Clase 011 — pathlib, lectura y escritura de archivos

Parte: 0 — Prerrequisitos · Fuente: Python Tutorial cap. 10 · pathlib docs · Effective Python (Slatkin) ítem 38. · Duración estimada: 60 min.

Clase 011 — pathlib, lectura y escritura de archivos

> *Parte: 0 — Prerrequisitos · Fuente: Python Tutorial cap. 10 · pathlib docs · Effective Python (Slatkin) ítem 38. · Duración estimada: 60 min.*

Objetivo

Que el alumno deje de usar os.path.join + strings y adopte pathlib.Path — API orientada a objetos, multiplataforma (Windows/Unix), con métodos legibles para todas las operaciones de filesystem que hace todo el tiempo en DS (leer CSV, listar archivos, crear carpetas).

Resultados de aprendizaje

Al finalizar la clase, el alumno podrá:

1. Construir paths con Path(...) / 'subdir' / 'file.csv' (operador /).
2. Leer/escribir archivos texto y binarios con métodos de Path (read_text, write_bytes).
3. Listar y filtrar archivos con iterdir, glob, rglob (recursivo).
4. Crear/eliminar estructuras de directorios sin pelear con os.makedirs(exist_ok=True).
5. Manejar rutas relativas vs absolutas y entender __file__.

Temas

#	Tema	Por qué importa
1	Path vs strings	Objetos con métodos > concatenación manual
2	Operador / para componer	Legible y multiplataforma.
3	read_text / write_text / read_bytes	One-liners para operaciones simples.
4	glob y rglob	Patrones tipo shell: .csv, /.py.
5	mkdir(parents=True, exist_ok=True)	Crea árbol completo idempotente.
6	Path(__file__).parent y resolve()	Localizar recursos relativos al script.

Versión profundizada — 2026

El tema moderno que vivía como complemento dentro de esta clase ahora tiene clase propia dedicada:

- Clase 032b — Parquet, Arrow, PyArrow, DuckDB

Definiciones y características

Path

: Objeto que representa una ruta de filesystem orientada a objetos (pathlib.Path). Sobreescribe / para componer rutas, multiplataforma (Windows usa \, Unix /, transparente). Tiene métodos para casi todo: leer, escribir, listar, mover, borrar.

Ruta absoluta vs relativa

: Absoluta: empieza desde la raíz (C:\dev\proyecto\data.csv o /home/user/data.csv). Relativa: parte del cwd (data.csv). Las rutas relativas dependen de dónde se ejecuta — fuente de bugs.

Path.cwd() vs Path(__file__).parent

: cwd() es el directorio donde se ejecutó el script (cambia según el invocante). __file__ es la ruta al archivo Python actual; .parent su carpeta. Usa __file__ para recursos que viven junto al script.

glob vs rglob

: Patrones tipo shell. glob('.csv') busca en el directorio actual (1 nivel). rglob('.csv') busca recursivo (todo el árbol). ** significa 'cualquier número de directorios'.

read_text / write_text

: One-liners para texto: Path('x.txt').write_text(contenido, encoding='utf-8'). Para binario: read_bytes / write_bytes. Para JSON/CSV usa librerías especializadas.

Dataset / recursos

Carpeta temporal creada en el notebook con archivos sintéticos. Sin descarga.

Ejercicios

1. Construye una ruta multiplataforma. Dado Path.home() / 'datos' / '2026' / 'enero.csv', imprime cómo se ve en Windows vs Unix.

2. Lista CSVs. En una carpeta con archivos mixtos (.csv, .txt, .py), lista solo los .csv ordenados por tamaño.

3. Búsqueda recursiva. En un árbol de carpetas, encuentra todos los .py que contengan la palabra TODO en su contenido.

4. Escribe + lee. Genera 3 archivos txt con write_text, léelos con read_text, concaténalos en uno solo.

5. Ruta del script. Escribe un script que cargue un dataset que vive al lado del script (no del cwd), usando Path(__file__).parent / 'data.csv'.

Homework verificable

Script inventario.py que recibe un directorio y produce un reporte CSV con: nombre, tamaño_bytes, extensión, última_modificación para cada archivo recursivamente, usando solo pathlib (no os).

Criterio de aceptación: El script corre tanto en Windows como en Linux/macOS sin cambios.

Errores comunes

Síntoma / mensaje	Causa y cómo arreglar
FileNotFoundError aunque el archivo existe	Estás usando ruta relativa y el cwd no es
PermissionError al escribir	Carpeta read-only, OneDrive sincronizando,
mkdir() falla si el directorio ya existe	Default es exist_ok=False. Fix: p.mkdir(pa
Mezclo os.path y pathlib	Algunos funciones esperan strings (open(),
glob('*.CSV') no encuentra archivo.csv	Case-sensitive en Linux, insensible en Win
ImportError: Missing optional dependency '	pandas delega Parquet/Arrow a pyarrow (o f

Preguntas frecuentes

¿pathlib o os.path?

pathlib para código nuevo. os.path es la API funcional vieja (strings + funciones); pathlib es OO y mucho más legible. Solo usa os.path para compatibilidad con código viejo.

¿Cómo evito el típico 'C:/Users/...' vs '/home/...' cross-platform?

No hardcodees rutas absolutas. Usa Path.home(), Path(__file__).parent, tempfile.gettempdir(). Y siempre Path + /, nunca strings concatenados.

¿Cómo leo un CSV grande con pathlib?

Pathlib es para paths, no parsing. Combina: pd.read_csv(Path('data') / 'big.csv'). El Path se convierte a string automáticamente.

¿Path('a') / 'b/c' o Path('a') / 'b' / 'c'?

Ambas funcionan: Path parsea separadores. Pero la primera es menos explícita; prefiere la segunda.

¿shutil o pathlib para mover/copiar?

shutil para operaciones recursivas (copytree, rmtree, move) — pathlib solo cubre operaciones simples. Combinable: shutil.copy(src_path, dst_path) acepta Path directo.

¿Cuándo dejo de usar CSV?

Regla práctica: CSV solo para intercambio con no-técnicos (alguien lo va a abrir en Excel o mandarlo por mail). Para cualquier dataset >100 MB, o uno que vas a releer más de una vez en tu pipeline, pasalo a Parquet: ahorrás disco, ganás velocidad de lectura y no perdés tipos. Si lo único que querés es cachear un DataFrame entre dos scripts de la misma máquina, usá Feather.

Referencias

- pathlib docs
- PEP 428 — Object-oriented filesystem paths
- Slatkin, Effective Python 2e — ítem 38 Use Pathlib instead of os.path.
- Apache Arrow / Parquet

Siguiente clase

Clase 012 — Logging

Apéndice: notebook (primer bloque)

Primera celda ejecutable del notebook de la clase.

```
import tempfile
from pathlib import Path
import time

# Carpeta de trabajo temporal
base = Path(tempfile.mkdtemp(prefix='lab011_'))
```

```
print(f'trabajo en: {base}')
```

Archivos complementarios

- notebook.ipynb