Mehr zu print (1)

- Wahrscheinlich haben Sie sich schon gefragt, warum manche Ausgaben mit print so kompliziert aussehen, z.B.

```
print("Ist " + s + " ein Palindrom? " + str(is_palindrom(s)))
```

- Nachdem wir Schlüsselwortparameter kennen, können wir die Funktion print genauer betrachten:

```
print(value1, ..., sep=' ', end='\n', file=sys.stdout, flush=False)
```

- Die Auslassungspunkte bedeuten, dass print beliebig viele, durch Kommata getrennte Argumente haben kann, die der Reihe nach auf die Konsole ausgegeben werden. sep ist die Trennzeichenkette zwischen den Ausgaben. Als Default-Wert ist ein Leerzeichen eingestellt. Beispiel:

```
s = "Anna"
print("Ist " + s + " ein Palindrom? " + str(is_palindrom(s)))      # Ausgabe: Ist Anna ein Palindrom? True
print("Ist", s, "ein Palindrom?", is_palindrom(s))                 # Ausgabe: Ist Anna ein Palindrom? True
```

Mehr zu print (2)

- Man spart sich sogar die Umwandlung des booleschen Wertes in eine Zeichenkette, da nicht mehr konkateniert wird.
- end wird nach der eigentlichen Ausgabe angehängt. Der Default ist \n, d.h. der Beginn einer neuen Zeile. Beispiel:

```
s = "Anna"
print("Ist", s, "ein Palindrom?", is_palindrom(s), sep="_", end="!")      # Ausgabe: Ist_Anna_ein Palindrom?_True!
```

- Wenn man end="" setzt, kann man mehrere print-ausgaben auch in einer Zeile haben.
- Standardmäßig wird auf die Konsole ausgegeben. Deshalb der Default-Parameter file=sys.stdout.
- Die Funktion print kann auch für die Ausgabe in Textdateien verwendet werden. Beispiel:

```
s = "Anna"
fh = open("ergebnis.txt", "w")
print("Ist", s, "ein Palindrom?", is_palindrom(s), file=fh)
fh.close()
```

Mehr zu print (3)

- Allerdings muss zuvor mit der Funktion open eine Datei geöffnet werden. Wegen der Angabe von "w" wird die Datei neu erstellt. Vorsicht: Falls die Datei bereits besteht, wird sie überschrieben. Die Datei wird im selben Verzeichnis wie das Skript angelegt. Oder man gibt einen absoluten oder relativen Pfad an. Beispiele:

  ```
  fh = open("c:/users/gold/documents/ergebnis.txt", "w")
  fh = open("../documents/ergebnis.txt", "w")
  ```

  Beachten Sie dabei die Schrägstriche. Nachdem die Dateiausgabe beendet ist, muss die Datei mit der Funktion close geschlossen werden.

- Der Parameter flush gibt an, ob die Ausgabe sofort in die Datei geschrieben wird (flush=True) oder erst beim Schließen (flush=False). Der Unterschied macht sich nur bemerkbar, wenn mehrere Ausgaben in selben Programm an verschiedenen Stellen vorhanden sind.

Mehr zu print (4)

- Natürlich kann man Dateien auch einlesen. Beispiel:

```
fh = open("zahlen.txt", "w")
print(1, 2, 3, 4, sep="\n", file=fh)
fh.close()

fh = open("zahlen.txt", "r")
summe = 0
for line in fh:
    summe = summe + int(line)
fh.close()

print(summe)
```

In die Datei zahlen.txt werden zeilenweise die Zahlen 1, 2, 3, 4 geschrieben. Danach werden in der Schleife die Zeilen der Reihe nach gelesen und die Zahlen aufaddiert.

Mehr zu print (5)

- Bei der Verwendung von Dateien können Fehler auftreten, z.B. kann es passieren, dass eine Datei nicht geöffnet werden kann, weil der Dateiname falsch geschrieben wurde, oder es kann vergessen werden, die Datei nach der Verwendung zu schließen.
- Um Fehler abzufangen, sollte man deshalb die with-Anweisung benutzen. Das Schließen der Dateien wird dabei von der with-Anweisung übernommen. Beispiel:

```
with open("zahlen.txt", "w") as fhi:
    print(1, 2, 3, 4, sep="\n", file=fhi)

with open("zahlen.txt", "r") as fho:
    summe = 0
    for line in fho:
        summe = summe + int(line)
print(summe)
```

Mehr zu print (6)

- Bei der Ausgabe mit print möchte man oft ein bestimmtes Format einhalten (Ausrichtung, Breite, Anzahl der Ziffern, etc.).
- Eine Möglichkeit mit der Funktion format haben wir schon gesehen. Beispiel:

```
print("Preis: {0:6.2f} Euro".format(8.15))     # Ausgabe: Preis:   8.15 Euro
```

- Es gibt noch weitere Funktionen zur Formatierung: center, ljust, rjust, zfill. Beispiel:

```
preis = "8.15"
print("Preis: ", preis.rjust(5, " "), "Euro")     # Ausgabe: Preis:   8.15 Euro
print("Preis: ", preis.zfill(5), "Euro")          # Ausgabe: Preis:  08.15 Euro
```

- 🕮 Kapitel 11.5, 12.1 – 12.2, 12.4, 12.8 in (Klein 2018)

Mehr zu print (7)

Aufgaben

1. Die Ergebnisse einer fiktiven Prüfung sind in der Datei punkte.csv enthalten. Ausschnitt aus dieser Datei:

```
90024;4.0;8.0;4.0;3.0;5.5;9.5;8.5;9.0
90025;6.5;8.5;10.5;5.5;16.0;10.5;10.0;9.5
90026;4.5;7.0;9.0;5.0;8.0;11.0;7.0;9.0
90027;4.0;6.5;1.5;2.5;2.5;5.0;5.5;4.5
```

   Jede Zeile ist ein Datensatz. Die Datei enthält 37 Datensätze. Die einzelnen Daten jedes Datensatzes sind durch Strichpunkte getrennt. Jeder Datensatz beginnt mit einer Matrikelnummer. Die Matrikelnummern sind von 90000 bis 90036 durchnummeriert. (Falls Sie zufällig eine dieser Matrikelnummern haben, hat das nichts zu bedeuten.) Danach kommen in jedem Datensatz acht Zahlen, die die erreichten Punkte in den acht Aufgaben der Prüfung angeben.

   Die Aufgabe soll ohne Verwendung von Python-csv-Funktionen gelöst werden.

   Schreiben Sie ein Skript, das … (weiter auf der nächsten Folie)

Mehr zu print (8)

Aufgaben

- die Datei punkte.csv zeilenweise einliest,
- jede eingelesene Zeile in ihre neun Teile splittet,
- dabei die Summe der erreichten Punkte und die Note berechnet,
- die Matrikelnummer, die Punkte für die Aufgaben, die Summe der Punkte und die Note in eine Datei noten.txt ausgibt.

Sie können die folgende Schachtelung der with-Anweisungen verwenden, um gleichzeitig aus punkte.csv zu lesen und in die Ausgabedatei zu schreiben:

```
with open("punkte.csv", "r") as fhi:
    with open("noten.txt", "w") as fho:
        ...
```

Mehr zu print (9)

Aufgaben

Zur Berechnung der Noten können Sie sich einen beliebigen Notenschlüssel ausdenken. Es können maximal 80 Punkte in dieser Prüfung erreicht werden. Die Ausgabe soll eine Kopfzeile haben und folgendermaßen formatiert werden. Ausschnitt aus der Datei noten.txt:

```
Matrikel   A1   A2   A3   A4   A5   A6   A7   A8 Punkte Note
   90024  4.0  8.0  4.0  3.0  5.5  9.5  8.5  9.0   51.5  2.3
   90025  6.5  8.5 10.5  5.5 16.0 10.5 10.0  9.5   77.0  1.0
   90026  4.5  7.0  9.0  5.0  8.0 11.0  7.0  9.0   60.5  1.7
   90027  4.0  6.5  1.5  2.5  2.5  5.0  5.5  4.5   32.0  5.0
```