1.5 Ein- und Ausgabe

Bisher können unsere Programme

- Eingaben von der Kommandozeile lesen (`sys.argv`),
- Eingaben von der Tastatur lesen (`input()`), und
- Ausgaben auf den Bildschirm schreiben (`print()`).

Wir werden in dieser Vorlesung ein allgemeineres Konzept zum

- Lesen von Eingaben aus Dateien
- Schreiben von Ausgaben in Dateien

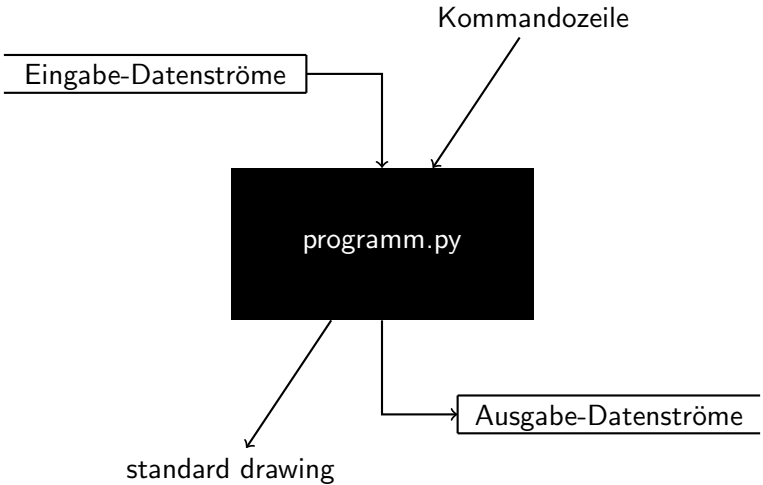
kennenlernen. „Lesen von der Tastatur“ und „Schreiben auf den Bildschirm“ sind zwei Spezialfälle davon.
Außerdem lernen wir noch etwas über „Schönschrift“.

Damit können wir Programme schreiben, die große Mengen fremder Daten verarbeiten und die Ergebnis-Daten unserer Programme anderen zugänglich zu machen.

Wir werden uns nur um das Lesen und Schreiben von Strings kümmern.

In der nächsten Vorlesung werden wir auch lernen, Graphiken auszugeben.

Schematische Darstellung



Vorlesung 05

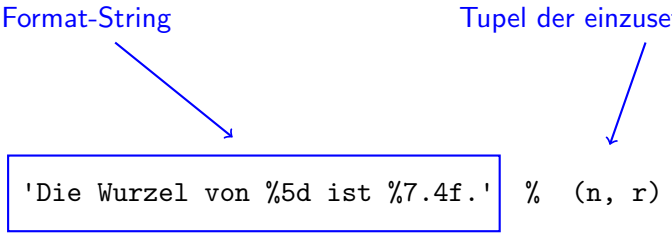
1 Formatierte „schöne“ Strings mit der %-Operation

```
'Die Wurzel von %5d ist %7.4f.'  %  (n, r)
```

ergibt mit `n=1000` und `r=31.62276` den String

```
'Die Wurzel von  1000 ist 31.6228.'
```

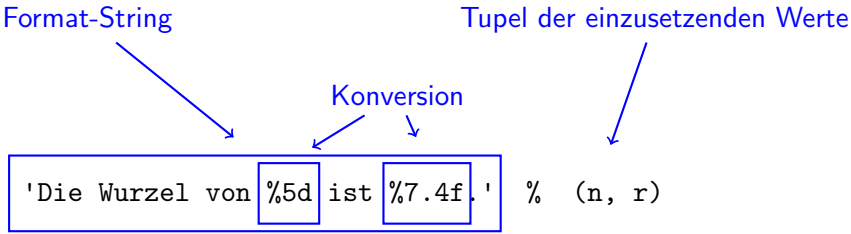
1 Formatierte „schöne“ Strings mit der %-Operation



ergibt mit `n=1000` und `r=31.62276` den String

```
'Die Wurzel von  1000 ist 31.6228.'
```

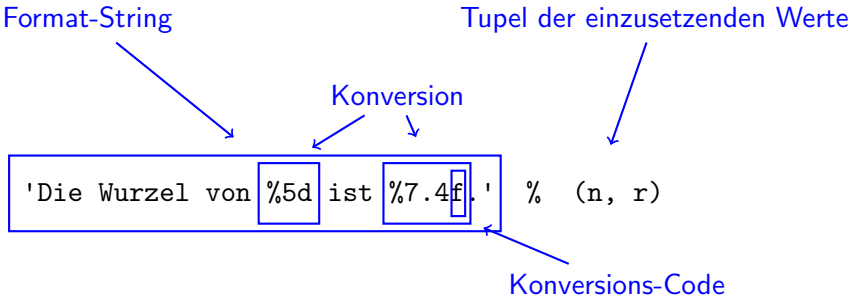
1 Formatierte „schöne“ Strings mit der %-Operation



ergibt mit `n=1000` und `r=31.62276` den String

```
'Die Wurzel von  1000 ist 31.6228.'
```

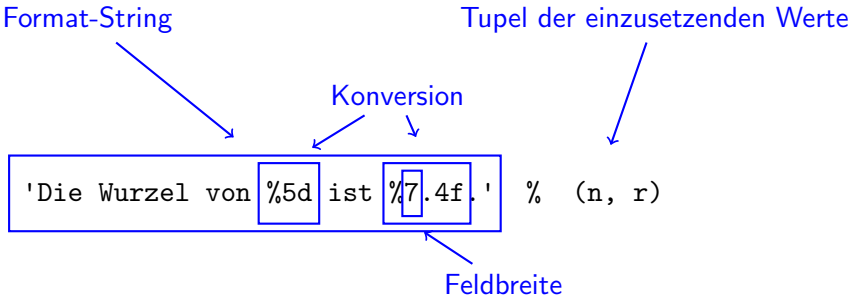
1 Formatierte „schöne“ Strings mit der %-Operation



ergibt mit `n=1000` und `r=31.62276` den String

```
'Die Wurzel von  1000 ist 31.6228.'
```

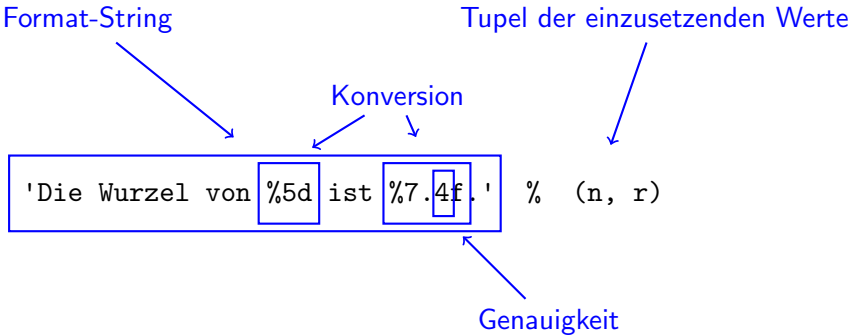
1 Formatierte „schöne“ Strings mit der %-Operation



ergibt mit `n=1000` und `r=31.62276` den String

```
'Die Wurzel von  1000 ist 31.6228.'
```

1 Formatierte „schöne“ Strings mit der %-Operation



ergibt mit `n=1000` und `r=31.62276` den String

```
'Die Wurzel von  1000 ist 31.6228.'
```

Typ	Konversions-Code	Format-String		Ergebnis
int	d	'%14d' % (1984,)	ergibt	' 1984'
		'%-14d' % 1984	ergibt	'1984 '
float	f	'%14.2f' % 1234.12345678	ergibt	' 1234.12'
		'%.6f' % 1234.12345678	ergibt	'1234.123457'
	e	'%14.4e' % 1234.12345678	ergibt	' 1.2341e+03'
string	s	'%14s' % 'Hello, World'	ergibt	' Hello, World'
		'%-14s' % 'Hello, World'	ergibt	'Hello, World '
		'%-14.5s' % 'Hello, World'	ergibt	'Hello '

Statt eines Tupels aus einem Element kann man auch das Element ohne Klammern und Komma schreiben.

```
#-------------------------------------------------------------------------------
# formatierteAusgabe.py
#-------------------------------------------------------------------------------
# Beispiele für die formatierte Ausgabe von int- und float-Werten.
#-------------------------------------------------------------------------------
import math

n = 1000
r = math.sqrt(n)

# Unformatierte Ausgabe:
# Gib int n und float r aus.
print('(1) Die Wurzel von ' + str(n) + ' ist ' + str(r) + '.')          # (1) Die Wurzel von 1000 ist 31.622776601683793.

# Formatierte Ausgaben:
# Gib int n und float r aus. Bei float werden standardmäßig 6 Nachkommastellen ausgegeben.
print( '(2) Die Wurzel von %d ist %f.' % (n, r) )                       # (2) Die Wurzel von 1000 ist 31.622777.

# Gib int n mit 5 Zeichen und float r mit 3 Nachkommastellen aus
# und füge einen zusätzlichen Zeilenumbruch am Zeilenende ein.
print( '(3) Die Wurzel von %5d ist %.3f.' % (n, r) )                    # (3) Die Wurzel von  1000 ist 31.623.

# Wenn Zeilenumbrüche wie im Format-String gemacht werden sollen,
# dann muss man ihn in ''' einklammern.
print( '''(4) Die Wurzel von                                            # (4) Die Wurzel von
%5d                                                                     #  1000
ist %.3f.''' % (n, r) )                                                 # ist 31.623.
```

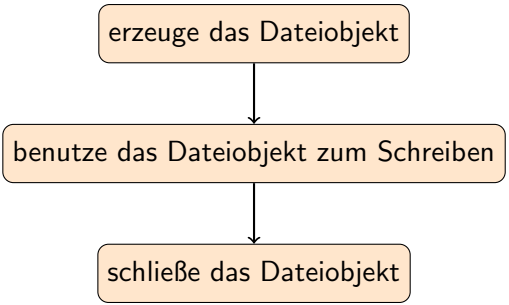
```
# Gib int n mit 5 Zeichen und float r mit 8 Zeichen und 3 Nachkommastellen aus.
print('(5) Die Wurzel von %5d ist %8.3f.' % (n, r))                    #  (5) Die Wurzel von  1000 ist   31.623.


# Gib int n mit 5 Zeichen und float r linksbündig mit 8 Zeichen und 3 Nachkommastellen aus
# und lasse den Zeilenumbruch am Ende der Ausgabe weg (geht nur in Python3).
print( '(6) Die Wurzel von %5d ist %-8.3f.' % (n, r) , end='' )


# Wenn die angegebene Zeichenzahl zu klein ist, wird sie ignoriert.
print( '(7) Die Wurzel von %2d ist %1.3f.' % (n, r) )

                                        #  (6) Die Wurzel von 1000 ist 31.623.(7) Die Wurzel von 1000 ist 31.623.
```

2 Ausgabe in Dateien

- Ausgabe in Dateiobjekt `f` mit `f.write()`
- Ausgabe in Dateiobjekt mit `print()`
- Ausgabe auf *standard output*
- Umleitung der Ausgabe von *standard output* in eine Datei

Ausgabe in Dateiobjekt `f` mit `f.write()`

Phasen der Benutzung eines Dateiobjekts:




```
f = open(dateiname, 'w')
```

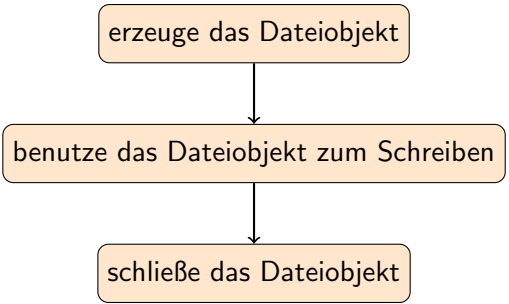
```
f.write(string)
```

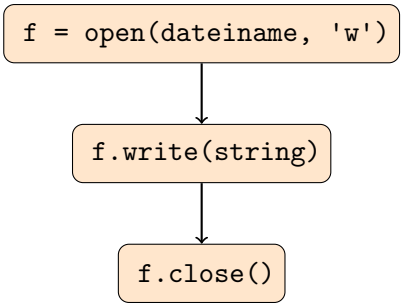
```
f.close()
```

- Mit `write()` können (hier) nur Strings geschrieben werden.
  Es findet keine automatische Umwandlung in Strings wie bei `print` statt.
- Alle Zeilenumbrüche müssen explizit geschrieben werden.

Ausgabe in Dateiobjekt `f` mit `f.write()`

Phasen der Benutzung eines Dateiobjekts:

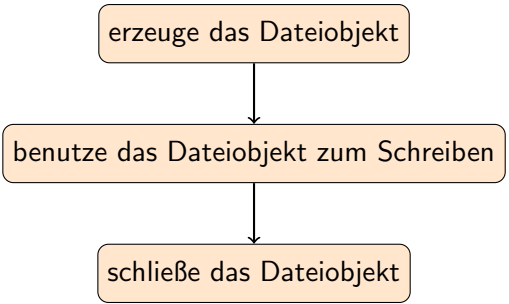


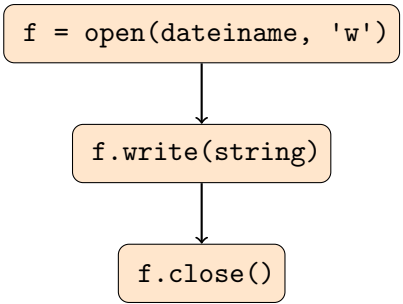


- Mit `write()` können (hier) nur Strings geschrieben werden.
  Es findet keine automatische Umwandlung in Strings wie bei `print` statt.
- Alle Zeilenumbrüche müssen explizit geschrieben werden.

Ausgabe in Dateiobjekt `f` mit `f.write()`

Phasen der Benutzung eines Dateiobjekts:





- Mit `write()` können (hier) nur Strings geschrieben werden.
  Es findet keine automatische Umwandlung in Strings wie bei `print` statt.
- Alle Zeilenumbrüche müssen explizit geschrieben werden.

```python
# ausgabe_mit_write.py
#-------------------------------------------------------------------------
# Lies n und einen Dateinamen von der Kommandozeile
# und gib n Würfe mit einem Würfel in die Datei mit dem Dateinamen aus.
# In jeder Zeile der Datei sollen 10 Würfe stehen.
#-------------------------------------------------------------------------
import sys, random

# Lies int n und str dateiname von der Kommandozeile.
n = int(sys.argv[1])
dateiname = sys.argv[2]

# Öffne die Datei dateiname zum Schreiben ('w'), d.h. als Ausgabedatei.
ausgabeDatei = open(dateiname, 'w')

# Schreibe n Würfelergebnisse als Strings in die Ausgabedatei.
# Alle 10 Würfelergebnisse wird ein Zeilenumbruch eingefügt.
for i in range(n):
  wurf = random.randint(1,6)
  ausgabeDatei.write(str(wurf))
  if (i+1)%10==0:
     ausgabeDatei.write('\n')

# Schließe die Ausgabedatei.
ausgabeDatei.close()
```

```python
# ausgabe_mit_write.py
#-------------------------------------------------------------------------
# Lies n und einen Dateinamen von der Kommandozeile
# und gib n Würfe mit einem Würfel in die Datei mit dem Dateinamen aus.
# In jeder Zeile der Datei sollen 10 Würfe stehen.
#-------------------------------------------------------------------------
import sys, random

# Lies int n und str dateiname von der Kommandozeile.
n = int(sys.argv[1])
dateiname = sys.argv[2]

# Öffne die Datei dateiname zum Schreiben ('w'), d.h.
ausgabeDatei = open(dateiname, 'w')

# Schreibe n Würfelergebnisse als Strings in die Ausga
# Alle 10 Würfelergebnisse wird ein Zeilenumbruch eing
for i in range(n):
  wurf = random.randint(1,6)
  ausgabeDatei.write(str(wurf))
  if (i+1)%10==0:
     ausgabeDatei.write('\n')

# Schließe die Ausgabedatei.
ausgabeDatei.close()
```

```
mundhenk@ma3: ~/Python/VL05-EinAus
mundhenk@home: python3 ausgabe_mit_write.py 42 www.txt
mundhenk@home:
```

```python
# ausgabe_mit_write.py
#-------------------------------------------------------------------------
# Lies n und einen Dateinamen von der Kommandozeile
# und gib n Würfe mit einem Würfel in die Datei mit dem Dateinamen aus.
# In jeder Zeile der Datei sollen 10 Würfe stehen.
#-------------------------------------------------------------------------
import sys, random

# Lies int n und str dateiname von der Kommandozeile.
n = int(sys.argv[1])
dateiname = sys.argv[2]

# Öffne die Datei dateiname zum Schreiben ('w'), d.h.
ausgabeDatei = open(dateiname, 'w')

# Schreibe n Würfelergebnisse als Strings in die Ausga
# Alle 10 Würfelergebnisse wird ein Zeilenumbruch eing
for i in range(n):
  wurf = random.randint(1,6)
  ausgabeDatei.write(str(wurf))
  if (i+1)%10==0:
     ausgabeDatei.write('\n')

# Schließe die Ausgabedatei.
ausgabeDatei.close()
```

```
mundhenk@ma3: ~/Python/VL05-EinAus
mundhenk@home: python3 ausgabe_mit_write.py 42 www.txt
mundhenk@home: cat www.txt
2432231556
4352524321
2344132116
3643244114
11mundhenk@home:
```

Ausgabe in Dateiobjekt `f` mit `print()`

Nach dem Erzeugen eines Dateiobjekts
kann man Ausgaben dorthin auch – wie gewohnt – mit `print()` machen.
Dazu muss man das Dateiobjekt als Argument mit dem Namen `file` angeben.

`print(x, file=f)` schreibe String `x` in das Dateiobjekt `f`

Am Ende muss das Dateiobjekt geschlossen werden.

```python
#-------------------------------------------------------------------------
# ausgabe_mit_print.py
#-------------------------------------------------------------------------
# Lies n und einen Dateinamen von der Kommandozeile
# und gib n Würfe mit einem Würfel in die Datei mit dem Dateinamen aus.
# Jeder Wurf steht in einer Zeile.
#-------------------------------------------------------------------------

import sys, random

# Lies int n und str dateiname von der Kommandozeile.
n = int(sys.argv[1])
dateiname = sys.argv[2]

# Öffne die Datei dateiname zum Schreiben ('w'), d.h. als Ausgabedatei.
ausgabeDatei = open(dateiname, 'w')

# Schreibe n Würfelergebnisse als Strings in die Ausgabedatei.
for i in range(n):
  wurf = random.randint(1,6)
  print(wurf, file=ausgabeDatei)

# Schließe die Ausgabedatei.
ausgabeDatei.close()
```

```python
#-------------------------------------------------------------------------
# ausgabe_mit_print.py
#-------------------------------------------------------------------------
# Lies n und einen Dateinamen von der Kommandozeile
# und gib n Würfe mit einem Würfel in die Datei mit dem Dateinamen aus.
# Jeder Wurf steht in einer Zeile.
#-------------------------------------------------------------------------

import sys, random

# Lies int n und str dateiname von der Kommandozeile.
n = int(sys.argv[1])
dateiname = sys.argv[2]

# Öffne die Datei dateiname zum Schreiben ('w'), d.h.
ausgabeDatei = open(dateiname, 'w')

# Schreibe n Würfelergebnisse als Strings in die Ausga
for i in range(n):
  wurf = random.randint(1,6)
  print(wurf, file=ausgabeDatei)

# Schließe die Ausgabedatei.
ausgabeDatei.close()
```

```
mundhenk@ma3: ~/Python/VL05-EinAus
mundhenk@home: python3 ausgabe_mit_print.py 8 www.txt
mundhenk@home:
```

```python
#-------------------------------------------------------------------------
# ausgabe_mit_print.py
#-------------------------------------------------------------------------
# Lies n und einen Dateinamen von der Kommandozeile
# und gib n Würfe mit einem Würfel in die Datei mit dem Dateinamen aus.
# Jeder Wurf steht in einer Zeile.
#-------------------------------------------------------------------------

import sys, random

# Lies int n und str dateiname von der Kommandozeile.
n = int(sys.argv[1])
dateiname = sys.argv[2]

# Öffne die Datei dateiname zum Schreiben ('w'), d.h.
ausgabeDatei = open(dateiname, 'w')

# Schreibe n Würfelergebnisse als Strings in die Ausga
for i in range(n):
  wurf = random.randint(1,6)
  print(wurf, file=ausgabeDatei)

# Schließe die Ausgabedatei.
ausgabeDatei.close()
```

```
mundhenk@ma3: ~/Python/VL05-EinAus
mundhenk@home: python3 ausgabe_mit_print.py 8 www.txt
mundhenk@home: cat www.txt
1
1
6
3
4
2
4
3
mundhenk@home:
```

Ausgabe auf standard output

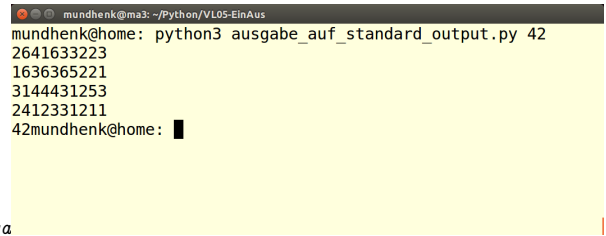
Standard output ist der Ausgabekanal, den wir in den bisherigen Vorlesungen benutzt haben.

Man kann also mit `print()` auf ihn schreiben.

Man kann auch das Dateiobjekt `sys.stdout` mit `write()` beschreiben.
Dazu muss man das Modul `sys` importieren.
`sys.stdout` wird „automatisch“ geöffnet und geschlossen.

```python
# ausgabe_auf_standard_output.py
#--------------------------------------------------------------------------
# Lies n von der Kommandozeile und gib n Würfe mit einem Würfel aus.
# In jeder Zeile sollen 10 Würfe stehen.
#--------------------------------------------------------------------------
import sys, random

# Lies int n von der Kommandozeile.
n = int(sys.argv[1])




# Schreibe n Würfelergebnisse als Strings in die Ausgabedatei.
for i in range(n):
  wurf = random.randint(1,6)
  sys.stdout.write(str(wurf))
  if (i+1)%10==0:
     sys.stdout.write('\n')
```

```python
# ausgabe_auf_standard_output.py
#-------------------------------------------------------------------------
# Lies n von der Kommandozeile und gib n Würfe mit einem Würfel aus.
# In jeder Zeile sollen 10 Würfe stehen.
#-------------------------------------------------------------------------
import sys, random

# Lies int n von der Kommandozeile.
n = int(sys.argv[1])



# Schreibe n Würfelergebnisse als Strings in die Ausga
for i in range(n):
  wurf = random.randint(1,6)
  sys.stdout.write(str(wurf))
  if (i+1)%10==0:
     sys.stdout.write('\n')
```

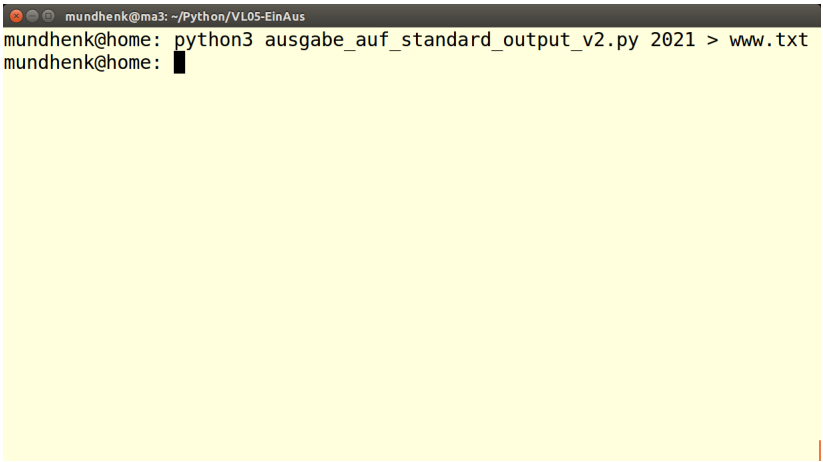


```python
# ausgabe_auf_standard_output_v2.py
#-------------------------------------------------------------------------
# Lies n von der Kommandozeile und gib n Würfe mit einem Würfel aus.
# In jeder Zeile sollen 10 Würfe stehen.
#-------------------------------------------------------------------------
import sys, random

# Lies int n von der Kommandozeile.
n = int(sys.argv[1])

# Benenne sys.stdout durch ausgabeDatei.
ausgabeDatei = sys.stdout

# Schreibe n Würfelergebnisse als Strings in die Ausga
for i in range(n):
  wurf = random.randint(1,6)
  ausgabeDatei.write(str(wurf))
  if (i+1)%10==0:
     ausgabeDatei.write('\n')
```

```
mundhenk@ma3: ~/Python/VL05-EinAus
mundhenk@home: python3 ausgabe_auf_standard_output.py 42
2641633223
1636365221
3144431253
2412331211
42mundhenk@home:
```

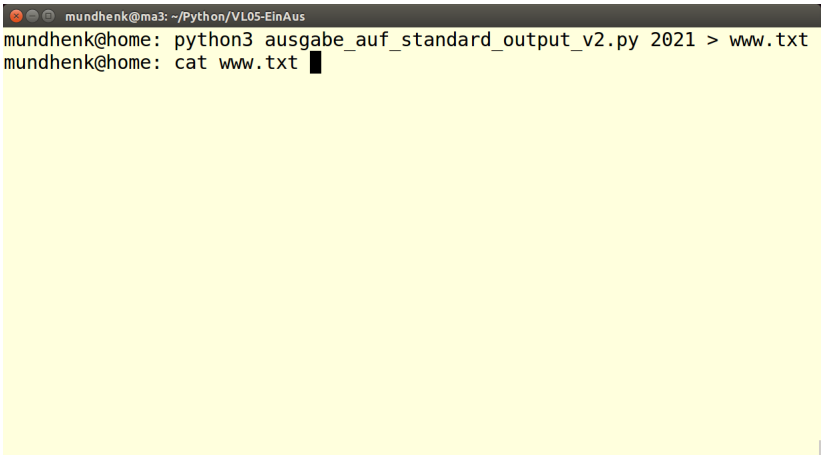
Umleitung von standard output in eine Datei

Ausgabeumleitung >



Umleitung von standard output in eine Datei

Ausgabeumleitung >

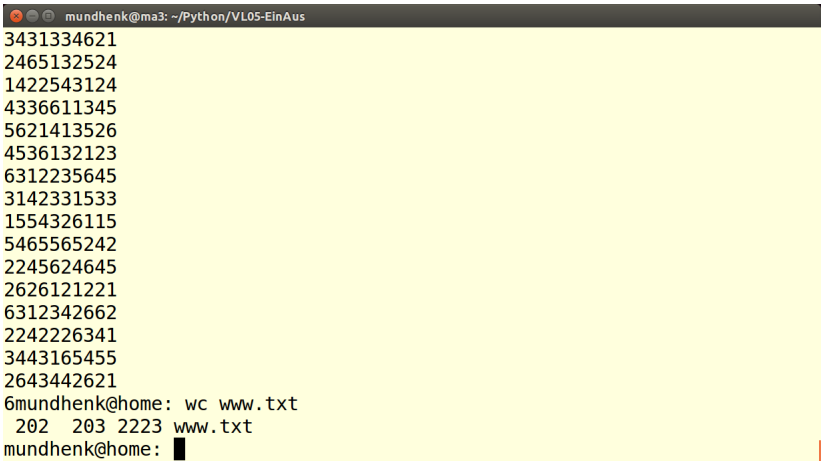


Umleitung von standard output in eine Datei

Ausgabeumleitung >

Umleitung von standard output in eine Datei

Ausgabeumleitung >



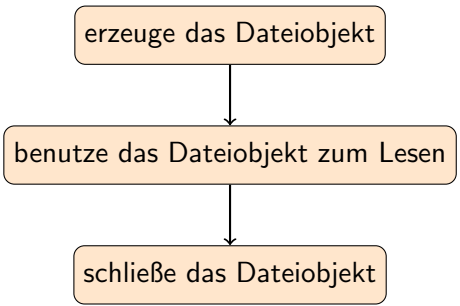
wc - print newline, word, and byte counts for each file

3 Eingabe aus Dateien

- Eingabe aus Dateiobjekt `f`
  - `f.read()`
  - `f.readline()`
  - `f.readlines()`
  - `for zeile in f:`
- Eingabe aus *standard input*
- Umleitung einer Datei in *standard input* zur Eingabe in ein Programm
- Nutzung der Ausgabe eines Programms als Eingabe für ein anderes Programm

Eingabe aus Dateiobjekt `f`

Phasen der Benutzung eines Dateiobjekts:




```
f = open(dateiname, 'r')
```

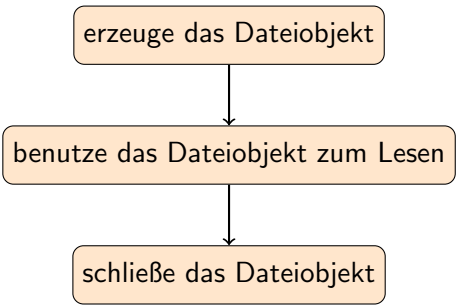
```
string = f.readxyz()
```

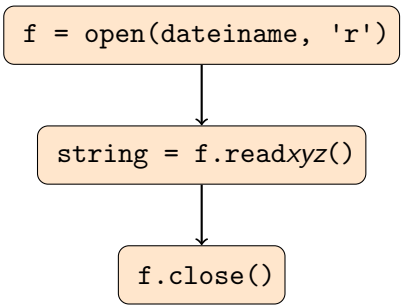
```
f.close()
```

- Mit den Lesefunktionen `readxyz()` können (hier) nur Strings gelesen werden.
- Alle Zeilenumbrüche etc. werden mitgelesen und müssen ggf. vom Programm entfernt werden.

Eingabe aus Dateiobjekt `f`

Phasen der Benutzung eines Dateiobjekts:





- Mit den Lesefunktionen `read`*xyz*`()` können (hier) nur Strings gelesen werden.
- Alle Zeilenumbrüche etc. werden mitgelesen und müssen ggf. vom Programm entfernt werden.

Eingabe aus Dateiobjekt `f`

Phasen der Benutzung eines Dateiobjekts:

- Mit den Lesefunktionen `read`*xyz*`()` können (hier) nur Strings gelesen werden.
- Alle Zeilenumbrüche etc. werden mitgelesen und müssen ggf. vom Programm entfernt werden.

Benutzung von Dateiobjekt `f` zum Lesen

`f.read()` hat die ganze Datei als einen String als Ergebnis.

`f.readlines()` hat die ganze Datei als Array aus ihren Zeilen als Ergebnis.

`f.readline()` hat die nächste Zeile der Datei inklusive abschließendem `\n` als Ergebnis.
Wenn bereits die ganze Datei gelesen wurde, ist das Ergebnis `''` (der leere String mit Länge 0).

`for zeile in f:` durchläuft alle Zeilen der Datei in einer Schleife.
`zeile` (String) enthält die „aktuelle“ Zeile der Datei.
Die `for`-Schleife wird beendet, wenn alle Zeilen der Datei durchlaufen wurden.

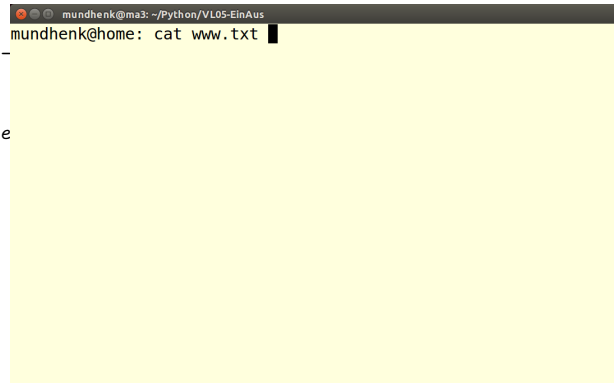
Eine *Zeile* endet (in der Regel) mit `\n`.
Die letzte Zeile einer Datei kann ohne `\n` enden.

f.read()

```
# eingabe_mit_read.py
#-------------------------------------------------------------------------
# Lies einen Dateinamen von der Kommandozeile,
# lies die Datei mit read() ein,
# und gib sie aus.
#-------------------------------------------------------------------------
import sys

# Lies einen Dateinamen (string) von der Kommandozeile.
dateiname = sys.argv[1]

# Öffne die Datei ...
eingabeDatei = open(dateiname,'r')

# ... lies sie ("im Stück") ein ...
inhalt = eingabeDatei.read()

# ... und schließe sie wieder.
eingabeDatei.close()

# Gib den Inhalt der Datei aus.
print(inhalt)
```

f.read()

```
# eingabe_mit_read.py
#-------------------------------------------------------------------------
# Lies einen Dateinamen von der Kommandozeile,
# lies die Datei mit read() ein,
# und gib sie aus.
#-------------------------------------------------------
import sys

# Lies einen Dateinamen (string) von der Kommandozeile
dateiname = sys.argv[1]

# Öffne die Datei ...
eingabeDatei = open(dateiname,'r')

# ... lies sie ("im Stück") ein ...
inhalt = eingabeDatei.read()

# ... und schließe sie wieder.
eingabeDatei.close()

# Gib den Inhalt der Datei aus.
print(inhalt)
```



f.read()

```python
# eingabe_mit_read.py
#-------------------------------------------------------------------------
# Lies einen Dateinamen von der Kommandozeile,
# lies die Datei mit read() ein,
# und gib sie aus.
#-------------------------------------------------------
import sys

# Lies einen Dateinamen (string) von der Kommandozeile
dateiname = sys.argv[1]

# Öffne die Datei ...
eingabeDatei = open(dateiname,'r')

# ... lies sie ("im Stück") ein ...
inhalt = eingabeDatei.read()

# ... und schließe sie wieder.
eingabeDatei.close()

# Gib den Inhalt der Datei aus.
print(inhalt)
```

```
mundhenk@ma3: ~/Python/VL05-EinAus
mundhenk@home: cat www.txt
4232353344
5323211464
3542642624
2432232264
21mundhenk@home:
```

f.read()

```
# eingabe_mit_read.py
#-------------------------------------------------------------------------
# Lies einen Dateinamen von der Kommandozeile,
# lies die Datei mit read() ein,
# und gib sie aus.
#-------------------------------------------------------
import sys

# Lies einen Dateinamen (string) von der Kommandozeile
dateiname = sys.argv[1]

# Öffne die Datei ...
eingabeDatei = open(dateiname,'r')

# ... lies sie ("im Stück") ein ...
inhalt = eingabeDatei.read()

# ... und schließe sie wieder.
eingabeDatei.close()

# Gib den Inhalt der Datei aus.
print(inhalt)
```

```
mundhenk@ma3: ~/Python/VL05-EinAus
mundhenk@home: cat www.txt
4232353344
5323211464
3542642624
2432232264
2mundhenk@home: python3 eingabe_mit_read.py www.txt
4232353344
5323211464
3542642624
2432232264
21
mundhenk@home:
```

f.readlines()

```
#-------------------------------------------------------------------------
# eingabe_mit_readlines.py
#-------------------------------------------------------------------------
# Lies einen Dateinamen von der Kommandozeile,
# lies die Datei mit readlines() ein,
# und gib sie aus.
#-------------------------------------------------------------------------
import sys

# Lies einen Dateinamen (string) von der Kommandozeile.
dateiname = sys.argv[1]

# Öffne die Datei,
# lies sie mit readlines() ein,
# und schließe sie wieder.
eingabeDatei = open(dateiname,'r')
inhalt = eingabeDatei.readlines()
eingabeDatei.close()

# Gib den Inhalt der Datei aus.
print(inhalt)
```

f.readlines()

```
#-------------------------------------------------------------------------
# eingabe_mit_readlines.py
#-------------------------------------------------------------------------
# Lies einen Dateinamen von der Kommandozeile
# lies die Datei mit readlines() ein,
# und gib sie aus.
#-----------------------------------------
import sys

# Lies einen Dateinamen (string) von der Komm
dateiname = sys.argv[1]

# Öffne die Datei,
# lies sie mit readlines() ein,
# und schließe sie wieder.
eingabeDatei = open(dateiname,'r')
inhalt = eingabeDatei.readlines()
eingabeDatei.close()

# Gib den Inhalt der Datei aus.
print(inhalt)
```

```
mundhenk@ma3: ~/Python/VL05-EinAus
mundhenk@home: cat www.txt
4232353344
5323211464
3542642624
2432232264
21mundhenk@home: python3 eingabe_mit_read.py www.txt
4232353344
5323211464
3542642624
2432232264
21
mundhenk@home:
```

f.readlines()

```
#-------------------------------------------------------------------------
# eingabe_mit_readlines.py
#-------------------------------------------------------------------------
# Lies einen Dateinamen von der Kommandozeile
# lies die Datei mit readlines() ein,
# und gib sie aus.
#-----------------------------------------
import sys

# Lies einen Dateinamen (string) von der Komm
dateiname = sys.argv[1]

# Öffne die Datei,
# lies sie mit readlines() ein,
# und schließe sie wieder.
eingabeDatei = open(dateiname,'r')
inhalt = eingabeDatei.readlines()
eingabeDatei.close()

# Gib den Inhalt der Datei aus.
print(inhalt)
```

```
mundhenk@ma3: ~/Python/VL05-EinAus
mundhenk@home: cat www.txt
4232353344
5323211464
3542642624
2432232264
21mundhenk@home: python3 eingabe_mit_read.py www.txt
4232353344
5323211464
3542642624
2432232264
21
mundhenk@home: python3 eingabe_mit_readlines.py www.txt
['4232353344\n', '5323211464\n', '3542642624\n', '2432232264\n', '21']
mundhenk@home:
```

f.readline()

```
# eingabe_mit_readline.py
#-------------------------------------------------------------------------------
# Lies einen Dateinamen von der Kommandozeile,
# lies die Datei zeilenweise mit readline() ein, und gib die Zeilen aus.
#-------------------------------------------------------------------------------
import sys
# Lies einen Dateinamen (string) von der Kommandozeile und öffne die Datei zum Lesen.
eingabeDatei = open(sys.argv[1],'r')

# Die while-Schleife wird für jede Zeile der Datei (string s) einmanl durchlaufen.
# Jede Zeile wird zusammen mit einem Zeilenzähler ausgegeben.
# Wenn s=='', dann ist das Ende der Datei erreicht worden
#   und das Lesen wird beendet.
s = eingabeDatei.readline()
zaehler = 0
while s != '':
  print(zaehler, s)
  zaehler += 1
  s = eingabeDatei.readline()

# Schließe die Datei.
eingabeDatei.close()
```

f.readline()

```
# eingabe_mit_readline.py
#-------------------------------------------------------------------------------
# Lies einen Dateinamen von der Kommandozeile,
# lies die Datei zeilenweise mit readline() ein, und gib die Zeilen aus
#-------------------------------------------------------------------
import sys
# Lies einen Dateinamen (string) von der Kommandozeile
eingabeDatei = open(sys.argv[1],'r')

# Die while-Schleife wird für jede Zeile der Datei (st
# Jede Zeile wird zusammen mit einem Zeilenzähler ausg
# Wenn s=='', dann ist das Ende der Datei erreicht wor
#   und das Lesen wird beendet.
s = eingabeDatei.readline()
zaehler = 0
while s != '':
  print(zaehler, s)
  zaehler += 1
  s = eingabeDatei.readline()

# Schließe die Datei.
eingabeDatei.close()
```

```
mundhenk@ma3: ~/Python/VL05-EinAus
mundhenk@home: cat www.txt
4232353344
5323211464
3542642624
2432232264
21mundhenk@home:
```

f.readline()

```
# eingabe_mit_readline.py
#-------------------------------------------------------------------------------
# Lies einen Dateinamen von der Kommandozeile,
# lies die Datei zeilenweise mit readline() ein, und gib die Zeilen aus
#-------------------------------------------------------------------------------
import sys
# Lies einen Dateinamen (string) von der Kommandozeile
eingabeDatei = open(sys.argv[1],'r')

# Die while-Schleife wird für jede Zeile der Datei (st
# Jede Zeile wird zusammen mit einem Zeilenzähler ausg
# Wenn s=='', dann ist das Ende der Datei erreicht wor
#   und das Lesen wird beendet.
s = eingabeDatei.readline()
zaehler = 0
while s != '':
  print(zaehler, s)
  zaehler += 1
  s = eingabeDatei.readline()

# Schließe die Datei.
eingabeDatei.close()
```

```
mundhenk@ma3: ~/Python/VL05-EinAus
mundhenk@home: cat www.txt
4232353344
5323211464
3542642624
2432232264
21mundhenk@home: python3 eingabe_mit_readline.py www.txt
0 4232353344

1 5323211464

2 3542642624

3 2432232264

4 21
mundhenk@home:
```

for zeile in f:

```
#-------------------------------------------------------------------------
# eingabe_mit_for.py
#-------------------------------------------------------------------------
# Lies einen Dateinamen von der Kommandozeile,
# durchlaufe die Datei zeilenweise mit for, und gib sie zeilenweise aus.
#-------------------------------------------------------------------------
import sys

# Öffne die Datei mit dem auf der Kommandozeile eingegebenen Namen.
eingabeDatei = open(sys.argv[1],'r')

# Lies die Datei zeilenweise ein und zähle die gelesenen Zeilen.
# Gib jede Zeile mit diesem Zähler aus.
zaehler = 0
for zeile in eingabeDatei:
  print(zaehler, zeile)
  zaehler += 1

# Schließe die Datei.
eingabeDatei.close()
```

for zeile in f:

```
#-------------------------------------------------------------------------
# eingabe_mit_for.py
#-------------------------------------------------------------------------
# Lies einen Dateinamen von der Kommandozeile,
# durchlaufe die Datei zeilenweise mit for, und gib si
#-------------------------------------------------------------------------
import sys

# Öffne die Datei mit dem auf der Kommandozeile eingeg
eingabeDatei = open(sys.argv[1],'r')

# Lies die Datei zeilenweise ein und zähle die gelesen
# Gib jede Zeile mit diesem Zähler aus.
zaehler = 0
for zeile in eingabeDatei:
  print(zaehler, zeile)
  zaehler += 1

# Schließe die Datei.
eingabeDatei.close()
```

```
mundhenk@ma3: ~/Python/VL05-EinAus
mundhenk@home: cat www.txt
4232353344
5323211464
3542642624
2432232264
21mundhenk@home:
```

for zeile in f:

```
#-------------------------------------------------------------------------
# eingabe_mit_for.py
#-------------------------------------------------------------------------
# Lies einen Dateinamen von der Kommandozeile,
# durchlaufe die Datei zeilenweise mit for, und gib si
#-------------------------------------------------------------------------
import sys

# Öffne die Datei mit dem auf der Kommandozeile eingeg
eingabeDatei = open(sys.argv[1],'r')

# Lies die Datei zeilenweise ein und zähle die gelesen
# Gib jede Zeile mit diesem Zähler aus.
zaehler = 0
for zeile in eingabeDatei:
  print(zaehler, zeile)
  zaehler += 1

# Schließe die Datei.
eingabeDatei.close()
```

```
mundhenk@ma3: ~/Python/VL05-EinAus
mundhenk@home: cat www.txt
4232353344
5323211464
3542642624
2432232264
21mundhenk@home: python3 eingabe_mit_for.py www.txt
0 4232353344

1 5323211464

2 3542642624

3 2432232264

4 21
mundhenk@home:
```

Eingabe von standard input

Standard input ist der Eingabekanal, den wir in den bisherigen Vorlesungen benutzt haben (außer bei Eingaben von der Kommandozeile).

Man kann also mit `input()` zeilenweise von ihm lesen.

Man kann ihn auch als das Dateiobjekt `sys.stdin` (aus dem Modul `sys`) benutzen.
`<Strg+d>` ist das Zeichen für „Ende der Datei“, das man an der Tastatur eingeben kann.
`sys.stdin` wird „automatisch“ geöffnet und geschlossen.

Eingabe von standard input lesen

Eingabe kommt von der Tastatur

```
#-------------------------------------------------------------------------
# eingabe_von_stdin.py
#-------------------------------------------------------------------------
# Lies die Eingaben von der Tastataur zeilenweise,
# und gib sie zeilenweise aus.
#-------------------------------------------------------------------------
import sys

# Öffne die Datei mit dem auf der Kommandozeile eingegebenen Namen.
eingabeDatei = sys.stdin

# Lies die Datei zeilenweise ein und zähle die gelesenen Zeilen.
# Gib jede Zeile mit diesem Zähler aus.
zaehler = 0
for zeile in eingabeDatei:
  print(zaehler, zeile)
  zaehler += 1

# Schließe die Datei.
eingabeDatei.close()
```

Eingabe von standard input lesen

Eingabe kommt von der Tastatur

```
#-------------------------------------------------------------------------
# eingabe_von_stdin.py
#-------------------------------------------------------------------------
# Lies die Eingaben von der Tastataur zeilenweise,
# und gib sie zeilenweise aus.
#-------------------------------------------------------
import sys

# Öffne die Datei mit dem auf der Kommandozeile eingeg
eingabeDatei = sys.stdin

# Lies die Datei zeilenweise ein und zähle die gelesen
# Gib jede Zeile mit diesem Zähler aus.
zaehler = 0
for zeile in eingabeDatei:
  print(zaehler, zeile)
  zaehler += 1

# Schließe die Datei.
eingabeDatei.close()
```

```
mundhenk@ma3: ~/Python/VL05-EinAus
mundhenk@home: python3 eingabe_von_stdin.py
Heute ist die
0 Heute ist die

5. Vorlesung.
1 5. Vorlesung.


2

Das war eine Leerzeile.
3 Das war eine Leerzeile.

Die nächste Eingabe ist <Strg+d>.
4 Die nächste Eingabe ist <Strg+d>.

mundhenk@home:
```

Eingabe von standard input lesen

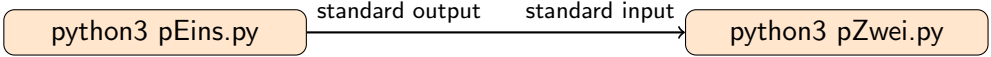
Eingabe kommt aus einer Datei (mit Eingabeumleitung <)

```
#-------------------------------------------------------------------------
# eingabe_von_stdin.py
#-------------------------------------------------------------------------
# Lies die Eingaben von der Tastataur zeilenweise,
# und gib sie zeilenweise aus.
#-------------------------------------------------------------
import sys

# Öffne die Datei mit dem auf der Kommandozeile eingeg
eingabeDatei = sys.stdin

# Lies die Datei zeilenweise ein und zähle die gelesen
# Gib jede Zeile mit diesem Zähler aus.
zaehler = 0
for zeile in eingabeDatei:
  print(zaehler, zeile)
  zaehler += 1

# Schließe die Datei.
eingabeDatei.close()
```

```
mundhenk@ma3: ~/Python/VL05-EinAus
mundhenk@home: cat www.txt
4232353344
5323211464
3542642624
2432232264
21mundhenk@home: python3 eingabe_von_stdin.py < www.txt
0 4232353344

1 5323211464

2 3542642624

3 2432232264

4 21
mundhenk@home:
```

Ausgaben eines Programms als Eingabe eines anderen Programms nutzen

Ausgabeweiterleitung |




```
mundhenk@ma3: ~/Python/VL05-EinAus
mundhenk@home: python3 ausgabe_auf_standard_output.py 42
6632515662
6666443236
2223411651
3255214346
45mundhenk@home: python3 ausgabe_auf_standard_output.py 42 | python3 eingabe_von_stdin.py
0 4415163356

1 5435421665

2 6563432432

3 4341451466

4 35
mundhenk@home:
```

Zusammenfassung (Teil 1)

Dateiobjekte kann man zum Schreiben und Lesen (Aus- und Eingabe) von Strings benutzen.

Dafür gibt es die Methoden

- zum Schreiben: `write()`
- zum Lesen: `readline()`, `for z in f:` (und `read()`, `readlines()`).

Standard output und standard input sind häufig benutzte Kanäle zur Aus- und Eingabe.
Mit > und < kann man sie zum Schreiben und Lesen von Dateien verwenden.
Mit | kann man Programme hintereinanderschalten.
`print()` und `input()` sind dafür zugeschnittene Kurzschreibweisen.

4 Programmier-Auftrag: bestimme die mittleren Tageshöchsttemperaturen in Jena pro Jahr aus Messdaten vom DWD

Der Deutsche Wetterdienst (DWD) stellt Messwerte von vielen Wetterstationen als Text-Dateien bereit (siehe `https://opendata.dwd.de/climate_environment/CDC/observations_germany/climate/daily/kl/historical/`).

Dort kann man Messdaten der Station Jena (Sternwarte) seit 1824 sehen.

Die Datei damit hat 69986 Zeilen und sieht ungefähr wie folgt aus.

```
STATIONS_ID;MESS_DATUM;QN_3;  FX;  FM;QN_4; RSK;RSKF; SDK;SHK_TAG;  NM; VPM;  PM; TMK; UPM; TXK; TNK; TGK;eor
     2444;18240101;-999;-999;-999;    2;  -999;-999;-999;-999;  -999;  -999;  990.70;  4.5;-999;    5.5;  1.6;-999;eor
     2444;18240102;-999;-999;-999;    2;  -999;-999;-999;-999;  -999;  -999;  984.00;  5.0;-999;    6.5;  3.1;-999;eor
     2444;18240103;-999;-999;-999;    2;  -999;-999;-999;-999;  -999;  -999;  997.80;  3.5;-999;    5.0;  1.9;-999;eor
     2444;18240104;-999;-999;-999;    2;  -999;-999;-999;-999;  -999;  -999; 1019.50;  1.2;-999;    2.3; -2.9;-999;eor
     2444;18240105;-999;-999;-999;    2;  -999;-999;-999;-999;  -999;  -999; 1021.30; -0.8;-999;    3.0;  -3.8;-999;eor
     2444;18240106;-999;-999;-999;    2;  -999;-999;-999;-999;  -999;  -999; 1013.80; -3.7;-999;    0.0; -10.4;-999;eor
...
     2444;18241231;-999;-999;-999;    2;  -999;-999;-999;-999;  -999;  -999; 1013.30;  6.6; 62.00;  7.0;  0.6;-999;eor
     2444;18250101;-999;-999;-999;    2;  -999;-999;-999;-999;  -999;  -999; 1008.00;  8.2; 54.00;  9.1;  2.1;-999;eor
...
     2444;20191231;-999;-999;-999;    3;   0.0;   0;-999;   0;  -999;   5.5;    -999;  3.3; 70.92;  6.3; -0.5;-2.7;eor
```

Aus dieser Datei sollen die mittleren Tageshöchsttemperaturen pro Jahr berechnet werden.

Beschreibung der Datei mit den Messdaten (aus Jena)

Unter `https://opendata.dwd.de/climate_environment/CDC/observations_germany/climate/daily/kl/historical/DESCRIPTION_obsgermany_climate_daily_kl_historical_en.pdf` findet man eine ausführliche Beschreibung der Daten.

Die für uns möglicherweise interessanten Daten eines Tages findet man in folgenden Spalten:

Wert	Format	Spalte
Station		0
Mess-Datum	YYYYMMDD	1
Tages-Niederschlagmenge		6
Tages-Sonnenscheindauer		8
Tagesmittel des Luftdrucks		12
Tagesmittel der Temperatur		13
Tagesmaximum der Lufttemperatur in 2m Höhe		15
Tagesminimum der Lufttemperatur in 2m Höhe		16
Tagesminimum der Lufttemperatur in 5cm Höhe		17

Welche Daten brauchen wir?

Wir wollen
die Jahresdurchschnitte der Tageshöchsttemperaturen in Jena
für jedes Jahr seit 1824 berechnen.

Wir brauchen also aus jeder Zeile/von jedem Tag
das Jahr und die Tageshöchsttemperatur.

Daraus können wir die mittlere Tageshöchsttemperatur für jedes Jahr berechnen.

Welche Daten brauchen wir?

Wir wollen
die Jahresdurchschnitte der Tageshöchsttemperaturen in Jena
für jedes Jahr seit 1824 berechnen.

Wir brauchen also aus jeder Zeile/von jedem Tag
das Jahr und die Tageshöchsttemperatur.

Daraus können wir die mittlere Tageshöchsttemperatur für jedes Jahr berechnen.

Welche Daten brauchen wir?

Wir wollen
die Jahresdurchschnitte der Tageshöchsttemperaturen in Jena
für jedes Jahr seit 1824 berechnen.

Wir brauchen also aus jeder Zeile/von jedem Tag
das Jahr und die Tageshöchsttemperatur.

Daraus können wir die mittlere Tageshöchsttemperatur für jedes Jahr berechnen.

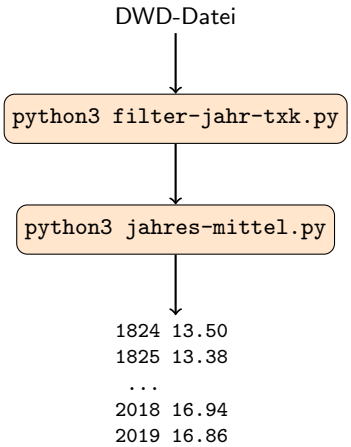
Die grobe Struktur für den ganzen Programmier-Auftrag

Wir teilen den Auftrag in zwei Teile.

Teil 1: filtere aus der DWD-Datei die relevanten Daten heraus, also Jahr und Tageshöchsttemperatur für jeden Tag

Teil 2: berechne daraus die Jahresmittelwerte

Die Ausgabe von Teil 1 wird als Eingabe von Teil 2 benutzt.
Diese „Schnittstelle“ (Form der ausgegebenen/zu lesenden Daten) muss vereinbart werden.



Die Schnittstelle zwischen den beiden Programmen

Das Programm für Teil 1,

das Jahr und Tageshöchsttemperatur zeilenweise aus der großen Datei herausfiltert,

soll Daten in folgender Form liefern („*Jahr Leerzeichen Tageshöchsttemperatur*“) :

```
1824 5.5
1824 6.5
...
1824 7.0
1825 9.1
...
```

Die Idee und die grobe Struktur des Programmes für Teil 1

- Lies die Eingabedatei zeilenweise von standard input.
- Filtere aus jeder Zeile das Jahr und die Tageshöchsttemperatur heraus und gib sie aus.

Das konkrete Ziel am Beispiel einer Zeile: aus

```
2444;18240101;-999;-999;-999; 2; -999;-999;-999;-999;  -999;  -999;  990.70;  4.5;-999;  5.5;  1.6;-999;eor
```

soll

```
     1824                                                                        5.5
```

herausgefiltert werden.

Wir haben ja die Vorstellung, dass die DWD-Datei aus Spalten besteht.
Das Jahr steht in Spalte 1 (mit Monat und Tag)
und die Tageshöchsttemperatur steht in Spalte 15.

Wie können wir einfach jede Zeile in diese Spalten zerlegen?

Die Idee und die grobe Struktur des Programmes für Teil 1

- Lies die Eingabedatei zeilenweise von standard input.
- Filtere aus jeder Zeile das Jahr und die Tageshöchsttemperatur heraus und gib sie aus.

Das konkrete Ziel am Beispiel einer Zeile: aus

```
2444;18240101;-999;-999;-999; 2; -999;-999;-999;-999;  -999;  -999;  990.70;  4.5;-999;  5.5;  1.6;-999;eor
```

soll

```
    1824                                                                          5.5
```

herausgefiltert werden.

Wir haben ja die Vorstellung, dass die DWD-Datei aus Spalten besteht.
Das Jahr steht in Spalte 1 (mit Monat und Tag)
und die Tageshöchsttemperatur steht in Spalte 15.

Wie können wir einfach jede Zeile in diese Spalten zerlegen?

Die Funktionen `str.split()` und `str.strip()`

`str.split(string, trenner)` teilt einen String überall dort,
wo das Zeichen `trenner` in ihm vorkommt, und gibt diese Teile als Array zurück.

Beispiel: In

```
zeile  = '2444;18240101; 5.5; 1.6; -999 ;eor\n'
spalten = str.split(zeile, ';')
```

ist `spalten` das Array

```
[ '2444', '18240101', ' 5.5', ' 1.6', ' -999 ', 'eor\n' ]
```

`str.split(string)` teilt `string` an jedem *whitespace* (Leerzeichen, Zeilenumbruch, Tab etc.)
und entfernt auch alle whitespaces aus dem Ergebnis.

`str.strip(string)` gibt einen String zurück, der aus `string` entsteht, indem
alle whitespaces am Anfang und am Ende entfernt werden.

Bsp.: `str.strip('  ab c \n ')` ergibt `'ab c'`.

Die Umsetzung des Filterns einer Zeile

Im Beispiel

```
zeile  = '2444;18240101;-999;-999;-999; 2;-999;-999;-999;-999;-999;-999; 990.70; 4.5;-999; 5.5; 1.6;-999;eo
spalten = str.split(zeile, ';')
```

ist `spalten` das Array

```
['2444', '18240101', '-999', '-999', '-999', ' 2', '-999', '-999', '-999', '-999', '-999', '-999',
                                        ' 990.70', ' 4.5', '-999', ' 5.5', ' 1.6', '-999', 'eor' ]
```

Das Jahr `'1824'` ist `spalten[1][0:4]` (Typ `str`),

und der gesuchte Temperaturwert `5.5` ist `float(spalten[15])` (Typ `float`).

- teile die Zeile in ihre Spalten auf
- das Jahr steht in Spalte `[1]` in den ersten 4 Stellen
- die Tageshöchsttemperatur steht in Spalte `[15]`

Zusammenfassung der Struktur des Programmes für Teil 1

- Lies die Eingabedatei zeilenweise von standard input.
  - Teile die Zeile in ihre Spalten auf.
  - Das Jahr steht in Spalte `[1]` in den ersten 4 Stellen.
  - Die Tageshöchsttemperatur steht in Spalte `[15]`.
  - Gib Jahr und Tageshöchsttemperatur aus.

Das Programm `filter-jahr-txk.py` (für Teil 1)

```
import sys

sys.stdin.readline()    # Überspringe die erste Zeile von standard input.
                        # In der DWD-Datei stehen dort keine relevanten Daten.

# Gehe alle weiteren Zeilen von standard input durch.
for zeile in sys.stdin:
  # Trenne die Zeile an den ';'s in ihre Spalten auf.
  spalten = str.split(zeile, ';')
  # Das Jahr steht in den ersten vier Zeichen von Spalte 1.
  jahr = spalten[1][0:4]
  # Die Tageshöchsttemperatur steht in Spalte 15.
  # Wir entfernen gleich "störende" Leerzeichen etc. am Anfang und am Ende.
  txk = str.strip(spalten[15])
  # Jahr und Tageshöchsttemperatur werden ausgegeben, falls es kein Fehl-Eintrag ist.
  if txk != '-999':
    print( '%s %s' % (jahr, txk) )
```

Das Programm `filter-jahr-txk.py` (für Teil 1)

```
import sys

sys.stdin.readline()    # Überspringe die er
                        # In der DWD-Datei s

# Gehe alle weiteren Zeilen von standard in
for zeile in sys.stdin:
  # Trenne die Zeile an den ';'s in ihre Sp
  spalten = str.split(zeile, ';')
  # Das Jahr steht in den ersten vier Zeich
  jahr = spalten[1][0:4]
  # Die Tageshöchsttemperatur steht in Spal
  # Wir entfernen gleich "störende" Leerzei
  txk = str.strip(spalten[15])
  # Jahr und Tageshöchsttemperatur werden a
  if txk != '-999':
    print( '%s %s' % (jahr, txk) )
```

```
mundhenk@ma3: ~/Python/VL05-EinAus

mundhenk@home: python3 filter-jahr-txk.py < Jena-Wetterdaten.txt
1824 5.5
1824 6.5
1824 5.0
1824 2.3
1824 3.0
1824 0.0
1824 -1.4
1824 0.0
1824 -2.5
1824 1.3
1824 -2.1
1824 0.0
1824 2.5
```

Das Programm `filter-jahr-txk.py` (für Teil 1)

```
import sys

sys.stdin.readline()    # Überspringe die er
                        # In der DWD-Datei s

# Gehe alle weiteren Zeilen von standard in
for zeile in sys.stdin:
  # Trenne die Zeile an den ';'s in ihre Sp
  spalten = str.split(zeile, ';')
  # Das Jahr steht in den ersten vier Zeich
  jahr = spalten[1][0:4]
  # Die Tageshöchsttemperatur steht in Spal
  # Wir entfernen gleich "störende" Leerzei
  txk = str.strip(spalten[15])
  # Jahr und Tageshöchsttemperatur werden a
  if txk != '-999':
    print( '%s %s' % (jahr, txk) )
```

```
mundhenk@ma3: ~/Python/VL05-EinAus
2019 10.5
2019 8.2
2019 11.7
2019 10.1
2019 8.6
2019 10.2
2019 8.6
2019 8.1
2019 6.7
2019 5.7
2019 2.4
2019 2.8
2019 9.6
2019 6.3
mundhenk@home:
```

Teil 2 – Berechnung der Jahresmittelwerte: Idee und grobe Programm-Struktur

- Lies die Eingabedaten zeilenweise ein (Jahr, Temperatur).
- Addiere die Temperaturen für ein Jahr auf und gib den Mittelwert aus.

---

1. für jede Zeile der Eingabe aus Jahr und Temperatur:

  1.1 falls sich das Jahr nicht geändert hat,

    addiere die Temperatur zur Summe der Temperaturen des Jahres

  1.2 sonst:

    gib Jahr und Mittelwerte aus

    beginne mit der Aufsummierung für das neue Jahr

```
1824 5.5
1824 6.5
1824 6.2
...
1824 4.2
1824 8.2
1824 7.0
1825 9.1
1825 8.7
1825 5.4
...
```

Teil 2 – Berechnung der Jahresmittelwerte: Idee und grobe Programm-Struktur

- Lies die Eingabedaten zeilenweise ein (Jahr, Temperatur).
- Addiere die Temperaturen für ein Jahr auf und gib den Mittelwert aus.

---

1. für jede Zeile der Eingabe aus Jahr und Temperatur:

  1.1 falls sich das Jahr nicht geändert hat,

    addiere die Temperatur zur Summe der Temperaturen des Jahres

  1.2 sonst:

    gib Jahr und Mittelwerte aus

    beginne mit der Aufsummierung für das neue Jahr

```
1824 5.5
1824 6.5
1824 6.2
...
1824 4.2
1824 8.2
1824 7.0
1825 9.1
1825 8.7
1825 5.4
...
```

Das Programm jahres-mittel.py

```
import sys
# Die erste Zeile wird gelesen. Damit kennt man das Jahr (jahr),
# den Startwert für die Summe aller Messwerte des Jahres (jahressumme) und
# die bisher aufsummierte Anzahl von Messwerten für das Jahr (zaehler).
zeile = sys.stdin.readline()
jahr, messwert = str.split(zeile)
jahressumme = float(messwert)
zaehler = 1
# Die übrigen Zeilen werden gelesen.
for zeile in sys.stdin:
  # Die Zeile besteht aus einem Jahr (njahr) und einem Messwert.
  njahr, messwert = str.split(zeile)
  if njahr==jahr:                    # Falls das Jahr in dieser Zeile das gleiche wie zuvor ist (kein Jahreswechsel),
    jahressumme += float(messwert)   #   dann wird der Messwert zur Jahressumme hinzuaddiert.
    zaehler += 1
  else:                              # Anderenfalls wurde der erste Messwert des nächsten Jahres gelesen (Jahreswechsel).
    print( '%s %.2f' % (jahr, jahressumme/zaehler) )  # Zuerst wird das alte Jahr und der Mittelwert der Messwerte ausgegeb
    jahr = njahr                     # Dann wird die Aufsummierung der Messwerte des neuen Jahres begonnen (wie oben).
    jahressumme = float(messwert)
    zaehler = 1
# Das Ergebnis des letzten Jahres muss noch ausgegeben werden.
print( '%s %.2f' % (jahr, jahressumme/zaehler) )
```

Die durchschnittlichen Tageshöchsttemperaturen pro Jahr:

```
mundhenk@ma3: ~/Python/VL05-EinAus
mundhenk@home: python3 filter-jahr-txk.py < Jena-Wetterdaten.txt | python3 jahres-mittel.py
1824 13.50
1825 13.38
1826 13.27
1827 12.92
1828 13.07
1829 10.47
1830 12.25
1831 13.37
1832 12.42
1833 13.21
1834 14.57
1835 12.97
1836 12.83
```

Die durchschnittlichen Tageshöchsttemperaturen pro Jahr:

Zusammenfassung

Wir haben gesehen, wie Daten eingelesen und ausgegeben werden können.

Wir haben gesehen, wie eine größere Aufgabe
in kleinere und einfachere Teilaufgaben aufgeteilt werden kann.

Jede Teilaufgabe ist unabhängig von den anderen lösbar
und hat ein festes Format für die Eingabe- und Ausgabe-Daten.