

# Exercices — Décorateurs & argparse

---

## Exercice 1 : timestamp\_decorator

Créez un decorator `timestamp_decorator` qui affiche un horodatage au format `YYYY-MM-DD HH:MM:SS` à chaque appel de la fonction décorée. Appliquez-le à quelques fonctions simples (addition, multiplication, etc.) pour vérifier que le timestamp s'affiche bien.

## Exercice 2 : netinfo.py

Vous allez créer un outil en ligne de commande appelé `netinfo.py`. Cet outil regroupe deux fonctionnalités réseau accessibles via des sous-commandes.

### Fonctionnalités attendues

#### Sous-commande `scan`

L'utilisateur fournit une adresse IP et une liste de ports. Le script affiche, pour chaque port, s'il est "ouvert" ou "fermé".

Pour cet exercice, pas besoin de réellement tester les ports : simulez le résultat en considérant qu'un port pair est ouvert et un port impair est fermé.

Arguments :

- `host` (positionnel, obligatoire) : l'adresse IP cible
- `-p` / `--ports` (optionnel) : un ou plusieurs numéros de ports. Si non précisé, scanner les ports 22, 80 et 443 par défaut.

Exemple d'utilisation :

```
$ python netinfo.py scan 192.168.1.1
Scan de 192.168.1.1
Port 22: fermé
Port 80: ouvert
Port 443: fermé

$ python netinfo.py scan 10.0.0.1 -p 8080 3306
Scan de 10.0.0.1
Port 8080: ouvert
Port 3306: ouvert
```

#### Sous-commande `whois`

L'utilisateur fournit un nom de domaine. Le script affiche des informations fictives sur ce domaine (IP, registrar, date d'expiration). Les valeurs affichées peuvent être inventées, l'objectif est de pratiquer argparse.

## Arguments :

- **domaine** (positionnel, obligatoire) : le nom de domaine à interroger

## Exemple d'utilisation :

```
$ python netinfo.py whois example.com
Whois pour example.com
IP: 93.184.216.34
Registrar: Example Registrar Inc.
Expiration: 2026-08-13
```

## Flag global **--verbose**

Ajoutez un flag **-v / --verbose** au niveau du parser principal (pas sur les sous-commandes). Quand ce flag est activé, le script affiche le temps d'exécution de la commande à la fin.

Ce comportement doit être implémenté via un **decorator** appliqué aux fonctions de chaque sous-commande. Le decorator vérifie **args.verbose** et, si activé, mesure et affiche la durée.

```
$ python netinfo.py scan 192.168.1.1 -v
Scan de 192.168.1.1
Port 22: fermé
Port 80: ouvert
Port 443: fermé

[Exécution: 0.001s]
```

## Contraintes

- Le script doit afficher l'aide (**--help**) correctement, y compris pour chaque sous-commande (`python netinfo.py scan --help`)
- Si l'utilisateur lance le script sans sous-commande, afficher l'aide générale
- Utilisez **nargs="+"** pour accepter plusieurs ports
- Le decorator de timing doit fonctionner de manière transparente : les fonctions **cmd\_scan** et **cmd\_whois** ne doivent contenir aucune logique de mesure de temps

## Indices

- **argparse.ArgumentParser** pour le parser principal
- **parser.add\_subparsers(dest="commande")** pour les sous-commandes
- **time.time()** pour mesurer la durée
- Le decorator reçoit **args** comme argument et peut accéder à **args.verbose**