Fiche Pratique — écrire & exécuter des scripts Python sur Windows / Linux / macOS

Vérifier Python (version + chemin)

Windows

```
py -V

py -0p

where python
```

Linux / macOS

```
python3 -V

which python3
```

Règle simple

- Windows : préfère `py`
- Linux/macOS : préfère `python3`

Lancer un script

Windows

```
py script.py

py -3.11 script.py
```

Linux / macOS

```
python3 script.py

python3.11 script.py
```

Un “script propre” (template minimal)

```
#!/usr/bin/env python3
from __future__ import annotations


import sys


def main() -> int:
    print("Hello")
    return 0


if __name__ == "__main__":
    raise SystemExit(main())
```

- `main()` + code de retour → pratique pour l’automatisation.
- `if __name__ == "__main__":` → évite l’exécution quand on importe.

Rendre exécutable (Linux/macOS)

Donner les droits

```
chmod +x script.py
```

Lancer directement

```
./script.py
```

Sur Windows, pas besoin de `chmod`. On lance via `py script.py`.

Environnements virtuels (venv) — indispensable en vrai

Créer un venv

Windows

```
py -m venv .venv
```

Linux/macOS

```
python3 -m venv .venv
```

Activer

Windows (PowerShell)

```
.\.venv\Scripts\Activate.ps1
```

Windows (cmd)

```
.\.venv\Scripts\activate.bat
```

Linux/macOS

```
source .venv/bin/activate
```

Désactiver (tous OS)

```
deactivate
```

Installer des dépendances (pip)

Toujours via `-m pip` (évite les mauvais pip) :

Windows

```
py -m pip install -U pip

py -m pip install requests

py -m pip freeze > requirements.txt

py -m pip install -r requirements.txt
```

Linux/macOS

```
python3 -m pip install -U pip
python3 -m pip install requests
python3 -m pip freeze > requirements.txt
python3 -m pip install -r requirements.txt
```

Arguments en ligne de commande (basique + robuste)

Minimal (sys.argv)

```
import sys
print(sys.argv)
```

Propre (argparse)

```
import argparse


p = argparse.ArgumentParser()
p.add_argument("target")
p.add_argument("--timeout", type=int, default=2)
args = p.parse_args()
print(args.target, args.timeout)
```

Chemins de fichiers (cross-platform)

Évite les `C:\...` et `/...` en dur. Utilise `pathlib` :

```
from pathlib import Path


base = Path(__file__).resolve().parent
inp = base / "data" / "targets.txt"
print(inp.read_text(encoding="utf-8"))
```

Encodage, fins de lignes, compatibilité

- Lis/écris en explicitant l'encodage :

```
open("file.txt", "r", encoding="utf-8")
```

- CSV propre :

```
import csv

with open("out.csv", "w", newline="", encoding="utf-8") as f:

    w = csv.writer(f)

    w.writerow(["col1", "col2"])
```

- Windows ↔ Unix : attention aux `\r\n` vs `\n`. Python gère bien, mais les outils autour parfois non.

Exécuter des commandes système

Simple

```
import subprocess

r = subprocess.run(["ping", "8.8.8.8"], capture_output=True,
text=True)

print(r.returncode)
```

Important

- **Windows vs Linux/macOS** : options différentes (ex: `ping -n` vs `ping -c`)
- Mets des timeouts si tu automatises :

```
subprocess.run(cmd, timeout=3)
```

Sorties “propres” pour l’automatisation

Logs (recommandé)

```
import logging

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

logging.info("Démarrage")
```

Codes de retour

- `0` = OK
- `>0` = erreur (utile pour scripts appelés par d’autres outils)

Planifier l’exécution (automation OS)

Windows — Task Scheduler

- Lance typiquement :

```
py C:\path\script.py
```

Linux — cron

```
crontab -e

# exemple : tous les jours à 07:00

0 7 * * * /usr/bin/python3 /opt/scripts/script.py
```

macOS — launchd (moderne)

- Utilise un `.plist` (plus propre que cron sur macOS).
  (Si tu veux, je te fournis un modèle prêt à coller.)

“Un seul fichier” portable : bonnes pratiques

- Pas de dépendances si tu peux (stdlib)
- Sinon : `requirements.txt`
- Documentation mini :
  - comment activer venv
  - comment lancer
  - exemples de commande
- Gestion d’erreurs claire (`try/except` + message)

Dépannage express

“python not found”

- Windows : utilise `py` (`py -V`)
- Linux/macOS : utilise `python3`

“pip installe au mauvais endroit”

- Utilise `python -m pip ...` (pas `pip ...`)

“Permission denied” (Linux/macOS)

```
chmod +x script.py
```

“ModuleNotFoundError”

- venv pas activé **ou** dépendance pas installée dans ce venv