.makedirs(r, ...)`
Eliminar archivo	`Path(r).unlink()`	`os.remove(r)`
Ruta actual	`Path.cwd()`	`os.getcwd()`
Leer archivo	`Path(r).read_text()`	`open(r).read()`

Regla: Usa pathlib para codigo nuevo. Entiende os.path para leer codigo existente.

3. sys - Argumentos y sistema

3.1 sys.argv - Argumentos de linea de comandos

sys.argv es una lista que contiene los argumentos pasados al script desde la terminal. El primer elemento [0] es el nombre del script.

```
# Terminal: python mi_script.py archivo.txt 100
# sys.argv = ["mi_script.py", "archivo.txt", "100"]
#              [0]             [1]           [2]
import sys
print(f"Script: {sys.argv[0]}")
print(f"Argumentos: {sys.argv[1:]}")
# Ejemplo practico
if len(sys.argv) < 2:
    print("Uso: python script.py <archivo>")
    sys.exit(1)   # Salir con codigo de error
archivo = sys.argv[1]
print(f"Procesando: {archivo}")
```

Importante: sys.argv siempre contiene STRINGS. Usa int() o float() para convertir.

3.2 sys.exit() - Terminar el script

```
import sys
sys.exit()     # Termina con exito (codigo 0)
sys.exit(0)    # Termina con exito (explicito)
sys.exit(1)    # Termina con error
sys.exit("Mensaje de error")  # Imprime mensaje y sale
```

3.3 Ejemplo completo con sys.argv

```
# contar_lineas.py
import sys
from pathlib import Path
if len(sys.argv) < 2:
    print("Uso: python contar_lineas.py <archivo>")
    sys.exit(1)
archivo = Path(sys.argv[1])
if not archivo.exists():
    print(f"Error: '{archivo}' no existe")
    sys.exit(1)
contenido = archivo.read_text()
lineas = contenido.strip().split("\n")
print(f"Archivo: {archivo.name}")
print(f"Lineas: {len(lineas)}")
print(f"Caracteres: {len(contenido)}")
```

4. datetime - Fechas y tiempos

4.1 Tipos principales

```
from datetime import datetime, date, time, timedelta
# datetime  -> fecha Y hora completa
# date      -> solo fecha (anio, mes, dia)
# time      -> solo hora (hora, min, seg)
# timedelta -> diferencia entre fechas
```

4.2 Crear fechas

```
from datetime import datetime, date
# Fecha y hora actual
ahora = datetime.now()
print(ahora)  # 2025-01-30 14:35:22.123456
# Solo fecha actual
hoy = date.today()
print(hoy)  # 2025-01-30
# Crear fecha especifica
navidad = date(2025, 12, 25)
# Crear fecha Y hora especifica
evento = datetime(2025, 6, 15, 18, 30, 0)
# Acceder a componentes
print(ahora.year)    # 2025
print(ahora.month)   # 1
print(ahora.day)     # 30
print(ahora.hour)    # 14
print(ahora.minute)  # 35
```

4.3 strftime - Formatear fecha a texto

strftime = 'string format time'. Convierte un objeto fecha a texto formateado.

```
ahora = datetime.now()
ahora.strftime("%d/%m/%Y")         # "30/01/2025"
ahora.strftime("%Y-%m-%d")         # "2025-01-30"
ahora.strftime("%H:%M:%S")         # "14:35:22"
ahora.strftime("%I:%M %p")         # "02:35 PM"
ahora.strftime("%d/%m/%Y %H:%M")   # "30/01/2025 14:35"
# Para nombres de archivos
ahora.strftime("backup_%Y%m%d.zip")  # "backup_20250130.zip"
```

Codigos de formato:

Codigo	Significado	Ejemplo
`%d`	Dia (01-31)	30
`%m`	Mes (01-12)	01
`%Y`	Anio 4 digitos	2025
`%y`	Anio 2 digitos	25
`%H`	Hora 24h (00-23)	14
`%I`	Hora 12h (01-12)	02

%M	Minutos (00-59)	35
%S	Segundos (00-59)	22
%p	AM/PM	PM
%B	Nombre del mes	January
%A	Nombre del dia	Thursday

4.4 strptime - Parsear texto a fecha

strptime = 'string parse time'. Convierte texto a objeto fecha.

```
# Parsear texto a fecha
texto = "30/01/2025"
fecha = datetime.strptime(texto, "%d/%m/%Y")
print(fecha)  # 2025-01-30 00:00:00
# Otro formato
texto = "2025-01-30 14:35"
fecha = datetime.strptime(texto, "%Y-%m-%d %H:%M")
# CUIDADO: El formato debe coincidir exactamente
# "30-01-2025" con "%d/%m/%Y" -> ERROR!
# "30-01-2025" con "%d-%m-%Y" -> OK!
```

Nemotecnia: strFtime = Format (fecha->texto). strPtime = Parse (texto->fecha).

4.5 timedelta - Calcular con fechas

```
from datetime import datetime, timedelta
ahora = datetime.now()
# Sumar tiempo
manana = ahora + timedelta(days=1)
en_una_semana = ahora + timedelta(weeks=1)
en_2_horas = ahora + timedelta(hours=2)
# Restar tiempo
ayer = ahora - timedelta(days=1)
# Diferencia entre fechas
fecha1 = datetime(2025, 1, 1)
fecha2 = datetime(2025, 12, 31)
diferencia = fecha2 - fecha1
print(diferencia.days)  # 364
# Parametros de timedelta:
# days, seconds, minutes, hours, weeks
# NO existe months ni years
```

5. Cheat Sheet - Referencia rapida

pathlib - Metodos principales

Metodo/Propiedad	Descripcion	Ejemplo
`Path(str)`	Crear objeto Path	`Path("datos/a.txt")`
`Path.cwd()`	Carpeta actual	`Path.cwd()`
`Path.home()`	Carpeta home	`Path.home()`
`p / str`	Unir rutas	`Path("a") / "b.txt"`
`.name`	Nombre completo	`"reporte.txt"`
`.stem`	Nombre sin extension	`"reporte"`
`.suffix`	Extension	`".txt"`
`.parent`	Carpeta padre	`"datos"`
`.exists()`	Existe?	`True / False`
`.is_file()`	Es archivo?	`True / False`
`.is_dir()`	Es carpeta?	`True / False`
`.mkdir(...)`	Crear carpeta	`p.mkdir(parents=True, exist_ok=True)`
`.iterdir()`	Listar contenido	`for item in p.iterdir():`
`.glob(pat)`	Buscar patron	`p.glob("**/*.py")`
`.read_text()`	Leer archivo	`contenido = p.read_text()`
`.write_text(s)`	Escribir archivo	`p.write_text("hola")`
`.unlink()`	Borrar archivo	`p.unlink(missing_ok=True)`
`.stat().st_size`	Tamano en bytes	`p.stat().st_size`
`.stat().st_mtime`	Fecha modificacion	`timestamp float`
`.absolute()`	Ruta absoluta	`p.absolute()`

datetime - Referencia rapida

Operacion	Codigo
Fecha/hora actual	`datetime.now()`
Solo fecha	`date.today()`
Crear fecha	`date(2025, 12, 25)`
Crear fecha+hora	`datetime(2025, 6, 15, 18, 30)`
Formatear	`fecha.strftime("%d/%m/%Y")`
Parsear	`datetime.strptime("30/01/25", "%d/%m/%y")`
Sumar dias	`fecha + timedelta(days=7)`
Restar dias	`fecha - timedelta(weeks=1)`
Diferencia	`(fecha2 - fecha1).days`
Timestamp a fecha	`datetime.fromtimestamp(ts)`

6. Proyecto ejemplo - Organizador de archivos

Este script organiza archivos de una carpeta por extension, creando subcarpetas automaticamente. Combina pathlib, shutil y sys.argv.

```
from pathlib import Path
import shutil
import sys
CATEGORIAS = {
    "imagenes": [".jpg", ".jpeg", ".png", ".gif", ".bmp", ".webp"],
    "documentos": [".pdf", ".doc", ".docx", ".xlsx", ".xls", ".txt", ".pptx"],
    "codigo": [".py", ".js", ".html", ".css", ".java", ".cpp", ".h"],
    "videos": [".mp4", ".avi", ".mkv", ".mov"],
    "audio": [".mp3", ".wav", ".flac"],
    "comprimidos": [".zip", ".rar", ".7z"],
}
def obtener_categoria(extension):
    extension = extension.lower()
    for categoria, extensiones in CATEGORIAS.items():
        if extension in extensiones:
            return categoria
    return "otros"
def organizar_carpeta(ruta_str):
    carpeta = Path(ruta_str)
    if not carpeta.exists():
        print(f"Error: '{ruta_str}' no existe")
        return
    movidos = 0
    for archivo in carpeta.iterdir():
        if not archivo.is_file():
            continue
        categoria = obtener_categoria(archivo.suffix)
        destino_dir = carpeta / categoria
        destino_dir.mkdir(exist_ok=True)
        destino = destino_dir / archivo.name
        if destino.exists():
            print(f"  Ya existe: {destino}")
            continue
        shutil.move(str(archivo), str(destino))
        print(f"  {archivo.name} -> {categoria}/")
        movidos += 1
    print(f"\nArchivos movidos: {movidos}")
if __name__ == "__main__":
    carpeta = sys.argv[1] if len(sys.argv) > 1 else "."
    print(f"Organizando: {Path(carpeta).absolute()}\n")
    organizar_carpeta(carpeta)
```

7. Ejercicios adicionales

NIVEL PRINCIPIANTE

Ejercicio P1: Crear estructura de proyecto

Crea un script que reciba un nombre de proyecto y cree la siguiente estructura de carpetas:

```
mi_proyecto/
    src/
    tests/
    docs/
    data/
    README.txt  (con el nombre del proyecto)
```

Requisitos: Usar pathlib. Que no de error si ya existen las carpetas. Mostrar mensaje de exito.

Ejercicio P2: Listar solo imagenes

Crea una funcion listar_imagenes(carpeta) que reciba una ruta y liste SOLO archivos de imagen (.jpg, .png, .gif, .bmp, .webp). Debe mostrar nombre, tamano en KB y extension.

Ejercicio P3: Calcular edad

Crea una funcion calcular_edad(fecha_nacimiento_str) que reciba una fecha en formato 'dd/mm/yyyy' y retorne un diccionario con anios, meses aproximados y dias totales vividos.

NIVEL INTERMEDIO

Ejercicio I1: Renombrador masivo

Crea un script que agregue la fecha actual al inicio del nombre de cada archivo en una carpeta. Ejemplo: foto.jpg -> 2025-01-30_foto.jpg. Debe funcionar con sys.argv para recibir la carpeta.

Ejercicio I2: Buscador de duplicados por nombre

Crea una funcion que busque archivos con el mismo nombre en una carpeta y todas sus subcarpetas (usando glob recursivo). Muestra los duplicados agrupados por nombre.

Ejercicio I3: Generador de reportes con fecha

Crea un script que genere un archivo de texto con estadisticas de una carpeta (total archivos, tamano total, archivos por extension). El nombre del archivo debe incluir la fecha: reporte_20250130_1435.txt

NIVEL AVANZADO

Ejercicio A1: Sincronizador de carpetas

Crea un script que compare dos carpetas y muestre: archivos que estan solo en la carpeta A, archivos solo en la carpeta B, y archivos que existen en ambas pero tienen diferente tamano. Usa pathlib y recibe ambas rutas por sys.argv.

Ejercicio A2: Limpiador de archivos viejos

Crea un script que reciba una carpeta y un numero de dias. Debe encontrar todos los archivos que NO han sido modificados en los ultimos N dias y moverlos a una carpeta 'archivo_YYYYMMDD'. Debe mostrar un resumen con cuantos archivos movio y el espacio liberado.

8. Soluciones completas

Solucion P1: Crear estructura de proyecto

```
from pathlib import Path
def crear_proyecto(nombre):
    base = Path(nombre)
    carpetas = ["src", "tests", "docs", "data"]
    for carpeta in carpetas:
        (base / carpeta).mkdir(parents=True, exist_ok=True)
        print(f"  Creada: {base / carpeta}")
    readme = base / "README.txt"
    readme.write_text(f"Proyecto: {nombre}\nCreado con Python pathlib\n")
    print(f"  Creado: {readme}")
    print(f"\nProyecto '{nombre}' creado exitosamente!")
crear_proyecto("mi_proyecto")
```

Solucion P2: Listar solo imagenes

```
from pathlib import Path
EXTENSIONES_IMG = {".jpg", ".jpeg", ".png", ".gif", ".bmp", ".webp"}
def listar_imagenes(carpeta_str="."):
    carpeta = Path(carpeta_str)
    if not carpeta.is_dir():
        print(f"Error: '{carpeta_str}' no es una carpeta")
        return
    imagenes = [f for f in carpeta.iterdir()
                if f.is_file() and f.suffix.lower() in EXTENSIONES_IMG]
    if not imagenes:
        print("No se encontraron imagenes")
        return
    print(f"Imagenes en {carpeta.absolute()}:")
    total = 0
    for img in sorted(imagenes):
        kb = img.stat().st_size / 1024
        total += kb
        print(f"  {img.name:30} {kb:8.1f} KB  ({img.suffix})")
    print(f"\nTotal: {len(imagenes)} imagenes, {total:.1f} KB")
listar_imagenes(".")
```

Solucion P3: Calcular edad

```
from datetime import datetime
def calcular_edad(fecha_str):
    nacimiento = datetime.strptime(fecha_str, "%d/%m/%Y")
    hoy = datetime.now()
    diferencia = hoy - nacimiento
    anios = diferencia.days // 365
    meses = (diferencia.days % 365) // 30
    dias_total = diferencia.days
    return {
        "anios": anios,
        "meses_aprox": meses,
        "dias_totales": dias_total
```

```
    }
resultado = calcular_edad("15/06/1995")
print(f"Edad: {resultado['anios']} anios")
print(f"Meses adicionales: ~{resultado['meses_aprox']}")
print(f"Dias totales vividos: {resultado['dias_totales']:,}")
```

Solucion I1: Renombrador masivo

```
from pathlib import Path
from datetime import date
import sys
def renombrar_con_fecha(carpeta_str="."):
    carpeta = Path(carpeta_str)
    hoy = date.today().strftime("%Y-%m-%d")
    renombrados = 0
    for archivo in carpeta.iterdir():
        if not archivo.is_file():
            continue
        if archivo.name.startswith(hoy):
            continue  # Ya tiene la fecha
        nuevo_nombre = f"{hoy}_{archivo.name}"
        nuevo_path = archivo.parent / nuevo_nombre
        if nuevo_path.exists():
            print(f"  Ya existe: {nuevo_nombre}")
            continue
        archivo.rename(nuevo_path)
        print(f"  {archivo.name} -> {nuevo_nombre}")
        renombrados += 1
    print(f"\nArchivos renombrados: {renombrados}")
carpeta = sys.argv[1] if len(sys.argv) > 1 else "."
renombrar_con_fecha(carpeta)
```

Solucion I2: Buscador de duplicados

```
from pathlib import Path
def buscar_duplicados(carpeta_str="."):
    carpeta = Path(carpeta_str)
    archivos_por_nombre = {}
    for archivo in carpeta.glob("**/*"):
        if archivo.is_file():
            nombre = archivo.name
            if nombre not in archivos_por_nombre:
                archivos_por_nombre[nombre] = []
            archivos_por_nombre[nombre].append(archivo)
    duplicados = {n: rutas for n, rutas in archivos_por_nombre.items()
                  if len(rutas) > 1}
    if not duplicados:
        print("No se encontraron duplicados")
        return
    print(f"Duplicados encontrados: {len(duplicados)} nombres\n")
    for nombre, rutas in sorted(duplicados.items()):
        print(f"  '{nombre}' ({len(rutas)} copias):")
        for ruta in rutas:
            kb = ruta.stat().st_size / 1024
            print(f"    {ruta} ({kb:.1f} KB)")
buscar_duplicados(".")
```

Solucion I3: Generador de reportes

```
from pathlib import Path
from datetime import datetime
def generar_reporte(carpeta_str="."):
```

```
    carpeta = Path(carpeta_str)
    ahora = datetime.now()
    nombre_reporte = ahora.strftime("reporte_%Y%m%d_%H%M.txt")
    extensiones = {}
    total_archivos = 0
    total_tamano = 0
    for item in carpeta.iterdir():
        if item.is_file() and not item.name.startswith("reporte_"):
            total_archivos += 1
            tamano = item.stat().st_size
            total_tamano += tamano
            ext = item.suffix if item.suffix else "(sin ext)"
            extensiones[ext] = extensiones.get(ext, 0) + 1
    lineas = [
        f"Reporte de: {carpeta.absolute()}",
        f"Fecha: {ahora.strftime('%d/%m/%Y %H:%M')}",
        f"{'=' * 40}",
        f"Total archivos: {total_archivos}",
        f"Tamano total: {total_tamano/1024:.1f} KB",
        f"\nPor extension:",
    ]
    for ext, cnt in sorted(extensiones.items()):
        lineas.append(f"  {ext}: {cnt} archivo(s)")
    reporte = Path(carpeta_str) / nombre_reporte
    reporte.write_text("\n".join(lineas))
    print(f"Reporte guardado: {reporte}")
generar_reporte(".")
```

Solucion A1: Sincronizador de carpetas

```
from pathlib import Path
import sys
def sincronizar(carpeta_a_str, carpeta_b_str):
    a = Path(carpeta_a_str)
    b = Path(carpeta_b_str)
    archivos_a = {f.name: f for f in a.iterdir() if f.is_file()}
    archivos_b = {f.name: f for f in b.iterdir() if f.is_file()}
    solo_a = set(archivos_a) - set(archivos_b)
    solo_b = set(archivos_b) - set(archivos_a)
    comunes = set(archivos_a) & set(archivos_b)
    print(f"Solo en {a.name}/: ({len(solo_a)})")
    for nombre in sorted(solo_a):
        print(f"  {nombre}")
    print(f"\nSolo en {b.name}/: ({len(solo_b)})")
    for nombre in sorted(solo_b):
        print(f"  {nombre}")
    diferentes = []
    for nombre in comunes:
        ta = archivos_a[nombre].stat().st_size
        tb = archivos_b[nombre].stat().st_size
        if ta != tb:
            diferentes.append((nombre, ta, tb))
    print(f"\nDiferente tamano: ({len(diferentes)})")
    for nombre, ta, tb in sorted(diferentes):
        print(f"  {nombre}: {ta/1024:.1f}KB vs {tb/1024:.1f}KB")
if len(sys.argv) < 3:
    print("Uso: python sync.py <carpeta_a> <carpeta_b>")
    sys.exit(1)
sincronizar(sys.argv[1], sys.argv[2])
```

Solucion A2: Limpiador de archivos viejos

```
from pathlib import Path
from datetime import datetime, timedelta
import shutil
import sys
def limpiar_viejos(carpeta_str, dias):
    carpeta = Path(carpeta_str)
    limite = datetime.now() - timedelta(days=dias)
    hoy_str = datetime.now().strftime("%Y%m%d")
    archivo_dir = carpeta / f"archivo_{hoy_str}"
    movidos = 0
    espacio = 0
    for item in carpeta.iterdir():
        if not item.is_file():
            continue
        fecha_mod = datetime.fromtimestamp(item.stat().st_mtime)
        if fecha_mod < limite:
            archivo_dir.mkdir(exist_ok=True)
            destino = archivo_dir / item.name
            if not destino.exists():
                tamano = item.stat().st_size
                shutil.move(str(item), str(destino))
                movidos += 1
                espacio += tamano
```

```
                dias_viejo = (datetime.now() - fecha_mod).days
                print(f"  {item.name} ({dias_viejo} dias)")
    print(f"\nResumen:")
    print(f"  Archivos movidos: {movidos}")
    print(f"  Espacio: {espacio/1024:.1f} KB")
    if movidos > 0:
        print(f"  Destino: {archivo_dir}")
if len(sys.argv) < 3:
    print("Uso: python limpiar.py <carpeta> <dias>")
    sys.exit(1)
limpiar_viejos(sys.argv[1], int(sys.argv[2]))
```