File I/O

In Python ist es natürlich möglich, Dateien einzulesen und Daten in Dateien abzuspeichern. Wir beschäftigen uns nur mit reinen Textdateien. Die wichtigsten Dateiformate für uns sind `.txt` und `.csv` . csv steht für comma-separated-value, die Dateien sind vom Prinzip her wie eine Tabelle, wobei die Spalten durch Kommata voneinander getrennt werden.

Dateien lesen

Dateien können mit `open(filename)` geöffnet werden. Wichtig ist, dass Dateien auch wieder geschlossen werden. Spätestens am Ende des Programm macht Python das automatisch, man kann Dateien vorher aber auch mit `.close()` schließen. Mit `.read()` oder `.readline()` kann Text aus der Datei eingelesen werden. Der Datentyp ist dabei immer `String` .

In [1]:

```python
# Beispiel 1

f = open("Der_kleine_Hobbit_Leseprobe.txt", encoding='utf8')
txt = f.readline()  # 1. Zeile
print(txt)
print('---')
txt = f.readline()  # 2. Zeile
print(txt)
print('---')
txt = f.readline()  # 3. Zeile
print(txt)
print('---')
txt = f.read()  # Rest vom Text
print(txt)
print('---')
txt = f.read()  # Jetzt kann nichts mehr eingelesen werden
print(txt)
```

```
Eine unvorhergesehene Gesellschaft

---


---
In einer Höhle in der Erde, da lebte ein Hobbit. Nicht in einem schmutzigen,
nassen Loch, in das die Enden von irgendwelchen Würmern herabbaumelten und d
as nach Schlamm und Moder roch. Auch nicht etwa in einer trockenen Kieshöhl
e, die so kahl war, dass man sich nicht einmal niedersetzen oder gemütlich f
rühstücken konnte. Es war eine Hobbithöhle, und das bedeutet Behaglichkeit.

---
Diese Höhle hatte eine kreisrunde Tür wie ein Bullauge. Sie war grün gestric
hen und in der Mitte saß ein glänzend gelber Messingknopf. Die Tür führte zu
einer röhrenförmig langen Halle, zu einer Art Tunnel, einem Tunnel mit getäf
elten Wänden. Der Boden war mit Fliesen und Teppichen ausgelegt, es gab Stüh
le da von feinster Politur und an den Wänden Haken in Massen für Hüte und Mä
ntel, denn der Hobbit hatte Besucher sehr gern. Der Tunnel wand und wand sic
h, führte aber nicht tief ins Innere des Berges hinein, den alle Leute viele
Meilen weit rund im Lande schlechthin »den Berg« nannten. Zahlreiche kleine,
runde Türen öffneten sich zu diesem Tunnel, zunächst auf der einen Seite und
dann auch auf der anderen. Treppen zu steigen brauchte der Hobbit nicht: Sch
lafräume, Badezimmer, Keller, Speisekammern (eine Masse von Speisekammern),
Kleiderschränke (ganze Räume standen ausschließlich für die Unterbringung se
iner Garderobe zur Verfügung), Küchen, Esszimmer – alles lag an demselben la
ngen Korridor. Die besten Zimmer lagen übrigens auf der linken Seite (wenn m
an hineinkommt), denn ausschließlich diese hatten Fenster, tief gesetzte, ru
nde Fenster, die hinaus auf den Garten blickten und über die Wiesen, die sic
h gemächlich hinab bis zum Fluss neigten.
Dieser Hobbit war ein sehr wohlhabender Hobbit und sein Name war Beutlin. Di
e Beutlins hatten seit undenklichen Zeiten in der Nachbarschaft des »Berges«
gelebt und die Leute hielten sie für außerordentlich achtbar – nicht nur wei
l die meisten der Beutlins reich, sondern weil sie noch nie in ein Abenteuer
verstrickt gewesen waren und nie etwas Unvorhergesehenes getan hatten. Man k
onnte im Voraus sagen, was ein Beutlin auf eine Frage antworten würde, ohne
dass man sich die Mühe machen musste, diese Frage wirklich zu stellen. Dies
hier aber ist eine Geschichte von einem Beutlin, der trotzdem Abenteuer erle
bte und sich selbst über völlig unvorhergesehene Fragen reden hörte. Viellei
cht verlor er bei seinen Nachbarn an Ansehen, aber er gewann – nun, ihr werd
```

```
et ja sehen, ob er am Ende überhaupt etwas gewann.
Die Mutter unseres Hobbits – was ist eigentlich ein Hobbit? Ich glaube, dass
die Hobbits heutzutage einer Beschreibung bedürfen, da sie selten geworden s
ind und scheu vor den »Großen Leuten«, wie sie uns zu nennen pflegen. Sie si
nd (oder waren) ungefähr halb so groß wie wir und kleiner als die bärtigen Z
werge (sie tragen jedoch keine Bärte). Es ist wenig, sozusagen gar nichts vo
n Zauberei an ihnen, ausgenommen die alltägliche Gabe, rasch und lautlos zu
verschwinden, wenn großes dummes Volk wie du und ich angetapst kommt und Rad
au macht wie Elefanten, was sie übrigens eine Meile weit hören können. Sie n
eigen dazu, ein bisschen fett in der Magengegend zu werden. Sie kleiden sich
in leuchtende Farben (hauptsächlich in Grün und Gelb). Schuhe kennen sie übe
rhaupt nicht, denn an ihren Füßen wachsen natürliche, lederartige Sohlen und
dickes, warmes, braunes Haar, ganz ähnlich wie das Zeug auf ihrem Kopf (das
übrigens kraus ist). Die Hobbits haben lange, geschickte, braune Finger, gut
mütige Gesichter und sie lachen ein tiefes, saftiges Lachen (besonders nach
den Mahlzeiten; Mittagessen halten sie zweimal am Tag, wenn sie es bekommen
können). Nun, das sei vorerst genug und wir wollen fortfahren.
Bilbo Beutlin hieß unser Hobbit und seine Mutter war die berühmte Belladonna
Tuk, eine der drei ausgezeichneten Töchter des alten Tuk. Der alte Tuk war d
as Haupt der Hobbits, die jenseits des »Wassers« wohnten, des schmalen Fluss
es am Fuß des Berges. Es wurde oft gemunkelt, dass vor langer Zeit einmal ei
n Tuk eine Fee geheiratet habe. Das war natürlich Unsinn. Aber sicherlich wa
r bei ihnen nicht alles hobbitmäßig. Denn ab und zu ging ein Angehöriger der
Tuks fort und stürzte sich in Abenteuer. Sie verschwanden heimlich und die F
amilie vertuschte es. Tatsache ist jedenfalls, dass die Tuks nicht ganz so g
eachtet waren wie die Beutlins, obgleich sie unzweifelhaft reicher waren.
Nicht, dass Belladonna Tuk jemals in irgendwelche Abenteuer verwickelt gewes
en wäre, nachdem sie die Frau von Mister Bungo Beutlin geworden war. Bungo,
Bilbos Vater, baute (teilweise mit ihrem Geld) für sie die kostspieligste Ho
bbithöhle, die jemals unterhalb oder oberhalb des Berges oder jenseits des W
assers gebaut worden war. Und dort lebten sie bis an das Ende ihrer Tage. In
dessen ist es wahrscheinlich, dass Bilbo, ihr einziger Sohn, obgleich er doc
h aussah und sich genauso benahm wie eine zweite Ausgabe seines grundsoliden
und behäbigen Vaters, irgendetwas Wunderliches in seinen Anlagen von der Tuk
seite übernommen hatte. Es war etwas, das nur auf die Chance wartete, um ans
Licht zu kommen. Die Chance ergab sich erst, als Bilbo Beutlin etwa fünfzig
Jahre alt geworden war, in der wunderschönen Hobbithöhle wohnte, die sein Va
ter erbaut hatte, und sich augenscheinlich zur Ruhe gesetzt hatte.

Quelle: https://www.dtv.de/_files_media/title_pdf/leseprobe-7151.pdf (http
s://www.dtv.de/_files_media/title_pdf/leseprobe-7151.pdf)

---
```

Beim Einlesen von Dateien verwendet Python quasi einen Cursor, der sich die Position innerhalb der Datei merkt. Beim Öffnen einer Datei befindet sich dieser Cursor ganz am Anfang, deshalb können wir direkt anfangen, Text einzulesen. `readline()` liest vom Cursor bis zum nächsten Zeilenumbruch, `read()` liest vom Cursor bis zum Ende der Datei. Aufgrund des Cursors kann in Beispiel 1 der zweite Aufruf von `f.read()` keinen Text mehr einlesen, da sich der Cursor bereits am Ende der Datei befindet. Mit `seek()` können wir die Position des Cursors verwenden. `seek(0)` setzt den Cursor wieder an den Anfang der Datei. Die Zahl zählt dabei die Zeichen. Wenn wir nur eine bestimmte Menge Text einlesen wollen, können wir `read()` auch mit einer Zahl als Argument aufrufen.

In [16]:

```
# Beispiel 2

f.seek(0)
txt = f.readline()
print(txt)
print('---')
f.seek(0)
txt = f.read(7)
print(txt)
```

```
Eine unvorhergesehene Gesellschaft

---
Eine un
```

Bei der Anzahl der Zeichen werden Escape-Sequenzen mitgezählt! Das heißt, ein Zeilenumbruch zählt als zwei Zeichen, weil die Escape-Sequenz für einen Zeilenumbruch `\n` ist.

In [17]:

```
# Beispiel 3

f.seek(0)
txt = f.readline()
print(repr(txt))
```

```
'Eine unvorhergesehene Gesellschaft\n'
```

In Dateien schreiben

File-Objekte haben auch eine `.write()` Funktion, mit der wir Text in eine Datei schreiben können. Probieren wir das mal aus:

In [19]:

```
# Beispiel 4

f.write("neuer Text")

f.close()
```

```
---------------------------------------------------------------------------
UnsupportedOperation                      Traceback (most recent call last)
<ipython-input-19-a8db89c7f052> in <module>
      1 # Beispiel 4
      2
----> 3 f.write("neuer Text")
      4
      5 f.close()

UnsupportedOperation: not writable
```

Als Fehlermeldung bekommen wir `not writable`. Wenn wir eine Datei öffnen, wird diese in einem bestimmten Modus geöffnet. Standardmäßig ist dies der Lesemodus ohne Schreibrechte. Diese Modi gibt es:

Zeichen	Modus
'r'	Read mode (standard)
'w'	Write mode (überschreibt den vorherigen Dateiinhalt)
'a'	Write mode, *append*: setzt die aktuelle Position ans Dateiende
'x'	Write mode, *exclusive creation*: klappt nur, wenn die Datei noch nicht existiert
'b'	Binary mode (+ other)
'+'	Read/Write mode (+ other)

Wenn wir eine Datei mit `w` öffnen, können wir also in diese Datei schreiben. Dabei wird allerdings der alte Inhalt der Datei gelöscht. Aber wenn eine Datei noch nicht existiert, wird sie automatisch erstellt.

In [23]:

```
# Beispiel 5

f = open('bsp_5.txt', 'w')
f.write("Hallo Welt!")

txt = f.read()
print(txt)
```

```
---------------------------------------------------------------------------
UnsupportedOperation                      Traceback (most recent call last)
<ipython-input-23-987ab9ba719d> in <module>
      4 f.write("Hallo Welt!")
      5
----> 6 txt = f.read()
      7 print(txt)

UnsupportedOperation: not readable
```

Jetzt können wir zwar in die Datei schreiben, aber nicht mehr aus der Datei lesen. Um beides zu ermöglichen, muss ein `+` an den Modus angehängt werden. Außerdem müssen wir daran denken, den Cursor wieder an den Anfang zu setzen, wenn wir lesen wollen. Denn auch das Schreiben in eine Datei verschiebt den Cursor entsprechend.

In [47]:

```
# Beispiel 6

f = open('bsp_5.txt', 'w+')
f.write("Hallo Welt!")

f.seek(0)
txt = f.read()
print(txt)
f.close()
```

```
Hallo Welt!
```

Buffer

Wenn wir mit `.write()` in eine Datei schreiben, und die Datei dann vom Dateiexplorer (Windows-Explorer) aus öffnen, kann es sein, dass unser Text noch nicht in der Datei steht. Um die Festplatten zu schonen, wird der Text zunächst in einen Zwischenspeicher geschrieben. Dieser Speicher wird regelmäßg (bei einer bestimmten Datenmenge) in die Datei geschrieben. Das Schreiben aus dem Zwischenspeicher in die Datei kann aber auch von uns veranlasst werden mit der `.flush()` Funktion. Beim Schließen der Datei wird auch noch einmal der Zwischenspeicher in die Datei geschrieben.

`print()`

Wenn wir einen Blick in die Dokumentation der `print()` Funktion werfen, sehen wir einen `file` -Parameter:

In [48]:

```
help(print)
```

```
Help on built-in function print in module builtins:

print(...)
    print(value, ..., sep=' ', end='\n', file=sys.stdout, flush=False)

    Prints the values to a stream, or to sys.stdout by default.
    Optional keyword arguments:
    file:  a file-like object (stream); defaults to the current sys.stdout.
    sep:   string inserted between values, default a space.
    end:   string appended after the last value, default a newline.
    flush: whether to forcibly flush the stream.
```

Dieser ist standardmäßig auf `sys.stdout` gesetzt, also die Kommandozeile. Es kann aber auch ein File-Objekt übergeben werden. Das ist also eine zweite Möglichkeit, Daten in eine Datei zu schreiben.

Iterable

Eine Datei kann auch in einer `for` -Schleife verwendet werden. Dabei werden nacheinander die Zeilen ausgelesen:

In [3]:

```
# Beispiel 7

f = open("Der_kleine_Hobbit_Leseprobe.txt", encoding='utf8')

for line in f:
    print(repr(line[:80]))  # gib nur die ersten 80 Zeichen jeder Zeile aus

f.close()
```

```
'Eine unvorhergesehene Gesellschaft\n'
'\n'
'In einer Höhle in der Erde, da lebte ein Hobbit. Nicht in einem schmutzige
n, nas'
'Diese Höhle hatte eine kreisrunde Tür wie ein Bullauge. Sie war grün gestri
chen '
'Dieser Hobbit war ein sehr wohlhabender Hobbit und sein Name war Beutlin. D
ie Be'
'Die Mutter unseres Hobbits – was ist eigentlich ein Hobbit? Ich glaube, das
s die'
'Bilbo Beutlin hieß unser Hobbit und seine Mutter war die berühmte Belladonn
a Tuk'
'Nicht, dass Belladonna Tuk jemals in irgendwelche Abenteuer verwickelt gewe
sen w'
'\n'
'Quelle: https://www.dtv.de/_files_media/title_pdf/leseprobe-7151.pdf\n' (ht
tps://www.dtv.de/_files_media/title_pdf/leseprobe-7151.pdf\n')
```

with

Mit dem `with` Statement können wir uns das Schließen der Datei sparen.

In [57]:

```
# Beispiel 8

with open("Der_kleine_Hobbit_Leseprobe.txt") as f:
    print(f.readline())

print(f.closed)
```

```
Eine unvorhergesehene Gesellschaft

True
```

Mit dem `with` Statement sagen wir, wir wollen die mit `open()` angegebene Datei als `f` verwenden. Dies gilt aber nur für den `with` -Code-Block, also den eingerückten Code. Am Ende dieses Code-Blocks wird die Datei automatisch geschlossen. Mit dem `.closed` Attribut können wir prüfen, dass die Datei tatsächlich geschlossen wurde.

Was alles schief gehen kann

Beim Arbeiten mit Dateien können diverse Fehler auftreten, zum Beispiel:

- Eine Datei, die geöffnet werden soll, existiert nicht.
- Eine Datei ist bereits in einem anderem Programm geöffnet.
- Ein anderes Programm modifiziert eine Datei während wir sie in Bearbeitung haben.
- Ein Benutzer hat nicht die Rechte, eine Datei zu öffnen oder zu ändern.
- Ein Laufwerk oder Ordner existiert nicht.
- Ein externes Laufwerk wird während des Bearbeitens abgezogen.
- Ein Netzlaufwerk ist wegen einer unterbrochenene Netzwerkverbindung nicht mehr verfügbar.
- Es ist nicht genügend Speicherplatz vorhanden, um die gewünschten Daten zu Schreiben.
- ...

Mit `try-except` haben wir eine Möglichkeit, diese Fehler abzufangen und entsprechend zu behandeln. Vor allem das Öffnen und Schreiben einer Datei sollte sichergestellt werden, damit die Datei auch im Fehlerfalls korrekt geschlossen werden kann.

`.split()` und `.join()`

Mit `.split()` können wir einen String an einem bestimmten Zeichen trennen. Das Ergebnis ist eine Liste, die die einzelnen Teile des Strings enthält.

In [58]:

```
# Beispiel 9

s = "Hier steht ein Satz mit sieben Wörtern."
l = s.split(" ")  # Leerzeichen als Trenner
print(l)
```

```
['Hier', 'steht', 'ein', 'Satz', 'mit', 'sieben', 'Wörtern.']
```

In diesem Beispiel wurde als Trennzeichen das Leerzeichen verwendet. Das heißt, bei jedem Leerzeichen fängt ein neues Element an. Der Trenner selbst, also das Leerzeichen, wird dabei gelöscht.

Mit `.join` kann diese Liste wieder zu einem String zusammengesetzt werden. `.join()` wird dabei auf dem String aufgerufen, der als Bindeglied dienen soll.

In [59]:

```
# Beispiel 10

s_neu = '--'.join(l)
print(s_neu)
```

```
Hier--steht--ein--Satz--mit--sieben--Wörtern.
```

Übung

1. Erstelle eine neue Datei `zahlen.csv` und schreibe in diese Datei 100 zufällige ganze Zahlen zwischen 1 und 99. Dabei sollen immer zehn Zahlen in einer Zeile stehen und die Zahlen mit Kommata getrennt werden. Die Escape-Sequenz für einen Zeilenumbruch ist `\n` und kann direkt in einem String verwendet werden. Die Datei sollte also ungefähr so aussehen:

   ```
   5,53,93,30,3,64,46,67,27,46
   51,30,94,98,3,63,85,79,88,55
   60,11,92,...
   ```

2. Lies deine Datei ein und berechne für jede Zeile die Summe der Zahlen. Verwende dafür wirklich die Daten, die du aus der Datei eingelesen hast.

Tipp: Die Zufallszahlen kannst du mit `random.randint(1,99)` generieren. Wenn du damit Probleme hast, fange erstmal damit an, irgendetwas in deine Datei zu schreiben, zum Beispiel die Zahlen von 1 bis 10 oder einen beliebigen String.
Wenn du deine Zufallszahlen zusätzlich in einer (oder mehreren) Liste(n) speicherst, kannst du dein Ergebnis kontrollieren. Mit `sum(<Liste>)` kannst du die Summe einer Liste berechnen.

In [ ]:

```
import random

# dein Code hier
```