LU08a - Einführung Bash-Befehle in Python

Python ist nicht nur eine vielseitige Programmiersprache für Anwendungen, Datenanalyse und maschinelles Lernen, sondern eignet sich auch hervorragend, um mit der Kommandozeile zu interagieren und Bash-Befehle auszuführen. Diese Funktionalität ist besonders nützlich, wenn Sie Skripte schreiben, die Prozesse automatisieren, Daten sammeln oder Systemaufgaben ausführen müssen.

Warum Bash-Befehle in Python nutzen?

Die Integration von Bash-Befehlen in Python bietet die Möglichkeit:

- vorhandene Bash-Skripte und Befehle wiederzuverwenden,
- komplexe Workflows zu automatisieren,
- sowohl die Vorteile von Python (Lesbarkeit, Datenverarbeitung) als auch von Bash (Systemzugriff) zu kombinieren.

Sicherheitshinweise

Eingaben validieren

Vermeiden Sie die direkte Übergabe von Benutzereingaben an Shell-Befehle, um Sicherheitsrisiken wie Code-Injection zu verhindern.

Beispiel:

```
# Unsicher: Benutzer kann Schadcode einschleusen
os.system(f"rm -rf {user_input}")
```

Stattdessen:

```
subprocess.run(['rm', '-rf', user_input], check=True)
```

Fehlerbehandlung

Überprüfen Sie Rückgabecodes und Ausgaben auf Fehler.

```
import subprocess

# Bash-Befehl, der ausgeführt werden soll
command = "ls -l /"

try:
    # Ausführung des Befehls
```

```
    result = subprocess.run(
        command,          # Der Befehl als String
        shell=True,       # Ausführung in einer Shell
        check=True,       # Fehler werfen, wenn der Rückgabecode ungleich 0
ist
        text=True,        # Gibt die Ausgaben als Strings zurück (anstatt
Bytes)
        stdout=subprocess.PIPE,  # Standardausgabe umleiten
        stderr=subprocess.PIPE   # Standardfehlerausgabe umleiten
    )
    # Ausgabe des Ergebnisses
    print("Befehl erfolgreich ausgeführt!")
    print("Rückgabecode:", result.returncode)
    print("Ausgabe:")
    print(result.stdout)
except subprocess.CalledProcessError as e:
    # Fehlerbehandlung, wenn der Befehl fehlschlägt
    print("Fehler bei der Ausführung des Befehls!")
    print("Rückgabecode:", e.returncode)
    print("Fehlermeldung:")
    print(e.stderr)
except Exception as ex:
    # Generelle Fehlerbehandlung
    print(f"Ein unerwarteter Fehler ist aufgetreten: {ex}")
```

Fazit

Python bietet flexible Werkzeuge zur Integration von Bash-Befehlen und ist eine ideale Wahl für die Entwicklung leistungsstarker Automatisierungs- und Verwaltungsskripte. Das `subprocess`-Modul ist die bevorzugte Methode, da es präzise Kontrolle und Sicherheit bietet. Durch das Verständnis dieser Techniken können Sie die Stärken von Python und Bash optimal kombinieren, um Ihre Workflows effizienter zu gestalten.