Hausaufgaben: Datenverarbeitung mit Python

Jowan Sulaiman
ABS IT

3. Juni 2025

Hausaufgabe 1: Grundlagen der Dateibearbeitung in Python

Theoriefragen

Beantworte die folgenden Fragen in eigenen Worten. Schreibe deine Antworten in eine separate Text- oder Markdown-Datei.

1. Was ist der Unterschied zwischen den Dateiöffnungsmodi `"r"`, `"w"` und `"a"` in Python? Erkläre kurz, was passiert, wenn du eine Datei in jedem dieser Modi öffnest.
2. Warum ist die Zeile `with open("meinedatei.txt", "r") as f:` eine gute Methode, um Dateien in Python zu öffnen und zu lesen? Nenne einen wichtigen Vorteil.
3. Stell dir vor, du hast eine sehr große Textdatei (z.B. ein Logbuch mit vielen Einträgen). Wäre es besser, die Datei Zeile für Zeile zu lesen oder den gesamten Inhalt auf einmal? Begründe kurz.
4. Das `os`-Modul in Python ist nützlich für Dateioperationen. Nenne eine Funktion aus dem `os`-Modul und beschreibe, was sie tut.

Praktische Aufgaben

Aufgabe 1.1: Mein einfaches Notizbuch

Schreibe ein Python-Skript, das als einfaches Notizbuch dient.

- Das Programm soll den Benutzer fragen, ob er eine neue Notiz hinzufügen oder alle Notizen anzeigen möchte.
- `Notiz hinzufügen:` Wenn der Benutzer eine Notiz hinzufügen möchte, soll das Programm ihn nach der Notiz fragen (eine einzelne Zeile Text genügt). Diese Notiz soll dann in einer Datei namens `meine_notizen.txt` gespeichert werden. Jede neue Notiz soll in einer neuen Zeile angehängt werden.
- **Notizen anzeigen:** Wenn der Benutzer die Notizen anzeigen möchte, soll das Programm den gesamten Inhalt der Datei `meine_notizen.txt` lesen und auf der Konsole ausgeben.
- **Beispiel-Interaktion:**

```
Was möchtest du tun? (schreiben / lesen / beenden): schreiben
Gib deine Notiz ein: Heute Python gelernt!
Notiz gespeichert.

Was möchtest du tun? (schreiben / lesen / beenden): lesen
--- Meine Notizen ---
Heute Python gelernt!
--- Ende der Notizen ---

Was möchtest du tun? (schreiben / lesen / beenden): beenden
```

Tipp

Verwende den Modus `"a"` zum Anhängen an die Datei und `"r"` zum Lesen. Denke an den `with`-Block!

Aufgabe 1.2: Datei-Inspektor

Schreibe ein Python-Skript, das den Benutzer nach einem Dateinamen fragt und dann versucht, den Inhalt dieser Datei anzuzeigen.

- Frage den Benutzer: `"Welche Datei möchtest du anzeigen?"`
- Lies den Dateinamen ein.
- Versuche, die Datei zu öffnen und ihren gesamten Inhalt auf der Konsole auszugeben.
- **Fehlerbehandlung:** Wenn die Datei nicht gefunden wird, soll das Programm nicht abstürzen, sondern eine freundliche Meldung ausgeben, z.B. `"Datei '[Dateiname]' nicht gefunden."`

Tipp

Verwende einen `try-except`-Block, um den `FileNotFoundError` abzufangen.

Aufgabe 1.3: Ordner-Inhalt auflisten

Schreibe ein Python-Skript, das alle Dateien und Ordner in einem vom Benutzer angegebenen Verzeichnis auflistet.

- Frage den Benutzer: `"Welchen Ordner möchtest du auflisten?"`
- Lies den Pfad zum Ordner ein.
- Verwende das `os`-Modul (insbesondere `os.listdir()`), um alle Einträge (Dateien und Unterordner) in diesem Verzeichnis aufzulisten und ihre Namen auf der Konsole auszugeben.
- **Fehlerbehandlung:** Gib eine Meldung aus, falls der angegebene Pfad kein gültiges Verzeichnis ist.

Hausaufgabe 2: Arbeiten mit JSON-Daten

Theoriefragen

Beantworte die folgenden Fragen in eigenen Worten.

1. Was ist JSON? Gib ein einfaches Beispiel, wofür man JSON verwenden könnte.
2. Nenne drei verschiedene Datentypen, die in JSON verwendet werden können (z.B. String, Zahl, ...). Gib für jeden dieser Typen an, welchem Python-Datentyp er entspricht.
3. Erkläre kurz den Unterschied zwischen `json.dumps()` und `json.dump()` in Python. (Ein Satz pro Funktion genügt).
4. Wenn du eine Python-Struktur (z.B. ein Dictionary) mit `json.dumps(daten, indent=4)` in einen JSON-String umwandelst, was bewirkt der Parameter `indent=4`?

Praktische Aufgaben

Aufgabe 2.1: Mein Lieblingsessen speichern

Schreibe ein Python-Skript, das Informationen über dein Lieblingsessen in einer JSON-Datei speichert und wieder ausliest.

- Erstelle ein Python-Dictionary, das dein Lieblingsessen beschreibt. Es sollte mindestens die Schlüssel `"name"` (z.B. `"Pizza Margherita"`), `"kategorie"` (z.B. `"Italienisch"`) und `"bewertung"` (eine Zahl von 1 bis 5) enthalten.
- Speichere dieses Dictionary als schön formatierte JSON-Datei unter dem Namen `lieblingsessen.json`. Verwende eine Einrückung von 2 Leerzeichen.
- Schreibe dann Code, der die Datei `lieblingsessen.json` wieder einliest und den Namen deines Lieblingsessens auf der Konsole ausgibt.

Tipp

Du benötigst `json.dump()` zum Speichern und `json.load()` zum Laden.

Aufgabe 2.2: Einfache Kontaktliste

Du möchtest eine kleine Liste von Kontakten verwalten.

- Erstelle eine Python-Liste, die zwei oder drei Kontakte enthält. Jeder Kontakt soll ein Dictionary sein mit den Schlüsseln `"name"` und `"telefonnummer"`. *Beispiel:*

```
kontakte = [
    {\texttt{"}name\texttt{"}: \texttt{"}Max Mustermann\texttt{"}, \texttt{"}telefonnummer\texttt{"}: \texttt{"}0123-45678\texttt{"}},
    {\texttt{"}name\texttt{"}: \texttt{"}Erika Musterfrau\texttt{"}, \texttt{"}telefonnummer\texttt{"}: \texttt{"}0987-65432\texttt{"}}
]
```

- Speichere diese Liste von Kontakten in einer JSON-Datei namens `kontakte.json`.
- Schreibe Code, der `kontakte.json` einliest und alle Namen und Telefonnummern formatiert auf der Konsole ausgibt.

Aufgabe 2.3: JSON-String verarbeiten

Gegeben ist der folgende Python-String, der JSON-Daten enthält:

```
json_text = '{\texttt{"}buch_titel\texttt{"}: \texttt{"}Python fortgeschritten\texttt{"}, \texttt{"}kapitel\texttt{"}: 10, \texttt{"}verfuegbar\texttt{"}: true}'
```

- Wandle diesen JSON-String in ein Python-Dictionary um.
- Gib den Wert des Schlüssels `"buch_titel"` aus.
- Gib den Wert des Schlüssels `"kapitel"` aus.

Tipp

Verwende `json.loads()` (achte auf das 's' am Ende!).

Hausaufgabe 3: Einführung in Reguläre Ausdrücke (RegEx)

Theoriefragen

Beantworte die folgenden Fragen in eigenen Worten.

1. Was ist ein Regulärer Ausdruck (kurz RegEx)? Versuche, es mit einem einfachen Beispiel zu erklären.
2. Erkläre kurz die Bedeutung der folgenden einfachen RegEx-Zeichen/Metazeichen:
   - `.` (der Punkt)
   - `*` (der Stern)
   - `\d` (Backslash d)
   - `[]` (eckige Klammern)
3. Was ist der Unterschied zwischen `re.search()` und `re.findall()` aus dem Python `re`-Modul?
4. Warum ist es oft eine gute Idee, Reguläre Ausdrücke in Python als "Raw Strings" zu schreiben (z.B. `r"mein_muster"`)?

Praktische Aufgaben

Hinweis zu RegEx

Reguläre Ausdrücke können anfangs knifflig sein. Nutze Online-Tools wie regex101.com (stelle die "Flavor" auf Python ein), um deine Muster zu testen und besser zu verstehen!

Aufgabe 3.1: Finde die Zahlen

Gegeben ist der folgende Text:

```
"Das Produkt A1 kostet 25 Euro, Produkt B22 kostet 199 Euro und Artikel C333 kostet 12 Euro."
```

- Schreibe ein Python-Skript, das alle Preisangaben (nur die Zahlen, z.B. 25, 199, 12) aus diesem Text extrahiert und als Liste ausgibt.

Tipp

Du suchst nach einer oder mehreren Ziffern. `\d+` könnte hier nützlich sein. Verwende `re.findall()`.

Aufgabe 3.2: Einfache "Hallo"-Suche

Schreibe ein Python-Skript, das prüft, ob das Wort "Hallo" (genau so geschrieben, mit großem 'H') in einem vom Benutzer eingegebenen Text vorkommt.

- Frage den Benutzer nach einem Satz.
- Verwende `re.search()`, um zu prüfen, ob "Hallo" im Satz enthalten ist.
- Gib aus: "Hallo gefunden!" oder "Hallo nicht gefunden."

Aufgabe 3.3: Wörter mit 'test' finden

Gegeben ist der Satz:

`"Das ist ein Test. Dieser Testfall ist wichtig. Ich teste gerne."`

- Schreibe ein Python-Skript, das alle Wörter findet, die die Zeichenkette `"test"` (unabhängig von Groß-/Kleinschreibung) enthalten.
- Gib die gefundenen Wörter als Liste aus. (Erwartete Ausgabe könnte sein: ['Test', 'Testfall', 'teste'])

Tipp

Suche nach Wörtern, die `"test"` enthalten. `\w*test\w*` könnte ein Ansatz sein. Um Groß-/Kleinschreibung zu ignorieren, kannst du das Flag `re.IGNORECASE` verwenden (z.B. `re.findall(muster, text, re.IGNORECASE)`).