



Programmieren und Problemlösen

Schleifen, Listen und Strings

Dennis Komm

Frühling 2021 – 4. März 2021

Ausgaben

Die Funktion `print()`

Ausgabe auf Bildschirm erfolgt mit `print()`

- **Standard:** Zeilenumbruch nach Ausgabe
- Zeilenumbruch kann per `end` ersetzt werden

```
print("Der Wert von x ist", end=" ")  
print(x)
```

- Mehrere Parameter ebenfalls mit Komma trennen

```
print("Der doppelte Wert von", x, "ist", 2 * x)
```

- **Standard:** Leerzeichen zwischen Strings
- Leerzeichen kann per `sep` ersetzt werden

Benutzereingaben

Benutzereingaben

Variablen

- Variablen sind „Behälter für Werte“
- Bislang wurden Werte im Programm festgelegt

```
name = "Brunhold"
print("Hallo", name)
```

- In der Praxis werden Werte meist
 - eingegeben durch Nutzerin
 - eingelesen aus Datei / Datenbank (später)

Celsius-zu-Fahrenheit-Rechner

Benutzereingabe erfolgt über Funktion `input()`

```
name = input("Geben Sie einen Namen ein: ")
print("Hallo", name)
```

Achtung

Eingabe ist String (der aus Ziffern bestehen kann) und **keine Zahl**

```
x = input("Geben Sie eine Zahl ein: ")
ausgabe = 3 * x
print(ausgabe) # String-Konkatenation statt Multiplikation
```

Celsius-zu-Fahrenheit-Rechner

Um Zahl zu erhalten, muss Eingabe mit Funktion `int()` konvertiert werden

```
x = input("Geben Sie die Temperatur in Grad Celsius ein: ")
celsius = int(x)
fahrenheit = 9 * celsius / 5 + 32
print("Die Temperatur in Grad Fahrenheit ist", fahrenheit)
```

oder kürzer...

```
celsius = int(input("Geben Sie die Temperatur in Grad Celsius ein: "))
fahrenheit = 9 * celsius / 5 + 32
print("Die Temperatur in Grad Fahrenheit ist", fahrenheit)
```

Einfache Schleifen

for-Schleifen

Wiederhole **Anweisungsblock** eine gewisse Anzahl Male

Eine **for-Schleife** besteht aus folgenden Teilen

- Im **Schleifenkopf** wird eine **Laufvariable** definiert
- Zusätzlich wird Bereich von Werten angegeben, die die Variable annimmt
- Der **Schleifenkörper** wird nacheinander mit jedem möglichen Wert der Laufvariablen ausgeführt

for-Schleifen

```
for i in range(1, 4):  
    print("Test")print("Test")  
    print(i)
```

Schleifenkopf
Schleifenkörper
(eingerückt) Laufvariable
Bereich

Achtung

Letzter Wert für *i* ist 3 und nicht 4

for-Schleifen

```
for i in range(1, 4):  
    print("Test")  
    print(i)
```

führt zu Ausführung von...

```
print("Test")  
print(1)
```

```
print("Test")  
print(2)
```

```
print("Test")  
print(3)
```

Übung – Quadratzahlen

Schreiben Sie ein Programm, das

- eine Zahl vom Benutzer einliest
- diese in einer Variablen *x* speichert
- die ersten *x* Quadratzahlen ausgibt (beginnend bei 1)



Übung – Quadratzahlen

```
x = int(input("Eingabe: "))  
  
for i in range(1, x+1):  
    print(i * i)
```

Verschachtelte Schleifen

Verschachtelte Schleifen

Programm

```
for i in range(0, 10):  
    for j in range(0, 10):  
        print(j, end=" ")  
    print()
```

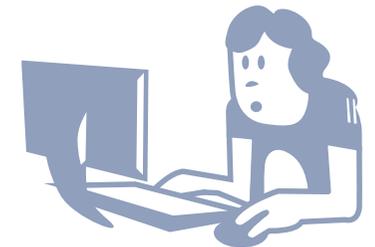
Ausgabe

```
0 1 2 3 4 5 6 7 8 9  
0 1 2 3 4 5 6 7 8 9  
0 1 2 3 4 5 6 7 8 9  
0 1 2 3 4 5 6 7 8 9  
0 1 2 3 4 5 6 7 8 9  
0 1 2 3 4 5 6 7 8 9  
0 1 2 3 4 5 6 7 8 9  
0 1 2 3 4 5 6 7 8 9  
0 1 2 3 4 5 6 7 8 9  
0 1 2 3 4 5 6 7 8 9
```

Übung – Zahlendreieck

Schreiben Sie ein Programm, das die folgende Ausgabe liefert:

```
0 1 2 3 4 5 6 7 8 9  
1 2 3 4 5 6 7 8 9  
2 3 4 5 6 7 8 9  
3 4 5 6 7 8 9  
4 5 6 7 8 9  
5 6 7 8 9  
6 7 8 9  
7 8 9  
8 9  
9
```



Übung – Zahlendreieck

```
for i in range(0, 10):
    for j in range(i, 10):
        print(j, end=" ")
    print()
```

Listen

Listen

Einfacher Zugriff auf zusammengehörige Daten

Beispiel

- `daten = [5, 1, 4, 3]`
- Jedes Element der Liste hat einen **Index**, beginnend bei 0
- Zugriff auf einzelne Elemente mit eckigen Klammern
- `daten[0] = 5`
- Länge der Liste = Anzahl Elemente
- `len(daten) = 4`

Schleifen über Listen

Alle Elemente einer Liste ausgeben

```
daten = [5, 1, 4, 3]
for item in daten:
    print(item)
```

Nimmt den Wert aller Elemente in Liste an

oder alternativ...

```
daten = [5, 1, 4, 3]
for i in range(0, len(daten)):
    print(daten[i])
```

Nimmt den Wert aller Indizes der Liste an

Schleifen über Listen – die Funktion `reversed()`

Liste rückwärts durchlaufen

```
daten = [6, 7, 5, 1]
for item in reversed(daten):
    print(item, end=" ")
```

oder alternativ...

```
for i in range(len(daten)-1, -1, -1):
    print(daten[i], end=" ")
```

Resultat

1 5 7 6

Einfache Initialisierung

Liste mit dem gleichen Wert in allen Zellen initialisieren

```
daten = [0] * 1000
print(daten)
```

```
daten = [0, 0, 0, 0] * 250
print(daten)
```

Resultat

[0, 0, 0, 0, 0, 0, 0, 0, ..., 0]

(1000 Nullen)

Die Funktion `append()`

Füge Element am Ende einer Liste ein

Beispiel

- Liste `daten`
- `daten.append(5)` fügt eine 5 am Ende ein

```
daten = [1, 4, 8]
daten.append(9)
daten.append(14)

print(daten)
```

Übung – Liste initialisieren

Schreiben Sie ein Programm, das

- eine Zahl vom Benutzer einliest
- diese in einer Variablen `x` speichert
- eine Liste mit den ersten `x` geraden Zahlen initialisiert (bei 0 beginnend)



Übung – Liste initialisieren

```
x = int(input("Eingabe: "))

daten = []

for i in range(0, x):
    daten.append(2 * i)

print(daten)
```

Listen zusammenfügen

Füge Listen zusammen mit +

Beispiel

- Liste daten
- daten + [1, 2] fügt Elemente 1 und 2 an

```
daten = [4, 6, 7]
daten2 = daten + [9]
daten2 = [1, 3] + daten2 + [12, 14]

print(daten2)
```

Zeichenketten (Strings)

Zeichenketten (Strings)

Zeichenketten sind „Listen von Zeichen“ (es gibt Unterschiede)

- Zeichen entsprechen (grösstenteils) Tasten auf der Tastatur
- Strings stehen in Anführungszeichen
- Zugriff auf einzelne Zeichen mit eckigen Klammern

- Zeichenkette `wort = "HALLO WELT"`
- `wort[0]` ist der erste Buchstabe
- `wort[1]` ist der zweite Buchstabe
- ...
- `wort[len(wort)-1]` ist der letzte Buchstabe (alternativ `wort[-1]`)

Zeichen – Die Unicode-Tabelle

0-18		19-37		38-56		57-75		76-94		95-113		114-127	
Dez.	Zeichen	Dez.	Zeichen	Dez.	Zeichen	Dez.	Zeichen	Dez.	Zeichen	Dez.	Zeichen	Dez.	Zeichen
0	NUL	19	DC3	38	&	57	9	76	L	95	-	114	r
1	SOH	20	DC4	39	'	58	:	77	M	96	_	115	s
2	STX	21	NAK	40	(59	:	78	N	97	a	116	t
3	ETX	22	SYN	41)	60	<	79	O	98	b	117	u
4	EOT	23	ETB	42	*	61	=	80	P	99	c	118	v
5	ENQ	24	CAN	43	+	62	>	81	Q	100	d	119	w
6	ACK	25	EM	44	.	63	?	82	R	101	e	120	x
7	BEL	26	SUB	45	-	64	@	83	S	102	f	121	y
8	BS	27	ESC	46	.	65	A	84	T	103	g	122	z
9	HT	28	FS	47	/	66	B	85	U	104	h	123	{
10	LF	29	GS	48	0	67	C	86	V	105	i	124	
11	VT	30	RS	49	1	68	D	87	W	106	j	125	}
12	FF	31	US	50	2	69	E	88	X	107	k	126	~
13	CR	32	SP	51	3	70	F	89	Y	108	l	127	DEL
14	SO	33	!	52	4	71	G	90	Z	109	m	...	
15	SI	34	"	53	5	72	H	91	[110	n		
16	DLE	35	#	54	6	73	I	92	\	111	o		
17	DC1	36	\$	55	7	74	J	93]	112	p		
18	DC2	37	%	56	8	75	K	94	^	113	q		

Zeichen – Die Unicode-Tabelle

Verwende Funktionen `ord()` und `chr()`

- `ord(x)` gibt Position von Zeichen `x` in Unicode-Tabelle an
- `chr(y)` gibt Zeichen an Position `y` in Unicode-Tabelle an

```
x = input("Geben Sie ein Zeichen ein: ")
print("Das Zeichen", x, "steht an Stelle", ord(x))
```

Übung – Buchstaben ausgeben

Schreiben Sie ein Programm, das

- alle 26 Grossbuchstaben ausgibt
- dazu eine `for`-Schleife verwendet

Zur Erinnerung

Der Buchstabe A steht an Stelle 65 in der Unicode-Tabelle



Übung – Buchstaben ausgeben

```
for i in range(65,91):
    print(chr(i))
```

Cäsar-Verschlüsselung



Gallia est omnis divisa in partes tres, quarum unam incolunt Belgae, aliam Aquitani, tertiam, qui ipsorum lingua Celtae, nostra Galli appellantur.

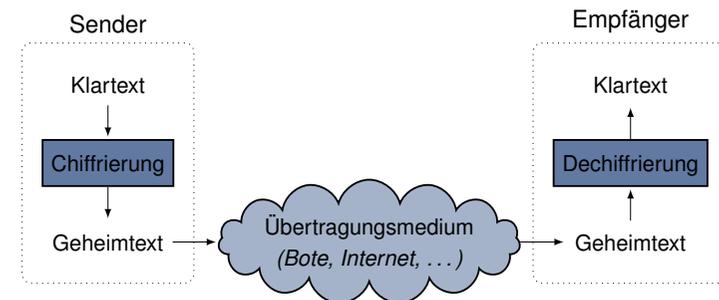
Gemeinsamer Kanal

Symmetrische Verschlüsselung

Situation

- Parteien A und B wollen über unsicheren Kanal sicher kommunizieren
 - Gemeinsamer Schlüssel k
 - A verschlüsselt Nachricht mit k
 - A sendet verschlüsselte Nachricht an B
 - B entschlüsselt Nachricht mit k
- A heisst **Sender**
- B heisst **Empfänger**
- **Symmetrisch:** Gleicher Schlüssel für Ver- und Entschlüsselung

Symmetrische Verschlüsselung



Cäsar-Verschlüsselung

Situation

- Parteien *A* und *B* wollen über unsicheren Kanal diesmal mittels Cäsar-Verschlüsselung kommunizieren
- Gemeinsamer Schlüssel *k* als Zahl zwischen 1 und 25
- *A* verschlüsselt Nachricht, indem *k* zu jedem Buchstaben addiert wird
- *A* sendet verschlüsselte Nachricht an *B*
- *B* entschlüsselt Nachricht, indem *k* von jedem Buchstaben abgezogen wird

Cäsar-Verschlüsselung

Verschiebe Buchstaben um festen Wert *k* durch Addition von *k*

Beispiel

A	B	C	D	E	F	G	H	I	J	K	L	M
W	X	Y	Z	A	B	C	D	E	F	G	H	I
N	O	P	Q	R	S	T	U	V	W	X	Y	Z
J	K	L	M	N	O	P	Q	R	S	T	U	V

- **Klartext:** HALLO WELT
- **Verschlüsselter Text:** DWHHK SAHP

Cäsar-Verschlüsselung

1. Eingegebener Buchstabe ist Unicode-Zeichen von **A** bis **Z**

A	B	...	W	X	Y	Z
65	66	...	87	88	89	90

2. Ziehe **65** ab, dann ist das Resultat zwischen **0** und **25**

A	B	...	W	X	Y	Z
0	1	...	22	23	24	25

3. Addiere nun Schlüssel (zum Beispiel **3**) und rechne **modulo 26**

A	B	...	W	X	Y	Z
3	4	...	25	0	1	2

4. Addiere nun wieder **65** zum Ergebnis

A	B	...	W	X	Y	Z
68	69	...	90	65	66	67

Division mit Rest (Modulo-Rechnen)

- Mit „%“ wird der Rest bei der ganzzahligen Division ausgegeben
- Analog wird mit “//” der Teil vor dem Komma ausgegeben

- $10 \% 3 = 1$, denn $9 = 3 \cdot 3$
- $10 \% 4 = 2$, denn $8 = 4 \cdot 2$
- $11 // 5 = 2$, denn $10 = 5 \cdot 2$
- $23 // 4 = 5$, denn $20 = 4 \cdot 5$
- $12 \% 3 = 0$, denn $12 = 4 \cdot 3$

$(10 - 3) \% 11$ ist dasselbe wie $(10 + (11 - 3)) \% 11$

Übung – Cäsar-Verschlüsselung

Schreiben Sie ein Programm, das

- einen gegebenen String durchläuft
- jeden Buchstaben mit einem Schlüssel k entschlüsselt
- dabei jeden Schlüssel k ausprobiert
- dafür folgende Formel verwendet:

$$e = (v - 65 - k) \% 26 + 65$$



Entschlüsseln Sie

TYQZCXLETVTDEVCPLETGPLCMPTE