Generalitat de Catalunya
Departament d'Educació
Institut Thos i Codina

Automatització de còpies de seguretat amb Python

Domini Públic (CC0)

Índex

Què aprendràs 1

1. El repte: centralitzar dades disperses **1**

2. Les eines: os i `shutil` **1**

3. L'estratègia lògica **3**

4. El cervell: la funció `copiaDades` **5**
4.1. Directoris: `shutil.copytree` 5
4.2. Fitxers: `shutil.copy2` 5
4.3. Diferències clau: `copytree` vs. `copy2` 5

5. Configuració: orígens i destí **6**

6. Màgia de rutes: construint el destí **6**

7. Execució: el bucle principal **7**

8. Robustesa: gestió d'errors **7**

9. Resum: el codi complet (versió bàsica) **7**

10. Passar a producció: versió millorada **8**
Ús 11

11. Programar-lo: execució automàtica **11**
Amb `cron` 11
Amb un temporitzador de `systemd` 11

12. Errors habituals i limitacions **12**

13. Referència **13**

Índex de figures **14**

Nivell: introducció a Python (variables, llistes, funcions, bucles i `try/except`).

Què aprendràs

En acabar, sabràs:

- Distingir si una ruta és un fitxer o una carpeta i actuar en conseqüència.
- Copiar fitxers (amb metadades) i arbres de directoris amb `shutil`.
- Construir rutes de destí que conservin l'estructura original.
- Gestionar errors perquè un fitxer que falta no aturi tota la còpia.
- Convertir el script en una eina de línia d'ordres, amb registre d'esdeveniments, i programar-lo.

1. El repte: centralitzar dades disperses

Tenim fitxers i carpetes escampats pel sistema (`/etc/`, fitxers `.txt`, exportacions `.sql`...) i els volem replicar en **un únic destí segur**.

Objectiu: un script que copiï fitxers i carpetes de diversos orígens a un únic destí.

Requisits:

- Acceptar tant fitxers solts com carpetes senceres.
- Mantenir l'estructura de directoris original.
- Gestionar errors (fitxers inexistents) sense aturar-se.

Exemple del resultat esperat. Amb `desti = '/copies/'`:

Origen	Es copia a
`/etc/hosts`	`/copies/etc/hosts`
`/home/usuari/`	`/copies/home/usuari/`
`/srv/dades/dump.sql`	`/copies/srv/dades/dump.sql`

2. Les eines: `os` i `shutil`

Mòdul	Rol	Mètodes clau
`os`	Interactuar amb el sistema de fitxers	`os.path.isdir`: comprova si és una carpeta`os.makedirs`: crea jerarquies de carpetes`os.path.join`: construeix rutes de manera segura
`shutil`	Operacions d'alt nivell (còpia)	`shutil.copytree`: copia arbres de directoris`shutil.copy2`: copia fitxers preservant metadades

```
import os
import shutil
```

NOTA

Per què `os.path.join` i no concatenar textos? Perquè posa el separador correcte (`/` o `\`) segons el sistema i evita errors de barres duplicades o que falten.

3. L'estratègia lògica

El script ha de «pensar» diferent segons si tracta una carpeta o un fitxer.

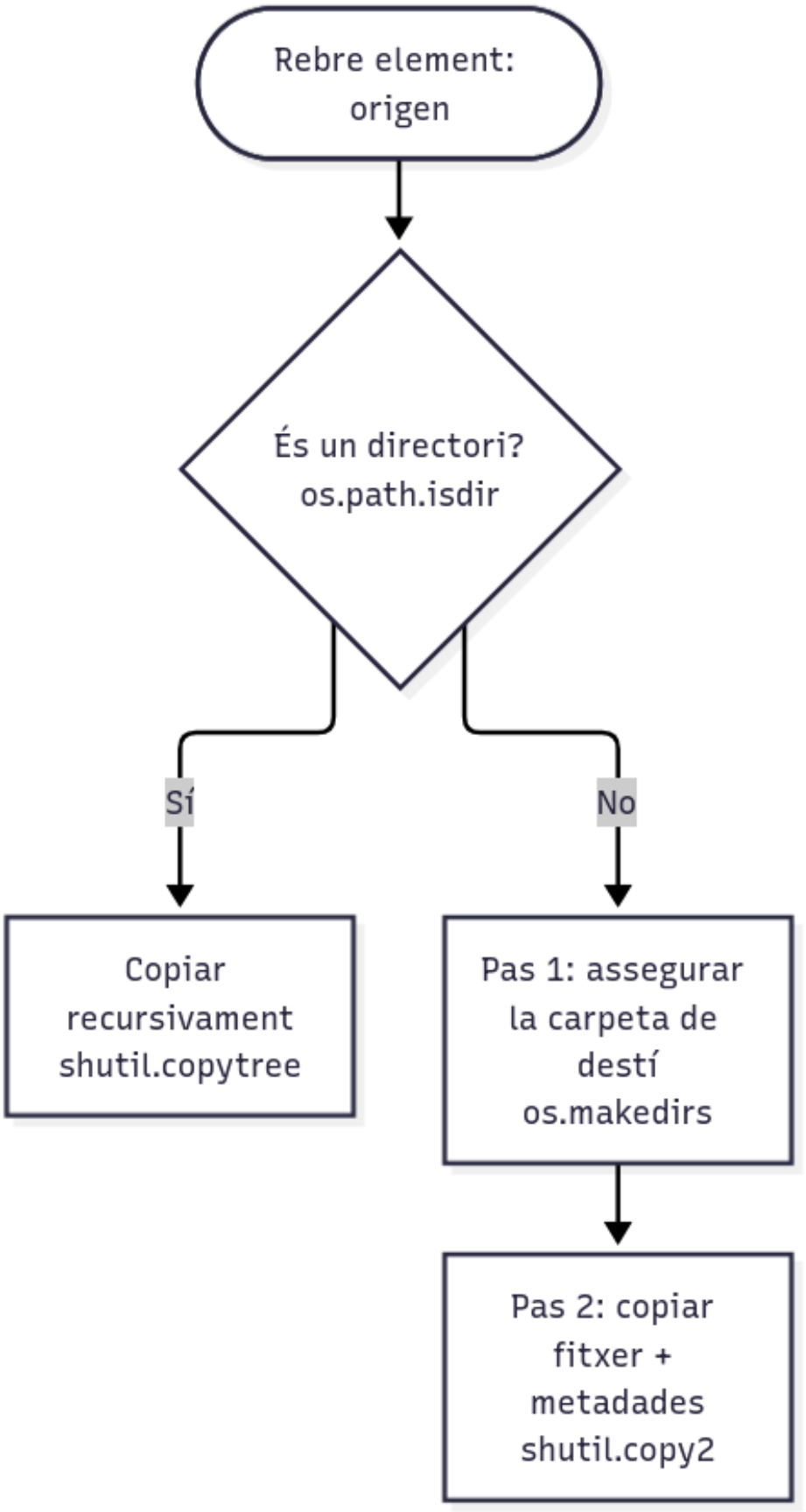

Figura 1: Estratègia lògica

4. El cervell: la funció `copiaDades`

```
def copiaDades(origen, desti):
```

- `origen`: l'element a copiar.
- `desti`: on s'ha de guardar.

La funció encapsula «la feina bruta»: executa la còpia adequada segons el tipus de dada.

4.1. Directoris: `shutil.copytree`

```
if os.path.isdir(origen):
    shutil.copytree(origen, desti, dirs_exist_ok=True)
```

- **Recursiu:** copia la carpeta i *tot* el seu contingut.
- **`dirs_exist_ok=True`:** permet fusionar dades encara que la carpeta de destí ja existeixi. Sense aquest paràmetre, la segona execució del script fallaria amb `FileExistsError`. Requereix Python 3.8 o superior.

4.2. Fitxers: `shutil.copy2`

```
else:
    os.makedirs(os.path.dirname(desti), exist_ok=True)
    shutil.copy2(origen, desti)
```

1. **Preparació:** `os.makedirs` crea la jerarquia de carpetes que falti. `copy2` **no** les crea, i sense aquest pas fallaria amb `FileNotFoundError`. `exist_ok=True` evita l'error si ja existeix.
2. **Execució:** `shutil.copy2` copia el contingut i conserva la data de modificació i els permisos.

4.3. Diferències clau: `copytree` vs. `copy2`

Característica	`copytree`	`copy2`
Objectiu	Carpetes senceres	Fitxers individuals
Comportament	Recursiu (copia fills i nets)	Unitari (un sol element)
Crea carpetes pare?	Sí (les del destí)	No: cal `makedirs` abans
Metadades	Per defecte usa `copy2` internament	Les preserva (mtime, permisos)

NOTA

`copy`, `copyfile` i `copy2`. `shutil.copyfile` només copia el contingut; `shutil.copy` hi afegeix els permisos; `shutil.copy2` hi afegeix també les dates. Per a còpies de seguretat, `copy2` és la millor opció. Cap d'aquestes funcions conserva el propietari (`chown`) del fitxer.

5. Configuració: orígens i destí

```
origens = ['/home/usuari/', '/etc/', 'document.txt']
desti = '/copies/'
```

- **`origens`:** una llista de *strings*. Pot barrejar rutes absolutes i relatives.
- **`desti`:** la ruta base on s'emmagatzemaran totes les còpies.

AVÍS

Permisos. Cal tenir lectura als orígens i escriptura al destí. Per copiar `/etc/` completa, segurament caldrà executar-lo amb `sudo`; fitxers com `/etc/shadow` només els pot llegir `root`.

6. Màgia de rutes: construint el destí

Volem que `/etc/hosts` acabi a `/copies/etc/hosts`:

```
/copies/  +  etc/hosts  =  /copies/etc/hosts
```

Una ruta absoluta comença amb `/`, i `os.path.join` **descarta tot el que hi ha abans d'una ruta absoluta**:

```
>>> os.path.join('/copies/', '/etc/hosts')
'/etc/hosts'                        # ← el destí s'ha perdut!
```

Per això cal treure la primera barra abans d'unir:

```
nomDesti = os.path.join(desti, org[1:])
```

AVÍS

Compte amb les rutes relatives. `org[1:]` talla el primer caràcter *sempre*: amb `'/etc/hosts'` funciona, però amb `'document.txt'` dona `'ocument.txt'`. Una alternativa que funciona en tots dos casos és:

```
nomDesti = os.path.join(desti, org.lstrip('/'))
```

Tampoc serveix a Windows (`C:\Dades`). A la secció 10 hi ha una versió amb `pathlib` que funciona a tots dos sistemes.

7. Execució: el bucle principal

```
for org in origens:
    nomDesti = os.path.join(desti, org.lstrip('/'))
    copiaDades(org, nomDesti)
```

- Itera sobre la llista `origens`.
- Calcula la nova ruta de destí dinàmicament.
- Crida `copiaDades` per a cada element.

8. Robustesa: gestió d'errors

```
try:
    # codi de còpia...
except FileNotFoundError:
    print('No existeix', origen)
except PermissionError:
    print('Sense permisos per a', origen)
```

Evita que el script s'aturi si un fitxer de la llista no existeix: informa de l'error i continua amb el següent element.

NOTA

Per què no un `except Exception:` genèric? Amagaria errors reals (disc ple, error de programació…). És millor capturar només els errors que sabem tractar.

9. Resum: el codi complet (versió bàsica)

```
import os
import shutil


def copiaDades(origen, desti):
    try:
        if os.path.isdir(origen):
            shutil.copytree(origen, desti, dirs_exist_ok=True)
        else:
            os.makedirs(os.path.dirname(desti), exist_ok=True)
            shutil.copy2(origen, desti)
    except FileNotFoundError:
        print('No existeix', origen)
    except PermissionError:
        print('Sense permisos per a', origen)
```

```
def main():
    origens = ['/home/usuari/', '/etc/', 'document.txt']
    desti = '/copies/'

    for org in origens:
        nomDesti = os.path.join(desti, org.lstrip('/'))
        copiaDades(org, nomDesti)


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

- **Modular:** separa lògica i configuració.
- **Intel·ligent:** distingeix fitxers i carpetes.
- **Segur:** gestiona errors i preserva metadades.
- **Llest per executar:** només cal definir orígens i destí.

10. Passar a producció: versió millorada

La versió bàsica serveix per aprendre, però té limitacions pràctiques:

Limitació	Conseqüència	Solució
Orígens i destí escrits dins del codi	Cal editar el script cada vegada	Arguments de línia d'ordres (`argparse`)
Només `print`	No queda registre; no es pot revisar un cop executat per `cron`	Mòdul `logging`
Sempre retorna èxit	Un `cron` no s'assabenta dels errors	Codi de sortida (`sys.exit(1)`)
`org.lstrip('/')`	No funciona amb Windows	`pathlib` + `relative_to(anchor)`
Cada execució sobreescriu l'anterior	No es pot recuperar una versió anterior	Subcarpeta amb data
Si el destí és dins d'un origen	Còpia recursiva infinita	Comprovació prèvia
`resolve()` segueix els enllaços	`/bin` es copia a `/copies/usr/bin`	`os.path.abspath`
Enllaços per defecte (`symlinks=False`)	Un enllaç trencat dona error i un enllaç cap al pare pot provocar recursió	`symlinks=True`

```
#!/usr/bin/env python3
"""Còpia de seguretat de fitxers i carpetes a un únic destí."""
import argparse
import logging
import os
import shutil
import sys
from datetime import datetime
from pathlib import Path

log = logging.getLogger("copies")
```

```
def ruta_absoluta(p: Path) -> Path:
    """Ruta absoluta normalitzada, SENSE seguir enllaços simbòlics."""
    return Path(os.path.abspath(p.expanduser()))


def ruta_desti(origen: Path, desti: Path) -> Path:
    """/etc/hosts + /copies -> /copies/etc/hosts (també amb rutes
     ↪  relatives)."""
    absolut = ruta_absoluta(origen)
    return desti / absolut.relative_to(absolut.anchor)


def esborra_enllacos(carpeta: Path) -> None:
    """Esborra els enllaços simbòlics d'una còpia anterior (copytree
     ↪  no els
    sobreescriu). os.walk no segueix enllaços, així que només toca el
     ↪  destí."""
    for dirpath, dirnames, filenames in os.walk(carpeta):
        for nom in dirnames + filenames:
            p = Path(dirpath, nom)
            if p.is_symlink():
                p.unlink()


def copia_dades(origen: Path, desti: Path) -> bool:
    """Copia un fitxer o una carpeta. Retorna True si tot ha anat
     ↪  bé."""
    try:
        if origen.is_symlink() and not origen.is_dir():
            # enllaç a fitxer (o trencat): es copia l'enllaç, no el
             ↪  contingut
            desti.parent.mkdir(parents=True, exist_ok=True)
            if desti.is_symlink() or desti.exists():
                desti.unlink()
            shutil.copy2(origen, desti, follow_symlinks=False)
        elif origen.is_dir():
            esborra_enllacos(desti)
            shutil.copytree(origen, desti, symlinks=True,
             ↪  dirs_exist_ok=True)
        else:
            desti.parent.mkdir(parents=True, exist_ok=True)
            shutil.copy2(origen, desti)
        log.info("OK    %s -> %s", origen, desti)
        return True
    except FileNotFoundError:
        log.error("No existeix: %s", origen)
    except PermissionError:
        log.error("Sense permisos: %s", origen)
    except shutil.Error as e:           # errors parcials dins d'un
     ↪  copytree
        log.error("Errors dins de %s: %s", origen, e)
```

```
    except OSError as e:                # disc ple, I/O, etc.
        log.error("Error de sistema amb %s: %s", origen, e)
    return False


def main() -> int:
    p = argparse.ArgumentParser(description=__doc__)
    p.add_argument("origens", nargs="+", type=Path, help="fitxers o
    ↪  carpetes")
    p.add_argument("-d", "--desti", required=True, type=Path,
    ↪  help="carpeta base")
    p.add_argument("--data", action="store_true",
                   help="crea una subcarpeta AAAA-MM-DD_HHMM dins del
                    ↪  destí")
    args = p.parse_args()

    logging.basicConfig(level=logging.INFO,
                        format="%(asctime)s %(levelname)s %(message)s")

    desti = ruta_absoluta(args.desti)
    if args.data:
        desti /= datetime.now().strftime("%Y-%m-%d_%H%M")

    errors = 0
    for org in args.origens:
        origen = ruta_absoluta(org)
        # Aquí sí cal resolve(): volem la ubicació real al disc
        if desti.resolve().is_relative_to(origen.resolve()):
            log.error("El destí és dins de l'origen, s'omet: %s",
            ↪  origen)
            errors += 1
            continue
        if not copia_dades(origen, ruta_desti(origen, desti)):
            errors += 1

    log.info("Acabat: %d origen(s), %d error(s)", len(args.origens),
    ↪  errors)
    return 1 if errors else 0


if __name__ == "__main__":
    sys.exit(main())
```

NOTA

`Path.is_relative_to` requereix Python 3.9 o superior.

Ús

```
# Còpia simple
python3 copies.py /etc/hosts /home/usuari/ document.txt -d /copies

# Amb subcarpeta amb data (conserva versions anteriors)
sudo python3 copies.py /etc /home/usuari -d /copies --data

# Comprovar el resultat
echo $?            # 0 = tot bé, 1 = hi ha hagut errors
ls -R /copies | head
```

Sortida d'exemple:

```
2026-11-25 11:14:45 INFO OK    /tmp/prova/origen ->
 ↪  /copies/2026-11-25_1114/tmp/prova/origen
2026-11-25 11:14:45 ERROR No existeix: /no/existeix
2026-11-25 11:14:45 INFO Acabat: 2 origen(s), 1 error(s)
```

11. Programar-lo: execució automàtica

Amb `cron`

```
crontab -e
```

```
# Cada dia a les 02:30
30 2 * * * /usr/bin/python3 /opt/scripts/copies.py /etc /home -d
 ↪  /copies --data >> /var/log/copies.log 2>&1
```

Cal usar rutes absolutes: `cron` té un `PATH` i un directori de treball mínims.

Amb un temporitzador de `systemd`

`/etc/systemd/system/copies.service`:

```
[Unit]
Description=Còpia de seguretat

[Service]
Type=oneshot
ExecStart=/usr/bin/python3 /opt/scripts/copies.py /etc /home -d
 ↪  /copies --data
```

`/etc/systemd/system/copies.timer`:

```
[Unit]
Description=Còpia de seguretat diària

[Timer]
OnCalendar=*-*-* 02:30:00
Persistent=true

[Install]
WantedBy=timers.target
```

```
sudo systemctl daemon-reload
sudo systemctl enable --now copies.timer
systemctl list-timers copies.timer
journalctl -u copies.service
```

Gràcies al codi de sortida, `systemd` marcarà el servei com a **failed** si hi ha errors.

12. Errors habituals i limitacions

- **`FileExistsError` en tornar a executar:** falta `dirs_exist_ok=True` (Vegeu secció 4.1).
- **`FileNotFoundError` en copiar un fitxer solt:** falta crear la carpeta pare amb `makedirs` (Vegeu secció 4.2).
- **El destí «desapareix»:** has fet `os.path.join(desti, '/ruta/absoluta')` (Vegeu secció 6).
- **Enllaços simbòlics:** per defecte, `copytree` copia el *contingut* apuntat. Un enllaç trencat dona error i un enllaç cap a un directori pare pot provocar una recursió. Amb `symlinks=True` es copia l'enllaç tal com és. Per a còpies d'`/etc/` o `/home/` és el que convé, i és el que fa el script de la secció 10.
- **`copytree` no sobreescriu enllaços:** si l'enllaç ja existia al destí, falla amb `File exists`. Per això el script esborra primer els enllaços de la còpia anterior (`esborra_enllacos`).
- **Còpia completa cada vegada:** aquest script ho copia tot, encara que no hagi canviat. Per a volums grans, l'eina adequada és `rsync -a` (còpies incrementals, propietaris i permisos). Vegeu rsync.
- **No és una còpia de seguretat completa.** Una còpia al mateix disc no protegeix d'un error de maquinari. Segueix la regla **3-2-1**: 3 còpies, 2 suports diferents i 1 fora de l'equip.
- **Bases de dades en marxa:** copiar els fitxers d'un servidor SQL actiu pot donar una còpia incoherent. Fes primer un *dump* (`mariadb-dump`, `mysqldump`, `pg_dump`) i copia el fitxer `.sql`.

13. Referència

Vull…	Faig servir
Saber si és carpeta / fitxer	`os.path.isdir(p)` / `os.path.isfile(p)`
Crear carpetes (i les pare)	`os.makedirs(p, exist_ok=True)`
Carpeta pare d'una ruta	`os.path.dirname(p)`
Unir rutes	`os.path.join(a, b)`
Copiar un fitxer amb metadades	`shutil.copy2(o, d)`
Copiar una carpeta sencera	`shutil.copytree(o, d, dirs_exist_ok=True)`
Ignorar patrons en copiar	`shutil.copytree(..., ignore=shutil.ignore_patterns('*.tmp'))`
Fer un `.tar.gz`	`shutil.make_archive(nom, 'gztar', carpeta)`

Índex de figures

1 Estratègia lògica 4

Versions d'aquest document

- HTML
- PDF
- ODT
- MD

Domini Públic (CC0)