Arbeiten mit Dateien

Claas de Boer, Tilman Hinnerichs

28. Januar 2021

Python-Grundlagen

Gliederung

1. Motivation

2. Lesen und Schreiben von Dateien

3. Aufgaben

Was zuletzt geschah

- Module und wo sie zu finden sind
- Pakete (Packages): `import, from, *`
- boilerplate

Motivation

Wofür Dateien?

Python oft in Scripten für Datenverarbeitung:

- Messwerte
- Logs
- Konfigurationen
- ...

↪ liegen in Dateien, sollen in weitere Dateien

Warum Python?

- Einfache Dateiformate (z.B. csv) in Python sehr einfach nutzbar
- Betriebssystemunabhängig (keine Shell-Magie auf Windows)
- Mit Python3: Keine Unicodeprobleme...

Lesen und Schreiben von Dateien

Konzept

Grundlegender Ablauf:

- Öffnen $\rightarrow$ Datei-Objekt
- Lesen / Schreiben $\rightarrow$ Methoden auf Datei-Objekt
- Schließen $\rightarrow$ Ressourcen freigeben

Dateien lesen

```
# Datei öffnen
f = open("test.txt", "r")
# Sämtlichen Inhalt lesen
print (f.read())
# Datei schließen
f.close()
```

Auch nützlich zum Lesen[1]:

```
f = open("test.txt", "r")
# Liest bis zu 100 Zeichen (auch Zeilenumbrüche)
preview = f.read(100)
# Liest alle nachfolgenden! Zeilen in Liste
rest = f.readlines()
# z.B. ["Zeile 1", "Zeile 2", "Zeile 3"]
f.close()
```

↪ jedes lesen schiebt Cursor weiter!

[1] https://www.tutorialspoint.com/python3/python_files_io.htm

Dateien schreiben

```
# Datei öffnen
f = open("test.txt", "w")
f.write("Hello World")
f.close()

```

```
f = open("test.txt", "w")
# Eine Zeile mit "Hello World" und Zeilenumbruch schreiben
f.write("Hello World\n")
# Danach weitere Zeilen anfügen
zeilen = "\n".join(["Zeile 1", "Zeile 2", "Zeile 3"])
f.write(zeilen)
f.close()

```

Dateien richtig lesen

```
f = open("test.txt", "r")
# Liest bis zu 100 Zeichen (auch Zeilenumbrüche)
preview = f.read(100)
# Liest alle nachfolgenden! Zeilen in Liste
rest = f.readlines()
# z.B. ["Zeile 1", "Zeile 2", "Zeile 3"]
f.close()
```

Dateien richtig lesen

```
f = open("test.txt", "r")
# Liest bis zu 100 Zeichen (auch Zeilenumbrüche)
preview = f.read(100)
# Liest alle nachfolgenden! Zeilen in Liste
rest = f.readlines()
# z.B. ["Zeile 1", "Zeile 2", "Zeile 3"]
f.close()
```

```
with open('test.txt', 'r') as f:
    for line in f:
        # trennt nach jedem '\n'
        # "Hallo Mama, Ich bin in einer Textdatei!\n"
        greetings, message = line.strip().split(',')
        ...
```

Dateien richtig schreiben

```
f = open("test.txt", "w")
# Eine Zeile mit "Hello World" und Zeilenumbruch schreiben
f.write("Hello World\n")
# Danach weitere Zeilen anfügen
zeilen = "\n".join(["Zeile 1", "Zeile 2", "Zeile 3"])
f.write(zeilen)
f.close()
```

Dateien richtig schreiben

```
f = open("test.txt", "w")
# Eine Zeile mit "Hello World" und Zeilenumbruch schreiben
f.write("Hello World\n")
# Danach weitere Zeilen anfügen
zeilen = "\n".join(["Zeile 1", "Zeile 2", "Zeile 3"])
f.write(zeilen)
f.close()
```

```
with open(file='test.txt', 'w') as f:
    # entweder
    f.write("Hallo Kind, toll gemacht!\n")
    # oder
    print("Hallo Kind, toll gemacht!\n", file=f)
```

Was man sonst noch mit Dateien machen kann

- r - Read - Default value. Opens a file for reading, error if the file does not exist
- a - Append - Opens a file for appending, creates the file if it does not exist
- w - Write - Opens a file for writing, creates the file if it does not exist
- x - Create - Creates the specified file, returns an error if the file exists

Zusätzlich:

- t - Text - Default value. Text mode
- b - Binary - Binary mode for e.g. images, pickled (See pickle package) files

Ein Beispiel

```
with open("demofile.txt", "rt") as f:
    # der standardfall, muss nicht explizit genannt werden
    ...
```

```
...
import pickle

spieler = Spieler(wuerfel, name)
spieler_list.append(spieler)

with open(file='spielerlist.pkl', mode='wb') as f:
    pickle.dump(spieler_liste, f)
```

Aufgaben

Einige Aufgaben

1. Downloaded https://loremipsum.de/downloads/version5.txt von `loremipsum.de`
2. Zähle die Wörter in diesem File
3. Kopiere den Inhalt in eine andere Datei
4. Lösche alle **new line character** in diesem neuen file.

Encodingszähmen leicht gemacht

- Character und Zahlen werden intern durch Bitsequenzen repräsentiert

Encodingszähmen leicht gemacht

- Character und Zahlen werden intern durch Bitsequenzen repräsentiert
- Gegeben eine Bitsequenz, wie soll diese interpretiert werden?

Encodingszähmen leicht gemacht

- Character und Zahlen werden intern durch Bitsequenzen repräsentiert
- Gegeben eine Bitsequenz, wie soll diese interpretiert werden?
- Header, Dateiendungen, Anderes Wissen
- `UTF-8, ASCII, UTF-16, ...`
- `Hex, Binary, Decimal`

```
>>> int('11', base=16)
17
>>> int('11', base=2)
3
>>> hex(17)
'0x11'
>>> bin(11)
'0b11'
```

```
with open(file='test', mode='r', encoding='utf8'):
    ...
```

ord und chr

`chr(i)`

Return the string representing a character whose Unicode code point is the integer i. For example, chr(97) returns the string 'a', while chr(8364) returns the string '€'. This is the inverse of ord().

```
>>> ord('a')
97
>>> chr(97)
'a'
>>> ord('A')
65
```

Einige Aufgaben

1. Schreibe ein Programm, welches eine Caesar Chiffre beschreibt. (`caesar_chiffre(text, offset)`)
2. Schreibe eine Liste aller geraden Quadratzahlen bis 1000 in eine Datei und lese diese Liste von der Datei wieder ein
3. Lese die Dokumentation von `pickle`, und schreibe ein Python Programm, welches ein Spielerobject abspeichert und lädt.