

Informatik I: Einführung in die Programmierung

3. Werte, Typen, Variablen und Ausdrücke

Albert-Ludwigs-Universität Freiburg



UNI
FREIBURG

Peter Thiemann

29. Oktober 2019



Exkurs: Datenrepräsentation



- Der Computer repräsentiert Daten als Folgen von **Bits**.
- Ein Bit (*binary digit*) ist die kleinste Informationseinheit. Sein Wert ist entweder 0 oder 1.
- Einfache technische Realisierung durch Schalter ein / Schalter aus bzw. Ladung vorhanden / nicht vorhanden.

Exkurs:
Datenrepräsentation

Werte und Typen

Variable

Ausdrücke



■ **Logisches Und:** $b_1 \wedge b_2$

Ergebnis ist 1, falls $b_1 = 1$ **und** $b_2 = 1$, sonst 0.

$$1 \wedge 1 = 1, \quad 1 \wedge 0 = 0, \quad 0 \wedge 1 = 0, \quad 0 \wedge 0 = 0$$

■ **Logisches Oder:** $b_1 \vee b_2$

Ergebnis ist 1, falls $b_1 = 1$ **oder** $b_2 = 1$, sonst 0.

$$1 \vee 1 = 1, \quad 1 \vee 0 = 1, \quad 0 \vee 1 = 1, \quad 0 \vee 0 = 0$$

■ **Nicht, Negation, Komplement:** $\neg b$

Ergebnis ist 1, falls $b = 0$. Ergebnis ist 0, falls $b = 1$.

$$\neg 1 = 0, \quad \neg 0 = 1$$

- Mit diesen drei Grundoperationen können **alle** möglichen Operationen auf Bits definiert werden.

Exkurs:
Datenrepräsentation

Werte und Typen

Variable

Ausdrücke

Beispiel



Jede Operation auf zwei Bits b_1 und b_2 kann durch ihre Wertetabelle (vier Bit) angegeben werden

b_1	b_2	$f(b_1, b_2)$	f_8	f_{11}
0	0		0	1
0	1		0	1
1	0		0	0
1	1		1	1

Exkurs:
Datenrepräsentation

Werte und Typen

Variable

Ausdrücke



Jede Operation auf zwei Bits b_1 und b_2 kann durch ihre Wertetabelle (vier Bit) angegeben werden

b_1	b_2	$f(b_1, b_2)$	f_8	f_{11}
0	0		0	1
0	1		0	1
1	0		0	0
1	1		1	1

Example (Auflösung)

$$f_8(b_1, b_2) = b_1 \wedge b_2$$

Exkurs:
Datenrepräsentation

Werte und Typen

Variable

Ausdrücke



Jede Operation auf zwei Bits b_1 und b_2 kann durch ihre Wertetabelle (vier Bit) angegeben werden

b_1	b_2	$f(b_1, b_2)$	f_8	f_{11}
0	0		0	1
0	1		0	1
1	0		0	0
1	1		1	1

Example (Auflösung)

$$f_8(b_1, b_2) = b_1 \wedge b_2$$

Example (Auflösung)

$$f_{11}(b_1, b_2) = (b_1 \wedge b_2) \vee \neg b_1 = \neg b_1 \vee b_2$$

Exkurs:
Datenrepräsentation

Werte und Typen

Variable

Ausdrücke



- Rechnen mit einem Bit ist nicht effizient.
- Die meisten Computer rechnen daher mit Bitvektoren der Breite 8 (ein **Byte**, auch **Octet**), 16, 32 oder 64.
- Letztere heißen auch 16-Bit (bzw. 32-Bit, 64-Bit) **Worte**.
- Der Aufbau des Computers (genauer gesagt, des Prozessors) ist auf eine Wortbreite ausgerichtet, die durch Bezeichnungen wie 32-Bit-Architektur bzw. 64-Bit-Architektur zum Ausdruck kommt.

Grundoperationen auf Worten

Bitweise logische Operationen



- Definiert auf Worten gleicher Breite.
- Wendet die Bit-Operationen auf die entsprechenden Positionen der Argumente an.
- **Logisches Und:** $w_1 \wedge w_2$
Beispiel: $1100 \wedge 1010 = 1000$
- **Logisches Oder:** $w_1 \vee w_2$
Beispiel: $1100 \vee 1010 = 1110$
- **Negation:** $\neg w$
Beispiel: $\neg 10 = 01$

Exkurs:
Datenrepräsentation

Werte und Typen

Variable

Ausdrücke



Mensch: Dezimalsystem

- **Stellenwertsystem** mit Basis 10
- Jede Stelle in der **Dezimaldarstellung** einer Zahl entspricht einer 10er-Potenz
- Beginnend von rechts mit 10^0
- Zehn Ziffern notwendig: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9

Exkurs:
Datenrepräsentation

Werte und
Typen

Variable

Ausdrücke



Mensch: Dezimalsystem

- **Stellenwertsystem** mit Basis 10
- Jede Stelle in der **Dezimaldarstellung** einer Zahl entspricht einer 10er-Potenz
- Beginnend von rechts mit 10^0
- Zehn Ziffern notwendig: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9

Beispiel

$$\begin{aligned}\underline{4711}_{10} &= 4 * 10^3 + 7 * 10^2 + 1 * 10^1 + 1 * 10^0 \\ &= 4000 + 700 + 10 + 1 \\ &= 4711\end{aligned}$$

Exkurs:
Datenrepräsentation

Werte und
Typen

Variable

Ausdrücke



Computer: Dual- oder Binärsystem

- Stellenwertsystem mit Basis **2**
- Jede Stelle in der **Binärdarstellung** einer Zahl entspricht einer **2er-Potenz**
- Beginnend von rechts mit **2^0**
- **Zwei** Ziffern notwendig: 0, 1 — **ein Bit!**

Exkurs:
Datenrepräsentation

Werte und
Typen

Variable

Ausdrücke



Computer: Dual- oder Binärsystem

- Stellenwertsystem mit Basis **2**
- Jede Stelle in der **Binärdarstellung** einer Zahl entspricht einer **2er-Potenz**
- Beginnend von rechts mit **2^0**
- **Zwei** Ziffern notwendig: 0, 1 — **ein Bit!**

Beispiel

$$\begin{aligned} \underline{101010}_2 &= 1 * 2^5 + 0 * 2^4 + 1 * 2^3 + 0 * 2^2 + 1 * 2^1 + 0 * 2^0 \\ &= 32 + 0 + 8 + 0 + 2 + 0 \\ &= 42 \end{aligned}$$

Exkurs:
Datenrepräsentation

Werte und
Typen

Variable

Ausdrücke



Programmierer: Hexadezimalsystem

- Stellenwertsystem mit Basis **16** (4 Bit pro Stelle)
- Jede Stelle in der **Hexadezimaldarstellung** einer Zahl entspricht einer **16er-Potenz**
- Beginnend von rechts mit **16⁰**
- **16** Ziffern: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, a, b, c, d, e, f

Exkurs:
Datenrepräsentation

Werte und
Typen

Variable

Ausdrücke



Programmierer: Hexadezimalsystem

- Stellenwertsystem mit Basis **16** (4 Bit pro Stelle)
- Jede Stelle in der **Hexadezimaldarstellung** einer Zahl entspricht einer **16er-Potenz**
- Beginnend von rechts mit **16^0**
- **16** Ziffern: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, a, b, c, d, e, f

Beispiel

$$\begin{aligned}\underline{beef}_{16} &= 11 * 16^3 + 14 * 16^2 + 14 * 16^1 + 15 * 16^0 \\ &= 11 * 4096 + 14 * 256 + 14 * 16 + 15 \\ &= 48879\end{aligned}$$

Exkurs:
Datenrepräsentation

Werte und Typen

Variable

Ausdrücke



- Welche natürlichen Zahlen lassen sich mit gegebener Wortbreite darstellen?

Wortbreite	Wertebereich
1	0...1
2	0...3
4	0...15
8	0...255
16	0...65.535
32	0...4.294.967.295
64	0...18.446.744.073.709.551.615
n	0... $2^n - 1$

Exkurs:
Datenrepräsentation

Werte und Typen

Variable

Ausdrücke



- Gegeben: natürliche Zahl n
- Gesucht: Darstellung von n im Stellenwertsystem mit Basis $B \geq 2$
- Verwende als Ziffern $0, 1, \dots, B-1$
- Schreibe von **rechts nach links** in die Ausgabe

Exkurs:
Datenrepräsentation

Werte und
Typen

Variable

Ausdrücke

Algorithmus

- 1 Berechne q und r als Quotient und Divisionsrest von n/B .
- 2 Schreibe den Rest r links an die Ausgabe.
- 3 Falls $q \neq 0$, weiter bei Punkt 1 mit $n \leftarrow q$.
- 4 Sonst Ende.

Beispiel: Darstellung in Basis B



Bestimme die Binärdarstellung ($B = 2$) von $n = 42$.

- $42/2 = 21$ Rest **0**
- $21/2 = 10$ Rest **1**
- $10/2 = 5$ Rest **0**
- $5/2 = 2$ Rest **1**
- $2/2 = 1$ Rest **0**
- $1/2 = 0$ Rest **1**
- Fertig, weil $q = 0$.
- Ergebnis 101010₂
- von unten nach oben abgelesen

Exkurs:
Datenreprä-
sentation

Werte und
Typen

Variable

Ausdrücke

Addieren von Zahlen in Binärdarstellung



- Wortbreite 1: $0+0=0$; $0+1=1$; $1+0=1$; $1+1=?$

Exkurs:
Datenreprä-
sentation

Werte und
Typen

Variable

Ausdrücke

Addieren von Zahlen in Binärdarstellung



- Wortbreite 1: $0+0=0$; $0+1=1$; $1+0=1$; $1+1=?$
- $1+1=0$ mit Übertrag 1

Exkurs:
Datenreprä-
sentation

Werte und
Typen

Variable

Ausdrücke



- Wortbreite 1: $0+0=0$; $0+1=1$; $1+0=1$; $1+1=?$
- $1+1=0$ mit Übertrag 1
- Damit weiter wie schriftliche Addition

Exkurs:
Datenrepräsentation

Werte und
Typen

Variable

Ausdrücke



- Wortbreite 1: $0+0=0$; $0+1=1$; $1+0=1$; $1+1=?$
- $1+1=0$ mit **Übertrag 1**
- Damit weiter wie schriftliche Addition
- Beispiel: $42 + 6$ (in Binärdarstellung: 101010_2 und 110_2)

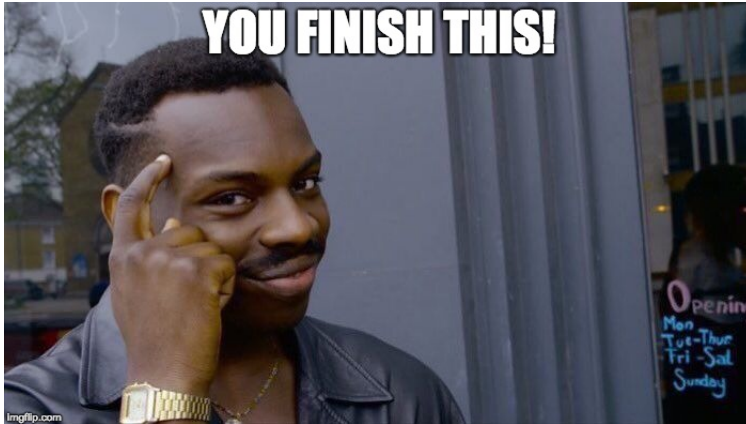
$$\begin{array}{r} 1\ 0\ 1\ 0\ 1\ 0 \\ +\ 0\ 0\ 0\ 1\ 1\ 0 \\ \hline 0 \\ 0 \end{array}$$

Exkurs:
Datenrepräsentation

Werte und
Typen

Variable

Ausdrücke



Exkurs:
Datenrepräsentation

Werte und Typen

Variable

Ausdrücke



Rechnerarithmetik

- Darstellung negativer Zahlen
- Subtraktion
- Multiplikation
- Division
- und Schaltkreise dafür

Exkurs:
Datenreprä-
sentation

Werte und
Typen

Variable

Ausdrücke



Rechnerarithmetik

- Darstellung negativer Zahlen
- Subtraktion
- Multiplikation
- Division
- und Schaltkreise dafür

Zum Nachdenken

Definiere die Addition von Bits mit Hilfe der Grundoperationen.

Exkurs:
Datenrepräsentation

Werte und
Typen

Variable

Ausdrücke



Werte und Typen

Exkurs:
Datenreprä-
sentation

Werte und
Typen

Variable

Ausdrücke



- Ein Datentyp besteht aus einer Menge von **Werten** und **Operationen** auf diesen Werten (**Semantik**).
- **Literale** sind die **Darstellung** (als **Zeichenkette**) von Werten des Datentyps (**Syntax**).



- Die ganze Zahl *16* als Wert wird z.B. durch das Literal `16` dargestellt, aber auch durch `0x10` (hexadezimale Darstellung) und `0b10000` (binäre Darstellung).
- Die Zeichenkette (der String) *'Hallo'* als Wert wird durch die Literale `'Hallo'`, `"Hallo"` und `'''Hallo'''` dargestellt.
- *200.0* wird durch `200.0` dargestellt, aber auch durch `2.0e+2` (Exponentendarstellung $2.0 * 10^2$).

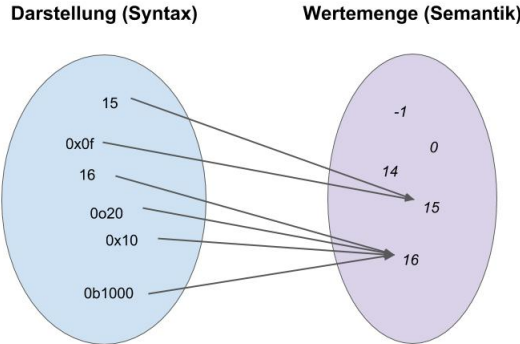
Exkurs:
Datenreprä-
sentation

Werte und
Typen

Variable

Ausdrücke

Beispiel: int



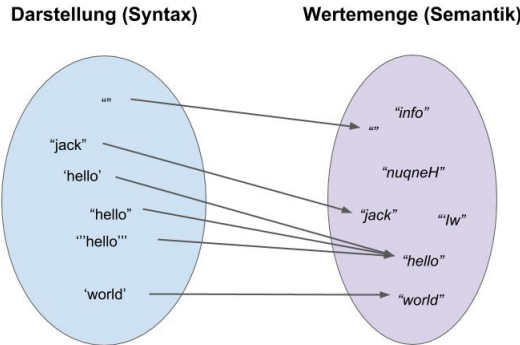
Exkurs:
Datenreprä-
sentation

Werte und
Typen

Variable

Ausdrücke

Beispiel: string



Exkurs:
Datenreprä-
sentation

Werte und
Typen

Variablen

Ausdrücke



- In Python besteht jeder Wert aus zwei Teilen:

Typ	Interne Repräsentation des Wertes
-----	-----------------------------------

- Die interne Repräsentation ist ein Bitmuster im Speicher, das entsprechend des Typs interpretiert wird.
- Beispiele

16	↔	int	<i>0x10</i>
2.24E44	↔	float	<i>0x10</i>
3.14159	↔	float	<i>0x40490fd0</i>
1078530000	↔	int	<i>0x40490fd0</i>
"hello"	↔	string	<i>0x68656c6c6f00</i>



Variable

Exkurs:
Datenreprä-
sentation

Werte und
Typen

Variable

Ausdrücke



- Werte können mit einem Namen (**Variablennamen**, **Bezeichner**, **Identifizier**) versehen werden.

Exkurs:
Datenreprä-
sentation

Werte und
Typen

Variable

Ausdrücke



- Werte können mit einem Namen (**Variablennamen**, **Bezeichner**, **Identifizier**) versehen werden.
- Das geschieht durch eine **Zuweisung**.

Exkurs:
Datenreprä-
sentation

Werte und
Typen

Variable

Ausdrücke



- Werte können mit einem Namen (**Variablennamen**, **Bezeichner**, **Identifizier**) versehen werden.
- Das geschieht durch eine **Zuweisung**.
- Dazu wird der Bezeichner auf der linken Seite und ein Ausdruck auf der rechten Seite eines Gleichheitszeichens geschrieben.

Exkurs:
Datenrepräsentation

Werte und Typen

Variable

Ausdrücke



- Werte können mit einem Namen (**Variablennamen**, **Bezeichner**, **Identifizier**) versehen werden.
- Das geschieht durch eine **Zuweisung**.
- Dazu wird der Bezeichner auf der linken Seite und ein Ausdruck auf der rechten Seite eines Gleichheitszeichens geschrieben.

Exkurs:
Datenrepräsentation

Werte und Typen

Variable

Ausdrücke

Python-Interpreter

```
>>> spam = 111
>>>
```



- Werte können mit einem Namen (**Variablennamen**, **Bezeichner**, **Identifizier**) versehen werden.
- Das geschieht durch eine **Zuweisung**.
- Dazu wird der Bezeichner auf der linken Seite und ein Ausdruck auf der rechten Seite eines Gleichheitszeichens geschrieben.

Exkurs:
Datenrepräsentation

Werte und
Typen

Variable

Ausdrücke

Python-Interpreter

```
>>> spam = 111
>>>
```



- Werte können mit einem Namen (**Variablennamen**, **Bezeichner**, **Identifizier**) versehen werden.
- Das geschieht durch eine **Zuweisung**.
- Dazu wird der Bezeichner auf der linken Seite und ein Ausdruck auf der rechten Seite eines Gleichheitszeichens geschrieben.

Exkurs:
Datenrepräsentation

Werte und
Typen

Variable

Ausdrücke

Python-Interpreter

```
>>> spam = 111
>>> spam
```



- Werte können mit einem Namen (**Variablennamen**, **Bezeichner**, **Identifier**) versehen werden.
- Das geschieht durch eine **Zuweisung**.
- Dazu wird der Bezeichner auf der linken Seite und ein Ausdruck auf der rechten Seite eines Gleichheitszeichens geschrieben.

Exkurs:
Datenrepräsentation

Werte und Typen

Variable

Ausdrücke

Python-Interpreter

```
>>> spam = 111
>>> spam
111
```

- “Die *Variable* `spam` erhält den *Wert* von 111.”



- Werte können mit einem Namen (**Variablennamen**, **Bezeichner**, **Identifier**) versehen werden.
- Das geschieht durch eine **Zuweisung**.
- Dazu wird der Bezeichner auf der linken Seite und ein Ausdruck auf der rechten Seite eines Gleichheitszeichens geschrieben.

Exkurs:
Datenrepräsentation

Werte und Typen

Variable

Ausdrücke

Python-Interpreter

```
>>> spam = 111
>>> spam
111
```

- “Die *Variable* `spam` erhält den *Wert* von 111.”
- Eine Zuweisung ist kein Ausdruck, sondern eine **Anweisung** und **hat keinen Wert!**



- Der **Zustand** eines Programms wird durch die **Belegung der Variablen** mit Werten und den aktuellen Ausführungspunkt beschrieben werden.

Python-Interpreter

```
>>> spam = 123
>>> egg = 'spam'
```

- Variablenbelegung nach der Ausführung:

Global frame

spam	123
egg	"spam"

Exkurs:
Datenrepräsentation

Werte und
Typen

Variable

Ausdrücke



- Ein Bezeichner besteht aus Buchstaben, Unterstrichen und Ziffern. Das erste Zeichen darf keine Ziffer sein.

Python-Interpreter

```
>>> Brägele = 1  
>>>
```

Exkurs:
Datenrepräsentation

Werte und
Typen

Variable

Ausdrücke



- Ein Bezeichner besteht aus Buchstaben, Unterstrichen und Ziffern. Das erste Zeichen darf keine Ziffer sein.

Python-Interpreter

```
>>> Brägele = 1
>>> Kaltes Wasser = 2
```

Exkurs:
Datenreprä-
sentation

Werte und
Typen

Variable

Ausdrücke



- Ein Bezeichner besteht aus Buchstaben, Unterstrichen und Ziffern. Das erste Zeichen darf keine Ziffer sein.

Python-Interpreter

```
>>> Brägele = 1
>>> Kaltes Wasser = 2
File "<stdin>", line 1
    Kaltes Wasser = 2
                   ^
SyntaxError: invalid syntax
>>>
```

Exkurs:
Datenrepräsentation

Werte und
Typen

Variable

Ausdrücke



- Ein Bezeichner besteht aus Buchstaben, Unterstrichen und Ziffern. Das erste Zeichen darf keine Ziffer sein.

Python-Interpreter

```
>>> Brägele = 1
>>> Kaltes Wasser = 2
File "<stdin>", line 1
    Kaltes Wasser = 2
                    ^
SyntaxError: invalid syntax
>>> 2you = 3
```

Exkurs:
Datenreprä-
sentation

Werte und
Typen

Variable

Ausdrücke



- Ein Bezeichner besteht aus Buchstaben, Unterstrichen und Ziffern. Das erste Zeichen darf keine Ziffer sein.

Python-Interpreter

```
>>> Brägele = 1
>>> Kaltes Wasser = 2
File "<stdin>", line 1
    Kaltes Wasser = 2
                    ^
SyntaxError: invalid syntax
>>> 2you = 3
File "<stdin>", line 1
    2you = 3
        ^
SyntaxError: invalid syntax
```

Exkurs:
Datenrepräsentation

Werte und
Typen

Variable

Ausdrücke



Python-Interpreter

```
>>> class = 'Theory'
```

Exkurs:
Datenreprä-
sentation

Werte und
Typen

Variable

Ausdrücke



Python-Interpreter

```
>>> class = 'Theory'
      File "<stdin>", line 1
        class = 'Theory'
            ^
SyntaxError: invalid syntax
```

Folgende **Schlüsselwörter** können nicht als Bezeichner benutzt werden:

False	class	finally	is	return
None	continue	for	lambda	try
True	def	from	nonlocal	while
and	del	global	not	with
as	elif	if	or	yield
assert	else	import	pass	
break	except	in	raise	

Exkurs:
Datenrepräsentation

Werte und
Typen

Variable

Ausdrücke



- Variablen sind erst verwendbar, nachdem ihnen ein Wert zugewiesen wurde.

Python-Interpreter

```
>>> spam = 3
>>>
```

Exkurs:
Datenrepräsentation

Werte und
Typen

Variable

Ausdrücke



- Variablen sind erst verwendbar, nachdem ihnen ein Wert zugewiesen wurde.

Python-Interpreter

```
>>> spam = 3
>>> spam
```

Exkurs:
Datenrepräsentation

Werte und
Typen

Variable

Ausdrücke



- Variablen sind erst verwendbar, nachdem ihnen ein Wert zugewiesen wurde.

Python-Interpreter

```
>>> spam = 3
>>> spam
3
>>>
```

Exkurs:
Datenrepräsentation

Werte und
Typen

Variable

Ausdrücke



- Variablen sind erst verwendbar, nachdem ihnen ein Wert zugewiesen wurde.

Python-Interpreter

```
>>> spam = 3
>>> spam
3
>>> egg
```

Exkurs:
Datenrepräsentation

Werte und
Typen

Variable

Ausdrücke



- Variablen sind erst verwendbar, nachdem ihnen ein Wert zugewiesen wurde.

Python-Interpreter

```
>>> spam = 3
>>> spam
3
>>> egg
Traceback (most recent call last): ...
NameError: name 'egg' is not defined
>>>
```

Exkurs:
Datenrepräsentation

Werte und
Typen

Variable

Ausdrücke



- Variablen sind erst verwendbar, nachdem ihnen ein Wert zugewiesen wurde.

Python-Interpreter

```
>>> spam = 3
>>> spam
3
>>> egg
Traceback (most recent call last): ...
NameError: name 'egg' is not defined
>>> Spam
```

Exkurs:
Datenrepräsentation

Werte und
Typen

Variable

Ausdrücke



- Variablen sind erst verwendbar, nachdem ihnen ein Wert zugewiesen wurde.

Python-Interpreter

```
>>> spam = 3
>>> spam
3
>>> egg
Traceback (most recent call last): ...
NameError: name 'egg' is not defined
>>> Spam
Traceback (most recent call last): ...
NameError: name 'Spam' is not defined
```

Exkurs:
Datenrepräsentation

Werte und
Typen

Variable

Ausdrücke



Ausdrücke

Exkurs:
Datenreprä-
sentation

Werte und
Typen

Variable

Ausdrücke



- Wir kennen bereits **Operatoren** auf Zahlen: $+$, $-$, $*$, \dots
- **Ausdrücke** werden aus Operatoren, Literalen, Variablen und weiteren Formen zusammengesetzt.
- Die **Auswertung eines Ausdrucks** liefert entweder einen (Typ-) Fehler oder einen Wert.

Exkurs:
Datenreprä-
sentation

Werte und
Typen

Variable

Ausdrücke



- Die **Auswertung eines Ausdrucks** beginnt bei den Literalen und Variablen und wendet dann die Operatoren auf die Werte der Teilausdrücke an.
- Bei arithmetischen Ausdrücken gelten die üblichen **Präzedenzregeln**:
 - immer die Klammerung zuerst beachtend,

Exkurs:
Datenrepräsentation

Werte und
Typen

Variable

Ausdrücke



- Die **Auswertung eines Ausdrucks** beginnt bei den Literalen und Variablen und wendet dann die Operatoren auf die Werte der Teilausdrücke an.
- Bei arithmetischen Ausdrücken gelten die üblichen **Präzedenzregeln**:
 - immer die Klammerung zuerst beachtend,
 - dann die Exponentiation auswertend,

Exkurs:
Datenreprä-
sentation

Werte und
Typen

Variable

Ausdrücke



- Die **Auswertung eines Ausdrucks** beginnt bei den Literalen und Variablen und wendet dann die Operatoren auf die Werte der Teilausdrücke an.
- Bei arithmetischen Ausdrücken gelten die üblichen **Präzedenzregeln**:
 - immer die Klammerung zuerst beachtend,
 - dann die Exponentiation auswertend,
 - danach Multiplikation und Division,

Exkurs:
Datenrepräsentation

Werte und
Typen

Variable

Ausdrücke



- Die **Auswertung eines Ausdrucks** beginnt bei den Literalen und Variablen und wendet dann die Operatoren auf die Werte der Teilausdrücke an.
- Bei arithmetischen Ausdrücken gelten die üblichen **Präzedenzregeln**:
 - immer die Klammerung zuerst beachtend,
 - dann die Exponentiation auswertend,
 - danach Multiplikation und Division,
 - dann Addition und Subtraktion,

Exkurs:
Datenreprä-
sentation

Werte und
Typen

Variable

Ausdrücke



- Die **Auswertung eines Ausdrucks** beginnt bei den Literalen und Variablen und wendet dann die Operatoren auf die Werte der Teilausdrücke an.
- Bei arithmetischen Ausdrücken gelten die üblichen **Präzedenzregeln**:
 - immer die Klammerung zuerst beachtend,
 - dann die Exponentiation auswertend,
 - danach Multiplikation und Division,
 - dann Addition und Subtraktion,
 - bei gleicher Präzedenz wird von links nach rechts ausgewertet; Ausnahme: Exponentiation von rechts nach links



Python-Interpreter

```
>>> spam = 3
>>>
```

Exkurs:
Datenrepräsentation

Werte und
Typen

Variable

Ausdrücke



Python-Interpreter

```
>>> spam = 3
>>> 3*1**spam
```

Exkurs:
Datenreprä-
sentation

Werte und
Typen

Variable

Ausdrücke



Python-Interpreter

```
>>> spam = 3
>>> 3*1**spam
3
>>>
```

Exkurs:
Datenreprä-
sentation

Werte und
Typen

Variable

Ausdrücke



Python-Interpreter

```
>>> spam = 3
>>> 3*1**spam
3
>>> (3*1)**spam
```

Exkurs:
Datenreprä-
sentation

Werte und
Typen

Variable

Ausdrücke



Python-Interpreter

```
>>> spam = 3
>>> 3*1**spam
3
>>> (3*1)**spam
27
>>>
```

Exkurs:
Datenreprä-
sentation

Werte und
Typen

Variable

Ausdrücke



Python-Interpreter

```
>>> spam = 3
>>> 3*1**spam
3
>>> (3*1)**spam
27
>>> 2*spam-1//2
```

Exkurs:
Datenreprä-
sentation

Werte und
Typen

Variable

Ausdrücke



Python-Interpreter

```
>>> spam = 3
>>> 3*1**spam
3
>>> (3*1)**spam
27
>>> 2*spam-1//2
6
>>>
```

Exkurs:
Datenreprä-
sentation

Werte und
Typen

Variable

Ausdrücke



Python-Interpreter

```
>>> spam = 3
>>> 3*1**spam
3
>>> (3*1)**spam
27
>>> 2*spam-1//2
6
>>> spam ** spam ** spam
```

Exkurs:
Datenreprä-
sentation

Werte und
Typen

Variable

Ausdrücke



Python-Interpreter

```
>>> spam = 3
>>> 3*1**spam
3
>>> (3*1)**spam
27
>>> 2*spam-1//2
6
>>> spam ** spam ** spam
7625597484987
```

Exkurs:
Datenreprä-
sentation

Werte und
Typen

Variable

Ausdrücke



- Strings verketteten mit dem Operator '+' (**Konkatenation**)

Python-Interpreter

```
>>> 'spam' + 'egg'
```

Exkurs:
Datenrepräsentation

Werte und
Typen

Variable

Ausdrücke



- Strings verketteten mit dem Operator '+' (**Konkatenation**)

Python-Interpreter

```
>>> 'spam' + 'egg'  
'spamegg'
```

- Strings mit ganzen Zahlen multiplizieren (Python)

Python-Interpreter

```
>>> 3 * 'spam'
```

Exkurs:
Datenrepräsentation

Werte und
Typen

Variable

Ausdrücke



- Strings verketteten mit dem Operator '+' (**Konkatenation**)

Python-Interpreter

```
>>> 'spam' + 'egg'  
'spamegg'
```

- Strings mit ganzen Zahlen multiplizieren (Python)

Python-Interpreter

```
>>> 3 * 'spam'  
'spamspamspam'  
>>>
```

Exkurs:
Datenrepräsentation

Werte und
Typen

Variable

Ausdrücke



- Strings verketteten mit dem Operator '+' (**Konkatenation**)

Python-Interpreter

```
>>> 'spam' + 'egg'  
'spamegg'
```

- Strings mit ganzen Zahlen multiplizieren (Python)

Python-Interpreter

```
>>> 3 * 'spam'  
'spamspamspam'  
>>> 0 * 'spam'
```

Exkurs:
Datenreprä-
sentation

Werte und
Typen

Variable

Ausdrücke



- Strings verketteten mit dem Operator '+' (**Konkatenation**)

Python-Interpreter

```
>>> 'spam' + 'egg'  
'spamegg'
```

- Strings mit ganzen Zahlen multiplizieren (Python)

Python-Interpreter

```
>>> 3 * 'spam'  
'spamspamspam'  
>>> 0 * 'spam'  
''  
>>>
```

Exkurs:
Datenrepräsentation

Werte und
Typen

Variable

Ausdrücke



- Strings verketteten mit dem Operator '+' (**Konkatenation**)

Python-Interpreter

```
>>> 'spam' + 'egg'  
'spamegg'
```

- Strings mit ganzen Zahlen multiplizieren (Python)

Python-Interpreter

```
>>> 3 * 'spam'  
'spamspamspam'  
>>> 0 * 'spam'  
''  
>>> -2 * 'spam'
```

Exkurs:
Datenrepräsentation

Werte und
Typen

Variable

Ausdrücke



- Strings verketteten mit dem Operator '+' (**Konkatenation**)

Python-Interpreter

```
>>> 'spam' + 'egg'  
'spamegg'
```

- Strings mit ganzen Zahlen multiplizieren (Python)

Python-Interpreter

```
>>> 3 * 'spam'  
'spamspamspam'  
>>> 0 * 'spam'  
''  
>>> -2 * 'spam'  
''
```

Exkurs:
Datenrepräsentation

Werte und
Typen

Variable

Ausdrücke



- Auf der rechten Seite einer Zuweisung dürfen Ausdrücke auftreten:

Python-Interpreter

```
>>> spam = 42
```

Exkurs:
Datenreprä-
sentation

Werte und
Typen

Variable

Ausdrücke



- Auf der rechten Seite einer Zuweisung dürfen Ausdrücke auftreten:

Python-Interpreter

```
>>> spam = 42
>>>
```

Exkurs:
Datenreprä-
sentation

Werte und
Typen

Variable

Ausdrücke



- Auf der rechten Seite einer Zuweisung dürfen Ausdrücke auftreten:

Python-Interpreter

```
>>> spam = 42
>>> egg = spam//7
>>>
```

Exkurs:
Datenreprä-
sentation

Werte und
Typen

Variable

Ausdrücke



- Auf der rechten Seite einer Zuweisung dürfen Ausdrücke auftreten:

Python-Interpreter

```
>>> spam = 42
>>> egg = spam//7
>>> egg
```

Exkurs:
Datenreprä-
sentation

Werte und
Typen

Variable

Ausdrücke



- Auf der rechten Seite einer Zuweisung dürfen Ausdrücke auftreten:

Python-Interpreter

```
>>> spam = 42
>>> egg = spam//7
>>> egg
6
```

- Es wird immer erst der Wert der rechten Seite bestimmt, dann an die Variable zugewiesen:

Python-Interpreter

```
>>> spam = 42
>>>
```

Exkurs:
Datenreprä-
sentation

Werte und
Typen

Variable

Ausdrücke



- Auf der rechten Seite einer Zuweisung dürfen Ausdrücke auftreten:

Python-Interpreter

```
>>> spam = 42
>>> egg = spam//7
>>> egg
6
```

- Es wird immer erst der Wert der rechten Seite bestimmt, dann an die Variable zugewiesen:

Python-Interpreter

```
>>> spam = 42
>>> spam = spam * 2
>>>
```

Exkurs:
Datenrepräsentation

Werte und
Typen

Variable

Ausdrücke



- Auf der rechten Seite einer Zuweisung dürfen Ausdrücke auftreten:

Python-Interpreter

```
>>> spam = 42
>>> egg = spam//7
>>> egg
6
```

- Es wird immer erst der Wert der rechten Seite bestimmt, dann an die Variable zugewiesen:

Python-Interpreter

```
>>> spam = 42
>>> spam = spam * 2
>>> spam
```

Exkurs:
Datenrepräsentation

Werte und
Typen

Variable

Ausdrücke



- Auf der rechten Seite einer Zuweisung dürfen Ausdrücke auftreten:

Python-Interpreter

```
>>> spam = 42
>>> egg = spam//7
>>> egg
6
```

- Es wird immer erst der Wert der rechten Seite bestimmt, dann an die Variable zugewiesen:

Python-Interpreter

```
>>> spam = 42
>>> spam = spam * 2
>>> spam
84
```

Exkurs:
Datenrepräsentation

Werte und Typen

Variable

Ausdrücke



- Ein **Datentyp** besteht aus einer Menge von **Werten** und **Operationen** auf diesen Werten (**Semantik**).
- Ein **Literal** ist die **Darstellung** (als **Zeichenkette**) eines Werts (**Syntax**).
- Jeder Wert hat einen bestimmten **Typ**.
- Werte erhalten durch **Zuweisung** einen Namen (**Variable**).
- Der Wert einer Variablen kann sich ändern.
- Bei einer Zuweisung wird immer erst die rechte Seite ausgewertet, dann wird der Wert zugewiesen!
- **Ausdrücke** werden aus Operatoren, Literalen und Variablen gebildet.
- Sie haben einen Wert!
- Zuweisungen sind **Anweisungen**; sie haben keinen Wert!

Exkurs:
Datenrepräsentation

Werte und Typen

Variable

Ausdrücke