## GABRIELE ROMANATO

### Python: eseguire comandi della shell

Scrivere uno script Python che esegua comandi della shell offre un potente strumento per automatizzare operazioni di sistema, gestire processi e interagire con il sistema operativo. Python fornisce diverse librerie per eseguire comandi shell, ognuna con i propri vantaggi e casi d'uso. In questo articolo, esploreremo alcuni dei modi principali per eseguire comandi della shell in Python, coprendo le librerie `subprocess`, `os`, e `sh`.

## 1. Utilizzo di `subprocess`

La libreria `subprocess` è lo strumento più flessibile e raccomandato per l'esecuzione di comandi shell in Python. Offre una grande flessibilità permettendo l'esecuzione di comandi nuovi, la connessione ai loro input/output/error pipes e l'ottenimento dei loro codici di ritorno.

```
import subprocess

# Eseguire un comando semplice
result = subprocess.run(["ls", "-l"], capture_output=True,
text=True)
print(result.stdout)

# Esecuzione di un comando e cattura dell'output
try:
    output = subprocess.check_output(["ls", "-l"], text=True)
    print(output)
except subprocess.CalledProcessError as e:
    print(e)
```

## 2. Utilizzo di `os.system` e `os.popen`

Prima dell'introduzione di `subprocess`, il modulo `os` era comunemente usato per eseguire comandi shell. Tuttavia, rispetto a `subprocess`, offre meno controllo e flessibilità.

`os.system()`: Esegue il comando indicato in una subshell.

```
import os

# Esempio di utilizzo os.system
os.system('ls -l')
```

`os.popen()`: Esegue il comando indicato in una subshell e restituisce un oggetto file collegato all'input o all'output standard del comando (a seconda dei parametri).

```
# Esempio di utilizzo os.popen
stream = os.popen('ls -l')
output = stream.read()
print(output)
```

## 3. Utilizzo di `sh` (solo su Unix-like)

Il modulo `sh` è un wrapper per `subprocess` progettato per rendere più semplice e intuitiva l'esecuzione di comandi shell. Non è incluso nella libreria standard di Python e funziona solo su sistemi Unix-like.

```
from sh import ls

# Esempio di utilizzo di sh
print(ls("-l"))
```

## Considerazioni sulla Sicurezza

Quando si eseguono comandi shell da uno script Python, è importante considerare le implicazioni per la sicurezza, specialmente se si lavora con input forniti dall'utente. Assicurati di sanificare tutti gli input per evitare vulnerabilità come l'iniezione di comandi.

## Conclusione

Python offre diverse opzioni per eseguire comandi della shell, ciascuna con i propri vantaggi. `subprocess` è la scelta più potente e flessibile per la maggior parte delle applicazioni, mentre `os` e `sh` possono essere opzioni più semplici per casi d'uso specifici. La selezione della libreria giusta dipende dalle esigenze specifiche del tuo progetto e dai requisiti di sicurezza.