Scripting

- Scripting en python (Subprocess)

Scripting en python (Subprocess)

Pour lancer proprement des commandes en Python avec subprocess tout en gérant les erreurs, il est recommandé d'utiliser subprocess.run() avec les options appropriées et un bloc try/except. Voici la méthode standard et robuste :

```
' import subprocess

try:
result = subprocess.run(
["votre_commande", "arg1", "arg2"], # Commande sous forme de liste
check=True, # Lève une exception si le code retour ≠ 0
capture_output=True, # Capture stdout et stderr
text=True, # Retourne les sorties en texte (str)
timeout=30 # (optionnel) Limite de temps en secondes
)
print("Sortie standard :", result.stdout)
except subprocess.CalledProcessError as e:
print(f"Commande échouée avec le code retour {e.returncode}")
print("Erreur :", e.stderr)
except FileNotFoundError:
print("La commande est introuvable.")
except subprocess.TimeoutExpired:
print("La commande a dépassé le temps imparti.")
except Exception as e:
print(f"Erreur inattendue : {e}")

'
```

Explications clés :

check=True : lève une exception (CalledProcessError) si la commande échoue.

capture_output=True : récupère la sortie standard et d'erreur.

timeout=30 : évite qu'un processus bloqué ne gèle votre script.

Gestion fine des exceptions : chaque erreur courante (CalledProcessError, FileNotFoundError, TimeoutExpired) est traitée séparément pour des messages clairs.

À éviter :

N’utilise pas de simple except Exception sans gérer les cas spécifiques, pour ne pas masquer la nature de l’erreur.

Privilégie la forme liste pour la commande (plutôt qu’une chaîne avec shell=True), sauf si tu as vraiment besoin d’une interprétation shell.

Ce schéma garantit une exécution propre, un diagnostic efficace et une bonne robustesse de ton code Python.