CloudCommand

# Cyber Security

CloudCommand GmbH chr.schumacher@gmx.tm

CloudCommand

# Grundlagen der Programmierung

CloudCommand GmbH chr.schumacher@gmx.tm

# Datei-verwaltung

# Lesen und schreiben einer Datei

- Python bringt von Haus aus einfache Möglichkeiten, um Dateien in eigenen Programmen zu handeln.
- Mit der Funktion ‚open' wird eine Datei geöffnet. Das Öffnen einer Datei kann in verschiedenen Modi geschehen (nächstes Kapitel)
- Nachdem eine Datei geöffnet ist kann sie eingelesen bzw. in sie hinein geschrieben werden.
- Zum Schluss der Dateioperation wird die Datei mit ‚close' geschlossen.

**Guter Programmierstil:**

- Mittels ‚try ... / except ...' sollten Fehler abgefangen werden. (z.B. ‚File not found ...')

# Lesen und schreiben einer Datei

```python
# HINWEIS:
# Das Programm erwartet die Datei 'beispieldatei.txt' im
# gleichen Verzeichnis, wie die 'main.py'

# Das Beispiel zeigt, wie eine komplette Datei eingelesen wird.


if __name__ == "__main__":
    print("\n\nInhalt der Datei 'beispieldatei.txt':\n<--Anfang-->")

    EineVariable = open('beispieldatei.txt', 'r')
    print(EineVariable.read())
    EineVariable.close()

    print("<--Ende-->")
```

# Lesen und schreiben einer Datei

**EineVariable = open(‘beispieldatei.txt’, ‘r’)**

- Die Datei wird zum Lesen geöffnet und der Inhalt der Variablen zugeordnet.
- Mögliche Parameter:
  - **r** - lesen
  - **r+** - lesen und schreiben
  - **w** - schreiben (neu beschreiben)
  - **a** - schreiben (wird hinten angehangen)
  - **x** - neue Datei anlegen

# Lesen und schreiben einer Datei

**print(EineVariable.read())**

- Der print Befehl gibt den Inhalt aus. „.read()‘ ist eine Methode, um auf den Inhalt der Variablen zuzugreifen.
- Ohne „.read()‘ erscheint ledigleich eine Meldung über **Dateiname**, **Modus** und **Zeichencodierung**.

```
<_io.TextIOWrapper name='beispieldatei.txt' mode='r' encoding='cp1252'>
```

**EineVariable.close()**

- Datei ordnungsgemäß schließen.

# Lesen und schreiben einer Datei

## Zeilenweises einlesen mit readline

```
1  if __name__ == "__main__":
2      print("\n\nInhalt der Datei 'beispieldatei.txt':\n<--Anfang-->")
3
4      EineVariable = open('beispieldatei.txt', 'r')
5      print(EineVariable.readline())      # Erste Zeile einlesen
6      print(EineVariable.readline())      # Zweite Zeile einlesen
7      print(EineVariable.readline())      # und so weiter ...
8      print(EineVariable.readline())
9      print(EineVariable.readline())
10     print(EineVariable.readline())
11     print(EineVariable.readline())
12
13     EineVariable.close()
14     print("\n<--Ende-->")
15
```

# Lesen und schreiben einer Datei

**Zeilenweises einlesen mit readline**

- Das **readline** Kommando liest bis zum Ende einer Zeile.
- Der **Dateizeiger** (zu verstehen als Cursor innerhalb einer Datei) bleibt nach dem einlesen an dieser Stelle stehen. Daher fährt das nächste readline-Kommando an dieser Stelle fort.

# Lesen und schreiben einer Datei

## Daten in eine Datei schreiben

```python
# Deklaration der Variable für den Dateinamen + Zuweisung
dateiname = 'daten.txt'

# Eine Eingabe vom Benutzer empfangen.
# Diese Eingabe soll in die Datei 'daten.txt' geschrieben werden.
benutzereingabe = input("Bitte geben Sie einen Text ein: ")

# Erstellen einer neuen Datei 'Daten.txt' im Schreibmodus
with open(dateiname, 'w') as datei:
    # Schreiben der Benutzereingabe in die Datei
    datei.write(benutzereingabe)

print(f"Der Text wurde erfolgreich in die Datei '{dateiname}' geschrieben.")

# Zum Schluss wird die Datei ordentlich geschlossen
datei.close()
```

# Lesen und schreiben einer Datei

**Daten in eine Datei schreiben**

- Datei im Schreibmodus öffnen bzw. erstellen.
- Mittels **write** wird die Eingabe in die Datei geschrieben.

# Weitere Optionen zum öffnen einer Datei

**Option x**

- Die Option **x** im open-Befehl öffnet eine Datei zum exklusiven Erstellen.
- Existiert bereits eine Datei mit diesem Namen, so kommt es zu einem Fehler bzw. einer  Exception.

```
# Dateiname
dateiname = 'testdatei.txt'

try:
    with open(dateiname, 'x') as datei:         # 'dateiname' im exclusiven-Modus öffnen
        datei.write("Testtext.")                # ... und einen Text hineinschreiben ...
    print(f"Die Datei '{dateiname}' wurde erfolgreich erstellt und beschrieben.")
except FileExistsError:
    print(f"Fehler: Die Datei '{dateiname}' existiert bereits. Keine Änderungen vorgenommen.")
```

# Weitere Optionen zum öffnen einer Datei

**Option a**

- Die Option **a** im open-Befehl öffnet eine Datei zum Anhängen von Daten (append).
- Existiert keine Datei mit diesem Namen, so wird eine Neue erstellt und der Inhalt hineingeschrieben.
- Existiert bereits eine Datei mit diesem Namen, wird die Datei geöffnet und die zu schreibenden Daten an das Ende der Datei angehängt.

# Weitere Optionen zum öffnen einer Datei

**Option a**

- Die Option **a** im open-Befehl öffnet eine Datei zum Anhängen von Daten (append).
- Existiert keine Datei mit diesem Namen, so wird eine Neue erstellt und der Inhalt hineingeschrieben.
- Existiert bereits eine Datei mit diesem Namen, wird die Datei geöffnet und die zu schreibenden Daten an das Ende der Datei angehängt.

# Weitere Optionen zum öffnen einer Datei

## Option a

```python
# Dateiname
dateiname = 'testdatei.txt'

# Benutzer nach einem String fragen
benutzereingabe = input("Bitte geben Sie einen Text ein, der an die Datei angehängt werden soll: ")

# Öffnen oder erstellen der Datei im Modus 'attach'
with open(dateiname, 'a') as datei:
    # Schreiben der Benutzereingabe in die Datei
    datei.write(benutzereingabe + '\n')

print(f"Der Text wurde erfolgreich an die Datei '{dateiname}' angehängt.")

```

# Pfadangaben

**Problemstellung:**

- Die Übergabe eines Pfades im gewohnten Schema (Windows) funktioniert nicht!

```
PfadUndName = "C:\Users\ABenutzername\PycharmProjects\Dateihandling_01_lesen_und_schreiben\beispield

EineVariable = open(PfadUndName, 'r')
print(EineVariable.read())

EineVariable.close()
```

- Fehler:

```
  File "C:\Users\AlexanderNiessen\PycharmProjects\Dateihandling_05_Pfadangaben_Beispiel01\main.py", line 7
    PfadUndName = "C:\Users\ABenutzername\PycharmProjects\Dateihandling_01_lesen_und_schreiben\beispieldatei.txt"
                  ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
SyntaxError: (unicode error) 'unicodeescape' codec can't decode bytes in position 2-3: truncated \UXXXXXXXX escape

Process finished with exit code 1
```

# Pfadangaben

**Für dieses Problem existieren drei mögliche Lösungen:**

- **Variante 1:** Die Linux Schreibweise. Es wird direkt der Slash anstelle des Backslash genutzt.

```
# HINWEIS:
#
# Die zu lesende Datei befindet sich nicht im aktuellen Projektverzeichnis:
# C:\Users\Benutzer01\PycharmProjects\Dateihandling_01_lesen_und_schreiben\beispieldatei.txt

# Lösung 1, um Pfade korrekt zu übergeben:

if __name__ == "__main__":
    PfadUndName = "C:/Users/Benutzer01/PycharmProjects/Dateihandling_01_lesen_und_schreiben/beispieldat

    EineVariable = open(PfadUndName, 'r')
    print(EineVariable.read())

    EineVariable.close()

```

# Pfadangaben

**Für dieses Problem existieren drei mögliche Lösungen:**

- **Variante 2:** Der Backslash in der Windowsschreibweise wird durch das Voranstellen eines Backslashes als Sonderzeichen gekennzeichnet.

```
# HINWEIS:
#
# Die zu lesende Datei befindet sich nicht im aktuellen Projektverzeichnis:
# C:\Users\Benutzer01\PycharmProjects\Dateihandling_01_lesen_und_schreiben\beispieldatei.txt

# Lösung 2, um Pfade korrekt zu übergeben:
# -> Der Backslash in der Windowsschreibweise wird durch einen weiteren
#    vorangestellten Backslash als Sonderzeichen maskiert.

if __name__ == "__main__":
    PfadUndName = "C:\\Users\\Benutzer01\\PycharmProjects\\Dateihandling_01_lesen_und_schreiben\\beisp

    EineVariable = open(PfadUndName, 'r')
    print(EineVariable.read())

    EineVariable.close()

```

# Pfadangaben

**Für dieses Problem existieren drei mögliche Lösungen:**

- **Variante 3:** Die Verwendung von „Raw-String“. Dem String wird ein ‚r‘ vorangestellt, welches den Interpreter anweist den String ohne Berücksichtigung von Sonderzeichen zu übernehmen.

```
12  if __name__ == "__main__":
13      PfadUndName = r"C:\Users\Benutzer01\PycharmProjects\Dateihandling_01_lesen_und_schreiben\beispie
14
15      EineVariable = open(PfadUndName, 'r')
16      print(EineVariable.read())
17
18      EineVariable.close()
```

# Weitere Funktionen zum Dateihandling

**Spezifische Funktionen zum Dateihandling:**

- Weitere Methoden zum Dateihandling finden sich im Modul **os** oder **csv**.

**Damit können u.a. folgende Funktionen realisiert werden:**

- Datei löschen
- Verzeichnisse anlegen, umbenennen oder löschen
- CSV-Dateien lesen und schreiben

# Weitere Funktionen zum Dateihandling

**Spezifische Funktionen (aus Modulen, wie z.B. OS) zum Dateihandling**

- Dateien löschen
- Verzeichnisse
- Dateizeiger (Seek)
- CSV-Dateien lesen und schreiben

# Weitere Funktionen zum Dateihandling

## Dateien löschen

```python
import os

# Dateiname:
datei_name = 'beispieldatei.bin'

# Überprüfen, ob die Datei existiert.
# Falls existent: -> löschen
if os.path.exists(datei_name):
    # Datei löschen
    os.remove(datei_name)
    print(f"Datei '{datei_name}' wurde gelöscht.")

# Falls die Datei nicht existiert:
# -> Gebe eine Meldung aus.
else:
    print(f"Datei '{datei_name}' existiert nicht.")
```

# Weitere Funktionen zum Dateihandling

**Dateien löschen**

- Import des Moduls **os**, um die benötigten Methoden zur Verfügung zu stellen.
- Überprüfung, ob die Datei existiert.
- Die Methode ‚remove' löscht die angegebene Datei. Der Dateiname wird als Parameter übergeben.

# Weitere Funktionen zum Dateihandling

**Nützliche Methoden aus dem Modul os:**

- **os.rename(src, dst)**
  - Benennt eine Datei oder ein Verzeichnis von src nach dst um.
- **os.remove(path)**
  - Löscht die Datei
- **path.os.rmdir(path)**
  - Löscht das leere Verzeichnis

# Weitere Funktionen zum Dateihandling

**Nützliche Methoden aus dem Modul os:**

- **path.os.mkdir(path, mode=0o777)**
  - Erstellt ein Verzeichnis path mit den angegebenen Zugriffsrechten.

- **os.listdir(path)**
  - Gibt eine Liste der Dateinamen im Verzeichnis path zurück.

- **os.chdir(path)**
  - Wechselt das aktuelle Arbeitsverzeichnis nach path.

# Weitere Funktionen zum Dateihandling

**Nützliche Methoden aus dem Modul os:**

- **path.os.getcwd()**
  - Gibt das aktuelle Arbeitsverzeichnis zurück.
- **os.path.exists(path)**
  - Überprüft, ob der Pfad path existiert.
- **os.path.isfile(path)**
  - Überprüft, ob der Pfad path eine Datei ist.
- **os.path.isdir(path)**
  - Überprüft, ob der Pfad path ein Verzeichnis ist.

# Weitere Funktionen zum Dateihandling

**Programm zur Demonstration der genannten Methoden:**

- Das aktuelle Arbeitsverzeichnis wird abgefragt
- erstellt ein neues Verzeichnis
- wechselt in dieses Verzeichnis
- erstellt in diesem Verzeichnis eine neue Datei
- benennt die Datei um
- löscht die Datei
- löscht zum Ende das Verzeichnis

# Weitere Funktionen zum Dateihandling

**CSV-Dateien lesen und ausgeben:**

- Das Modul ‚CSV' wird zum Anfang eines Programms über ‚import' eingebunden
- Basisfunktionen zum Umgang mit CSV-Dateien:
  - lesen
  - schreiben
  - anpassungsfähig: delimiter, quotechar

# Weitere Funktionen zum Dateihandling

**Beispielprogramm:** CSV-Dateie einlesen und ausgeben:

- Das folgende Programm liest die Datei „datei.csv“ in ein Array ein und gibt dieses anschließend zeilenweise aus.
- Dazu werden zwei Funktionen benötigt, um das Hauptprogramm schlank zu halten:
  - Funktion 1: **read_csv_to_array(csv_file_path)** (Liest die Datei ein)
  - Funktion 2: **print_2d_array(data)** (Dient dazu die Daten auszugeben)

# Weitere Funktionen zum Dateihandling

```python
import csv
# ------------------------------------------------------------------------
def read_csv_to_array(file_path):
    # Zweidimensionales Array deklarieren
    data = []

    # CSV-Datei öffnen und lesen
    with open(file_path, newline='', encoding='utf-8') as csvfile:
        csvreader = csv.reader(csvfile)

        # Jede Zeile in das zweidimensionale Array einfügen
        for row in csvreader:
            data.append(row)

    return data
# ------------------------------------------------------------------------
```

# Weitere Funktionen zum Dateihandling

```
# ------------------------------------------------------------------------
def print_2d_array(array):
    for row in array:
        print(row)
# ------------------------------------------------------------------------
```

# Weitere Funktionen zum Dateihandling

```
# Hauptprogramm [main.py]
# ======================

# Pfad zur CSV-Datei
csv_file_path = 'datei.csv'

print("Das Programm liest eine CSV-Datei in ein Array ein")
print("und gibt anschließen die Daten des Arrays aus.\n")
print("Ausgabe:")

# CSV-Datei einlesen
data = read_csv_to_array(csv_file_path)

# Zweidimensionales Array ausgeben
print_2d_array(data)
```

# CloudCommand

CloudCommand GmbH chr.schumacher@gmx.tm