

# Python für System Administratoren



## Inhaltsverzeichnis

Aufgabe 1 - Dateien umbenennen.....	2
Aufgabe 2 - Filial-Umsätze zusammenfassen.....	3
Aufgabe 3 - Prozesse überwachen.....	4
Aufgabe 4 - Doppelte Dateien suchen.....	5
Aufgabe 5 - Verzeichnisse sichern.....	6
Aufgabe 6 - Qualitätssicherungsprogramme ausführen.....	7
Aufgabe 7 - Verzeichnisstruktur visualisieren.....	8
Aufgabe 8 - Youtube Videos runter laden.....	9
Aufgabe 9 - Prozess überwachen und bei Bedarf nachstarten.....	10
Aufgabe 10 - Umsatzdaten täglich per Email versenden.....	11
Aufgabe 11 - PDF-Dateien erzeugen.....	12
Aufgabe 12 - Dateien pro Dateityp zählen.....	13
Aufgabe 13 - Filialumsatzdatei anpassen.....	14
Aufgabe 14 - Prozesse starten.....	15
Aufgabe 15 - Kundenstruktur anlegen.....	16
Aufgabe 16 - Kundendaten sichern.....	17
Aufgabe 17 - Erzeugung eines SQL-Skripts.....	18

## Aufgabe 1 - Dateien umbenennen



### Aufgabenstellung

Die Namen aller Dateien in einem Verzeichnis sollen geändert werden. Da es sehr viele Dateien sind und diese Aufgabe wahrscheinlich demnächst regelmäßig auf Sie zukommt, soll ein Python Programm erstellt werden, welches diese Aufgabe automatisch durchführt. Python unterstützt das Arbeiten mit Verzeichnissen und Dateien durch die Pakete `os` und `shutil`.

Bei der Namensänderung soll bei allen betroffenen Dateien ein Substring durch einen anderen Substring ersetzt werden. Z.B. soll in allen Dateinamen der Substring `x77` durch `alpa95` ersetzt werden.

### Lösung

- `pfsa_01.py`

### Alternative Aufgabenstellung

Sie sollen in Zukunft für ein Python Programm zur Umbenennung von Dateien verantwortlich sein. Das Programm `pfsa_01.py` liegt Ihnen vor. Analysieren Sie das Programm und bearbeiten Sie folgende Aufgaben bzw. Fragen:

- Was macht das Programm genau? Organisieren Sie sich Beispieldaten und führe Sie die Funktionalität vor.
- Welche anderen Pakete/Module werden von dem Programm genutzt? Wofür sind die Module jeweils da (Ein Satz reicht als Antwort)? Visualisieren Sie die Modulhierarchie graphisch.
- Das Programm ist funktional zerlegt. Visualisieren Sie die Funktionshierarchie graphisch.
- In welcher Funktion des Programms wird die eigentliche Umbenennung vorgenommen? Welche Hilfsfunktion aus einem benutzten Modul wird hierbei genutzt?
- In der Funktion `hauptprogramm` und in anderen gibt es die Variablen `anzahl_dateien_gesamt` und `anzahl_dateien_betroffen`, Was unterscheidet diese?
- Von welchem Datentyp ist die Variable `datei_liste_betroffen`?
- In welcher Zeile wird bestimmt, ob ein Dateiname geändert werden muss?

## Aufgabe 2 - Filial-Umsätze zusammenfassen

### Aufgabenstellung



Während der Nacht wird pro Filiale eine CSV-Datei mit den Umsätzen des Vortags an die Zentrale gesendet. Der Dateiname besitzt das Format `Filialname_Datum.csv`, so dass aus ihm sowohl die Filiale als auch das Datum hervorgeht. In der Datei gibt es pro Warengruppe eine Zeile, die durch Komma getrennt den Warengruppenamen und den Umsatz pro Warengruppe enthält.

Morgens müssen Sie die Dateien zu einer CSV-Datei zusammen fassen. Der Dateiname soll `umsatzuebersicht_Datum.csv` lauten. Die Datei soll pro Filiale und Warengruppe eine Zeile enthalten, die durch Komma getrennt folgende Informationen enthält: Datum, Filiale, Warengruppe, Umsatz.

Da diese Routineaufgabe schnell langweilig wird, sollen Sie die Arbeit mit einem Python-Skript automatisieren. Beispieldaten zum Testen liegen vor. Die Filial-Dateien liegen immer in einem eigenen Verzeichnis bereit, welches keine anderen Dateien enthält. In diesem soll auch die Übersichts-Datei erstellt werden.

### Lösung

- `pfsa_02.py`

## Aufgabe 3 - Prozesse überwachen

### Aufgabenstellung

In der Datei `pfsa_03.ini`, befinden sich zeilenweise die Namen von Prozessen. Diese Prozesse sollen von einem Python-Programm überwacht werden, d.h. das Python-Programm testet in regelmäßigen Abständen, ob die Prozesse laufen. Wenn der Prozess läuft soll zusätzlich angegeben werden, wie viele Instanzen mit diesem Namen laufen. Die Ausgabe soll auf der Konsole erfolgen. Die Dauer des Überwachungsintervalls soll durch eine Konstante festgelegt werden.



### Ausbaumöglichkeiten

- Die Ergebnisse werden zusätzlich in der Datei `pfsa_03.log` gespeichert.

### Lösung

- `pfsa_03.py`

## Aufgabe 4 - Doppelte Dateien suchen



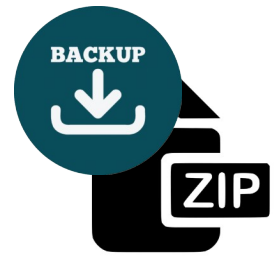
### Aufgabenstellung

Ausgehend von einem durch den Benutzer einzugebenden Basis-Verzeichnis, soll die darunter liegende Verzeichnisstruktur nach doppelten Dateien durchsucht werden. Zwei Dateien gelten hierbei vereinfacht als gleich, wenn sie den gleichen Namen besitzen. Die Ausgabe soll in eine Text-Datei erfolgen.

### Lösung

- `pfsa_04.py`

## Aufgabe 5 - Verzeichnisse sichern



### Aufgabenstellung

Automatisieren die Datensicherung mit einem Python-Programm. In der Datei `pfsa_05.ini`, befinden sich zeilenweise die Namen von Verzeichnissen (vollständiger Pfad). Diese Verzeichnisse sollen jeweils in einer Zip-Datei gesichert werden. Der Name der Zip-Datei soll `Verzeichnis_Datum.zip` lauten (diesmal nur eigentlicher Verzeichnisname ohne Pfad). Die Zip-Dateien sollen in einem Sicherungsverzeichnis abgelegt werden. Der Name der Ini-Datei und des Sicherungsverzeichnisses sind als Konstanten im Python-Programm zu definieren.

### Hinweise

In dem Paket `shutil` finden Sie Funktionalität zum zippen und Dateien verschieben.

### Lösung

- `pfsa_05.py`

## Aufgabe 6 - Qualitätssicherungsprogramme ausführen



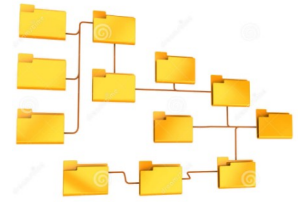
### Aufgabenstellung

Laut Programmierrichtlinien sollen als Maßnahmen zur Qualitätssicherung die Formatierungswerkzeuge `autopep8` und `black`, so wie die Code Analysatoren `flake8` und `pylint` über alle erstellten Python Dateien laufen. Schreiben Sie ein Python-Programm, welches die vier Werkzeuge automatisch ausführt. Die zu überprüfenden Dateien werden über die Kommandozeile mitgegeben. Konfigurieren Sie die vier Werkzeuge so, dass eine möglichst große Übereinstimmung mit den BK-GuT-Programmierrichtlinien hergestellt wird. Zur Konfiguration kann eine Konfigurationsdatei oder Kommandozeilenargumente genutzt werden.

### Lösung

- `pfsa_06.py`

## Aufgabe 7 - Verzeichnisstruktur visualisieren



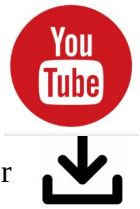
### Aufgabenstellung

Ausgehend von einem durch den Benutzer einzugebenden Basis-Verzeichnis, soll die darunter liegende Verzeichnisstruktur visualisiert werden. Pro Ebene soll jeweils um 2 Leerzeichen weiter eingerückt werden. Für jedes Verzeichnis soll angegeben werden, wie viele Unterverzeichnisse und Dateien es enthält. Die Ausgabe soll in eine Text-Datei erfolgen.

### Lösung

- `pfsa_07.py`

## Aufgabe 8 - Youtube Videos runter laden



### Aufgabenstellung

Sie arbeiten in einer Werbe-Agentur und müssen öfters für den Chef und andere Mitarbeiter Youtube Videos runter laden. Schreiben Sie ein Python-Programm, welches automatisch Youtube Videos runter lädt. Die URL-Adressen befinden sich in der Konfigurations-Datei `pfsa_08.ini`. Als Hilfe können Sie das Paket `pytube` nutzen.

### Lösung

- `pfsa_08.py`

## Aufgabe 9 - Prozess überwachen und bei Bedarf nachstarten



### Aufgabenstellung

Alle 15 Minuten soll überprüft werden, ob ein bestimmter Prozess noch läuft. Es soll beispielhaft der Editor (`notepad.exe`) genutzt werden. Wenn der Prozess nicht mehr läuft, soll er nachgestartet werden. Das Python-Programm soll sinnvolle Meldungen über seine Aktivitäten auf der Konsole ausgeben. Nehmen Sie während der Entwicklung ein Intervall von 5 Sekunden. Benutzen Sie das Paket `schedule` um einen zentralen Scheduler zu realisieren.

### Ausbaumöglichkeiten

- Die Ergebnisse werden zusätzlich in der Datei `pfsa_09.log` gespeichert.

### Lösung

- `pfsa_09.py`

## Aufgabe 10 - Umsatzdaten täglich per Email versenden



### Aufgabenstellung

Jeden Tag um 8:00 Uhr soll eine Datei mit Umsatzdaten (`pfsa_10.csv`) versendet werden. Automatisieren Sie diese Aufgabe mit einem Python-Programm. Benutzen Sie das Paket `schedule` um einen zentralen Scheduler zu realisieren, das Paket `smtpplib` für den Zugriff auf den Email-Server.

### Lösung

- `pfsa_10.py`

## Aufgabe 11 - PDF-Dateien erzeugen



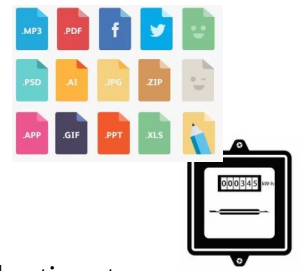
### Aufgabenstellung

Sie müssen regelmäßig aus einer Vielzahl von Berichten, die als Word-Dateien (DOCX-Dateien) vorliegen, zur Archivierung und Weiterverarbeitung PDF-Dateien erstellen. Da die manuelle Erstellung der PDF-Dateien zeitaufwendig und langweilig ist, wollen Sie die Arbeit mit einem Python-Programm automatisieren. Die DOCX-Dateien liegen in einem Verzeichnis, die PDF-Dateien sollen in ein zweites kommen. Als Hilfsmittel können Sie das Paket `docx2pdf` nutzen.

### Lösung

- `pfsa_11.py`

## Aufgabe 12 - Dateien pro Dateityp zählen



### Aufgabenstellung

Ausgehend von einem durch den Benutzer einzugebenden Basis-Verzeichnis, soll bestimmt werden, welche Dateitypen es in dem Basis-Verzeichnis inklusive Unterverzeichnissen gibt. Ebenso soll ausgegeben werden, wie viele Dateien es pro Typ gibt und die Gesamtanzahl von Dateien

### Lösung

- `pfsa_12.py`

## Aufgabe 13 - Filialumsatzdatei anpassen



### Aufgabenstellung

Die Filialumsätze eines Tages sind bereits in einer Datei namens `umsatzuebersicht_datum.csv` zusammengefasst (siehe Aufgabe 2). Bevor die Datei an den amerikanischen Mutterkonzern weitergeleitet werden kann, muss das Datumsformat der ersten Spalte von `ttmmjjjj` ohne trennende Punkte auf das amerikanische Format `mm/tt/jjjj` umgestellt werden.

Da diese Routineaufgabe schnell langweilig wird, sollen Sie die Arbeit mit einem Python-Skript automatisieren. Die Namen der Eingabedatei und der bereinigten Ausgabedatei sollen als Kommandozeilenparameter mitgegeben werden. Beispieldaten zum Testen liegen vor (`umsatzuebersicht_13072022.csv`). Als Hilfsmittel können Sie die Pakete `pandas` und `argparse` nutzen.

### Lösung

- `pfsa_13.py`

## Aufgabe 14 - Prozesse starten



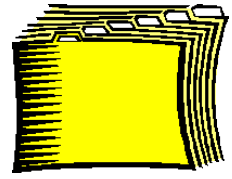
### Aufgabenstellung

Erstellen Sie ein Python-Programm, welches drei Programme startet. Starten Sie beispielhaft den Texteditor (notepad), den Taschenrechner (calc) und die DOS-Command-Box (cmd). Als Hilfsmittel können Sie das Paket `os` nutzen.

### Lösung

- `pfsa_14.py`

## Aufgabe 15 - Kundenstruktur anlegen



Erstellen Sie ein Python-Programm, welches ein neues Kundenverzeichnis anlegt. Das Skript soll folgende Funktionalität bereitstellen:

1. Das Skript fragt den Benutzer nach der `KundenID` und dem `Kundennamen`.
2. Es wird ein neues Kundenverzeichnis angelegt. Der Name des Verzeichnisses ist die `KundenID`. In welchem Wurzelverzeichnis das Kundenverzeichnis anzulegen ist, steht in der Umgebungsvariablen `KUNDENDATENWURZEL`. Diese müssen Sie selber vorher in Windows anlegen.
3. In dem Wurzelverzeichnis gibt es bereits ein Unterverzeichnis `Vorlagen`. Alle in diesem Verzeichnis befindlichen Dateien sollen in das neu erstellte Kundenverzeichnis kopiert werden. Ein Vorlagenverzeichnis finden Sie in Logineo als Zip-Datei.
4. Es ist eine Datei mit dem Namen `Kundendaten_KundenID.txt` zu erstellen, die die folgenden zwei Zeilen als Inhalt enthält:

```
ID: KundenID  
Name: Kundennamen
```

Das Skript soll den Benutzer über die Aktionen informieren.

### Lösung

- `pfsa_15.py`

### Erweiterungsmöglichkeiten

- Das Python-Programm ist mit einem GUI zu versehen.

## Aufgabe 16 - Kundendaten sichern



In einem Wurzelverzeichnis gibt es pro Kunden ein Unterverzeichnis. Der Name des Kundenverzeichnisses ist die KundenID. Der Name des Wurzelverzeichnisses steht in der Umgebungsvariablen KUNDENDATENWURZEL.

Erstellen Sie ein Python-Programm, welches Unterverzeichnisse sichert. Das Python-Programm soll folgende Funktionalität bereitstellen:

1. Pro Unterverzeichnis wird eine ZIP-Datei erstellt. Der Name der Zip-Datei lautet `sicherung_ jjjjmmtt_verzeichnisname.zip`. Sollte diese bereits bestehen, so wird die bestehende gelöscht.
2. Die erzeugten ZIP-Dateien sind in einem Sicherungsverzeichnis abzuspeichern. Der Name des Sicherungsverzeichnisses steht in der Umgebungsvariablen KUNDENDATENSICHERUNG.
3. Pro Sicherung ist eine Log-Datei zu erstellen. Der Name der Log-Datei lautet `sicherung_ jjjjmmtt_log.txt`. In der Log-Datei steht zeilenweise der Name des Unterverzeichnisses und der Name der ZIP-Datei.

Das Skript soll den Benutzer über die Aktionen informieren.

### Lösung

- `pfsa_16.py`

### Anmerkung

- Die Aufgabe knüpft an Aufgabe 15.

### Erweiterungsmöglichkeiten

- Die erzeugten Sicherungen werden zusätzlich auf einem Cloud-Laufwerk abgespeichert.
- Das Python-Programm ist mit einem GUI zu versehen.

## Aufgabe 17 - Erzeugung eines SQL-Skripts



Erstellen Sie ein Python-Programm, welches ein SQL-Skript für eine MySQL-Datenbank erstellt.

- a) Zuerst wird der Vor- und Nachname einer Person abgefragt.
- b) Anschließend wird ein SQL-Skript erzeugt, welches drei SQL-Anweisungen enthält.
  1. Die erste Anweisung legt einen neuen Benutzer `Nachname_Vorname` an.
  2. Die zweite Anweisung legt eine Datenbank namens `db_Nachname_Vorname` an.
  3. Die dritte Anweisung räumt dem Benutzer sämtliche Rechte auf der Datenbank zu.

Das SQL-Skript soll den Dateinamen `benutzer.sql` besitzen.

### Lösung

- `pfsa_17.py`