

# Informatik I: Einführung in die Programmierung

## Werte, Typen, Variablen und Ausdrücke

Albert-Ludwigs-Universität Freiburg



**UNI  
FREIBURG**

Bernhard Nebel

21. Oktober 2016



# Werte und Typen



- **Werte** und ihre **Darstellung** (*Literale*) gehören zu den Basiskomponenten von Programmen.
- Die Zeichenkette (der String) *'Hallo'* als Wert wird durch die Literale `'Hallo'`, `"Hallo"` und `'''Hallo'''` dargestellt.
- Die ganze Zahl *16* als Wert wird z.B. durch das Literal `16` dargestellt, aber auch durch `0x10` (hexadezimale Darstellung), `0b10000` (binäre Darstellung) und `0o20` (oktale Darstellung).
- *200.0* wird durch `200.0` dargestellt, aber auch durch `2.0e+2` (Exponentendarstellung  $2.0 * 10^2$ ).

Werte und  
Typen

Variablen

Ausdrücke



- Jeder Wert gehört zu (genau) einem **Typ**
- Mithilfe der Funktion `type` kann man den Typ eines Wertes bzw. des Literals erfahren:

Werte und  
Typen

Variablen

Ausdrücke

## Python-Interpreter

```
>>> type('hello world')
```

- Jeder Wert gehört zu (genau) einem **Typ**
- Mithilfe der Funktion `type` kann man den Typ eines Wertes bzw. des Literals erfahren:

Werte und  
Typen

Variablen

Ausdrücke

## Python-Interpreter

```
>>> type('hello world')  
<class 'str'>  
>>>
```



- Jeder Wert gehört zu (genau) einem **Typ**
- Mithilfe der Funktion `type` kann man den Typ eines Wertes bzw. des Literals erfahren:

Werte und  
Typen

Variablen

Ausdrücke

## Python-Interpreter

```
>>> type('hello world')  
<class 'str'>  
>>> type("hello world")
```



- Jeder Wert gehört zu (genau) einem **Typ**
- Mithilfe der Funktion `type` kann man den Typ eines Wertes bzw. des Literals erfahren:

Werte und  
Typen

Variablen

Ausdrücke

## Python-Interpreter

```
>>> type('hello world')
<class 'str'>
>>> type("hello world")
<class 'str'>
```



- Jeder Wert gehört zu (genau) einem **Typ**
- Mithilfe der Funktion `type` kann man den Typ eines Wertes bzw. des Literals erfahren:

Werte und  
Typen

Variablen

Ausdrücke

## Python-Interpreter

```
>>> type('hello world')
<class 'str'>
>>> type("hello world")
<class 'str'>
>>>
```



- Jeder Wert gehört zu (genau) einem **Typ**
- Mithilfe der Funktion `type` kann man den Typ eines Wertes bzw. des Literals erfahren:

Werte und  
Typen

Variablen

Ausdrücke

## Python-Interpreter

```
>>> type('hello world')
<class 'str'>
>>> type("hello world")
<class 'str'>
>>> type(3.14)
```

- Jeder Wert gehört zu (genau) einem **Typ**
- Mithilfe der Funktion `type` kann man den Typ eines Wertes bzw. des Literals erfahren:

Werte und  
Typen

Variablen

Ausdrücke

## Python-Interpreter

```
>>> type('hello world')
<class 'str'>
>>> type("hello world")
<class 'str'>
>>> type(3.14)
<class 'float'>
```



- Jeder Wert gehört zu (genau) einem **Typ**
- Mithilfe der Funktion `type` kann man den Typ eines Wertes bzw. des Literals erfahren:

Werte und  
Typen

Variablen

Ausdrücke

## Python-Interpreter

```
>>> type('hello world')
<class 'str'>
>>> type("hello world")
<class 'str'>
>>> type(3.14)
<class 'float'>
>>>
```

- Jeder Wert gehört zu (genau) einem **Typ**
- Mithilfe der Funktion `type` kann man den Typ eines Wertes bzw. des Literals erfahren:

Werte und  
Typen

Variablen

Ausdrücke

## Python-Interpreter

```
>>> type('hello world')
<class 'str'>
>>> type("hello world")
<class 'str'>
>>> type(3.14)
<class 'float'>
>>> type(3)
```

- Jeder Wert gehört zu (genau) einem **Typ**
- Mithilfe der Funktion `type` kann man den Typ eines Wertes bzw. des Literals erfahren:

Werte und  
Typen

Variablen

Ausdrücke

## Python-Interpreter

```
>>> type('hello world')
<class 'str'>
>>> type("hello world")
<class 'str'>
>>> type(3.14)
<class 'float'>
>>> type(3)
<class 'int'>
```



# Variablen



- Man kann einem Wert einen Namen (**Variablennamen**) geben. Dazu werden der Name auf der linken und das entsprechende Literal auf der rechten Seite eines Gleichheitszeichens geschrieben. Eine solche Operation wird **Zuweisung** genannt:

Werte und  
Typen

Variablen

Ausdrücke



- Man kann einem Wert einen Namen (**Variablennamen**) geben. Dazu werden der Name auf der linken und das entsprechende Literal auf der rechten Seite eines Gleichheitszeichens geschrieben. Eine solche Operation wird **Zuweisung** genannt:

Werte und  
Typen

Variablen

Ausdrücke

## Python-Interpreter

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





- Man kann einem Wert einen Namen (**Variablennamen**) geben. Dazu werden der Name auf der linken und das entsprechende Literal auf der rechten Seite eines Gleichheitszeichens geschrieben. Eine solche Operation wird **Zuweisung** genannt:

Werte und  
Typen

Variablen

Ausdrücke

## Python-Interpreter

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



- Man kann einem Wert einen Namen (**Variablennamen**) geben. Dazu werden der Name auf der linken und das entsprechende Literal auf der rechten Seite eines Gleichheitszeichens geschrieben. Eine solche Operation wird **Zuweisung** genannt:

Werte und  
Typen

Variablen

Ausdrücke

## Python-Interpreter

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



- Man kann einem Wert einen Namen (**Variablennamen**) geben. Dazu werden der Name auf der linken und das entsprechende Literal auf der rechten Seite eines Gleichheitszeichens geschrieben. Eine solche Operation wird **Zuweisung** genannt:

Werte und  
Typen

Variablen

Ausdrücke

## Python-Interpreter

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

- Man sagt: Der *Wert* 111 wird der *Variablen* spam zugewiesen.

- Man kann einem Wert einen Namen (**Variablennamen**) geben. Dazu werden der Name auf der linken und das entsprechende Literal auf der rechten Seite eines Gleichheitszeichens geschrieben. Eine solche Operation wird **Zuweisung** genannt:

Werte und  
Typen

Variablen

Ausdrücke

## Python-Interpreter

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

- Man sagt: Der *Wert* 111 wird der *Variablen* `spam` zugewiesen.
- In Python stellt man sich besser eher vor, dass der Wert ein Namensschild erhält (ein Wert kann auch mehr als ein Namensschild erhalten).



- Im Gegensatz zur mathematischen Notation kann sich der Wert einer Variablen durch Neuzuweisung ändern (Namensschild umhängen)

Werte und  
Typen

Variablen

Ausdrücke

## Python-Interpreter

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



- Im Gegensatz zur mathematischen Notation kann sich der Wert einer Variablen durch Neuzuweisung ändern (Namensschild umhängen)

Werte und Typen

Variablen

Ausdrücke

## Python-Interpreter

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



- Im Gegensatz zur mathematischen Notation kann sich der Wert einer Variablen durch Neuzuweisung ändern (Namensschild umhängen)

Werte und Typen

Variablen

Ausdrücke

## Python-Interpreter

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

- Im Gegensatz zur mathematischen Notation kann sich der Wert einer Variablen durch Neuzuweisung ändern (Namensschild umhängen)

## Python-Interpreter

```
>>> spam = 111
>>> spam
111
>>> spam = 112
>>>
```



- Im Gegensatz zur mathematischen Notation kann sich der Wert einer Variablen durch Neuzuweisung ändern (Namensschild umhängen)

Werte und  
Typen

Variablen

Ausdrücke

## Python-Interpreter

```
>>> spam = 111
>>> spam
111
>>> spam = 112
>>> spam
```



- Im Gegensatz zur mathematischen Notation kann sich der Wert einer Variablen durch Neuzuweisung ändern (Namensschild umhängen)

Werte und  
Typen

Variablen

Ausdrücke

## Python-Interpreter

```
>>> spam = 111
>>> spam
111
>>> spam = 112
>>> spam
112
```



- Der Typ einer Variablen ist immer der Typ des Wertes den die Variable benennt:

## Python-Interpreter

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

Werte und  
Typen

Variablen

Ausdrücke

- Der Typ einer Variablen ist immer der Typ des Wertes den die Variable benennt:

## Python-Interpreter

```
>>> spam = 'egg'  
>>> type(spam)
```

Werte und  
Typen

Variablen

Ausdrücke



- Der Typ einer Variablen ist immer der Typ des Wertes den die Variable benennt:

## Python-Interpreter

```
>>> spam = 'egg'  
>>> type(spam)  
<class 'str'>  
>>>
```

Werte und  
Typen

Variablen

Ausdrücke



- Der Typ einer Variablen ist immer der Typ des Wertes den die Variable benennt:

## Python-Interpreter

```
>>> spam = 'egg'  
>>> type(spam)  
<class 'str'>  
>>> spam = 42  
>>>
```

Werte und  
Typen

Variablen

Ausdrücke



- Der Typ einer Variablen ist immer der Typ des Wertes den die Variable benennt:

## Python-Interpreter

```
>>> spam = 'egg'  
>>> type(spam)  
<class 'str'>  
>>> spam = 42  
>>> type(spam)
```

Werte und  
Typen

Variablen

Ausdrücke

- Der Typ einer Variablen ist immer der Typ des Wertes den die Variable benennt:

## Python-Interpreter

```
>>> spam = 'egg'  
>>> type(spam)  
<class 'str'>  
>>> spam = 42  
>>> type(spam)  
<class 'int'>
```

- Das heißt, dass im Gegensatz zu anderen Programmiersprachen die Variablen **dynamisch typisiert** sind.

Werte und  
Typen

Variablen

Ausdrücke



- Der **Zustand** eines Berechnungsprozess kann vollständig durch die **Wertebelegung der Variablen** und den aktuellen Ausführungspunkt beschrieben werden. Die Wertebelegung kann durch ein **Zustandsdiagramm** visualisiert werden.

Werte und  
Typen

Variablen

Ausdrücke

## Python-Interpreter

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

- Das Zustandsdiagramm nach der Ausführung:

Global frame	
spam	123
egg	"spam"

# Erlaubte Variablenamen



In Variablenamen erlaubt sind Groß- und Kleinbuchstaben einschließlich Umlauten und Unterstriche sowie Ziffern. Das erste Zeichen darf keine Ziffer sein.

Werte und  
Typen

Variablen

Ausdrücke

## Python-Interpreter

```
>>> Heißwasser = 1  
>>>
```

# Erlaubte Variablennamen



In Variablennamen erlaubt sind Groß- und Kleinbuchstaben einschließlich Umlauten und Unterstriche sowie Ziffern. Das erste Zeichen darf keine Ziffer sein.

Werte und  
Typen

Variablen

Ausdrücke

## Python-Interpreter

```
>>> Heißwasser = 1
>>> Kaltes Wasser = 2
```

# Erlaubte Variablenamen



In Variablenamen erlaubt sind Groß- und Kleinbuchstaben einschließlich Umlauten und Unterstriche sowie Ziffern. Das erste Zeichen darf keine Ziffer sein.

Werte und  
Typen

Variablen

Ausdrücke

## Python-Interpreter

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

# Erlaubte Variablennamen



In Variablennamen erlaubt sind Groß- und Kleinbuchstaben einschließlich Umlauten und Unterstriche sowie Ziffern. Das erste Zeichen darf keine Ziffer sein.

Werte und  
Typen

Variablen

Ausdrücke

## Python-Interpreter

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

In Variablennamen erlaubt sind Groß- und Kleinbuchstaben einschließlich Umlauten und Unterstriche sowie Ziffern. Das erste Zeichen darf keine Ziffer sein.

Werte und  
Typen

Variablen

Ausdrücke

## Python-Interpreter

```
>>> Heißwasser = 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
```



## Python-Interpreter

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

Werte und  
Typen

Variablen

Ausdrücke

## Python-Interpreter

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

Werte und  
Typen

Variablen

Ausdrücke

**Schlüsselwörter** dürfen nicht als Variablennamen 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	





- Variablen werden ins Leben gerufen, sobald ihnen erstmals ein Wert zugewiesen wird. Sie sind nicht verwendbar, bevor ihnen ein Wert zugewiesen wurde:

## Python-Interpreter

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

Werte und  
Typen

Variablen

Ausdrücke

- Variablen werden ins Leben gerufen, sobald ihnen erstmals ein Wert zugewiesen wird. Sie sind nicht verwendbar, bevor ihnen ein Wert zugewiesen wurde:

## Python-Interpreter

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

Werte und  
Typen

Variablen

Ausdrücke

- Variablen werden ins Leben gerufen, sobald ihnen erstmals ein Wert zugewiesen wird. Sie sind nicht verwendbar, bevor ihnen ein Wert zugewiesen wurde:

## Python-Interpreter

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

Werte und  
Typen

Variablen

Ausdrücke

- Variablen werden ins Leben gerufen, sobald ihnen erstmals ein Wert zugewiesen wird. Sie sind nicht verwendbar, bevor ihnen ein Wert zugewiesen wurde:

## Python-Interpreter

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

Werte und  
Typen

Variablen

Ausdrücke

- Variablen werden ins Leben gerufen, sobald ihnen erstmals ein Wert zugewiesen wird. Sie sind nicht verwendbar, bevor ihnen ein Wert zugewiesen wurde:

Werte und  
Typen

Variablen

Ausdrücke

## Python-Interpreter

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

- Variablen werden ins Leben gerufen, sobald ihnen erstmals ein Wert zugewiesen wird. Sie sind nicht verwendbar, bevor ihnen ein Wert zugewiesen wurde:

## Python-Interpreter

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

Werte und  
Typen

Variablen

Ausdrücke

- Variablen werden ins Leben gerufen, sobald ihnen erstmals ein Wert zugewiesen wird. Sie sind nicht verwendbar, bevor ihnen ein Wert zugewiesen wurde:

Werte und  
Typen

Variablen

Ausdrücke

## 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
```



# Ausdrücke





Wir hatten bereits **Operatoren** auf Zahlen kennen gelernt: +, −, \*, ...

**Ausdrücke** werden aus Operatoren, Literalen und Variablen zusammen gesetzt und haben einen Wert, der sich bei arithmetischen Ausdrücken nach den üblichen **Präzedenzregeln** ergibt, d.h.

- immer die Klammerung zuerst beachtend,

Werte und  
Typen

Variablen

Ausdrücke



Wir hatten bereits **Operatoren** auf Zahlen kennen gelernt: +, −, \*, ...

**Ausdrücke** werden aus Operatoren, Literalen und Variablen zusammen gesetzt und haben einen Wert, der sich bei arithmetischen Ausdrücken nach den üblichen

**Präzedenzregeln** ergibt, d.h.

- immer die Klammerung zuerst beachtend,
- dann die Exponentiation auswertend,

Werte und  
Typen

Variablen

Ausdrücke



Wir hatten bereits **Operatoren** auf Zahlen kennen gelernt: +, −, \*, ...

**Ausdrücke** werden aus Operatoren, Literalen und Variablen zusammen gesetzt und haben einen Wert, der sich bei arithmetischen Ausdrücken nach den üblichen

**Präzedenzregeln** ergibt, d.h.

- immer die Klammerung zuerst beachtend,
- dann die Exponentiation auswertend,
- danach Multiplikation und Division,

Werte und  
Typen

Variablen

Ausdrücke



Wir hatten bereits **Operatoren** auf Zahlen kennen gelernt: +, −, \*, ...

**Ausdrücke** werden aus Operatoren, Literalen und Variablen zusammen gesetzt und haben einen Wert, der sich bei arithmetischen Ausdrücken nach den üblichen

**Präzedenzregeln** ergibt, d.h.

- immer die Klammerung zuerst beachtend,
- dann die Exponentiation auswertend,
- danach Multiplikation und Division,
- dann Addition und Subtraktion,

Werte und  
Typen

Variablen

Ausdrücke



Wir hatten bereits **Operatoren** auf Zahlen kennen gelernt: +, −, \*, ...

**Ausdrücke** werden aus Operatoren, Literalen und Variablen zusammen gesetzt und haben einen Wert, der sich bei arithmetischen Ausdrücken nach den üblichen

**Präzedenzregeln** ergibt, d.h.

- immer die Klammerung zuerst beachtend,
- dann die Exponentiation auswertend,
- danach Multiplikation und Division,
- dann Addition und Subtraktion,
- bei gleicher Präzedenz wird von von links nach rechts ausgewertet, außer bei der Exponentiation

Werte und  
Typen

Variablen

Ausdrücke



Werte und  
Typen

Variablen

Ausdrücke

## Python-Interpreter

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



Werte und  
Typen

Variablen

Ausdrücke

## Python-Interpreter

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



## Python-Interpreter

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





## Python-Interpreter

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



## Python-Interpreter

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



## Python-Interpreter

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



## Python-Interpreter

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



## Python-Interpreter

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



## Python-Interpreter

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



- Auf Strings gibt es nur den Operator '+' (**Konkatenation**)

## Python-Interpreter

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

Werte und  
Typen

Variablen

Ausdrücke

- Auf Strings gibt es nur den Operator '+' (**Konkatenation**)

## Python-Interpreter

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

- Außerdem kann man Strings mit ganzen Zahlen multiplizieren:

## Python-Interpreter

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

Werte und  
Typen

Variablen

Ausdrücke



- Auf Strings gibt es nur den Operator '+' (**Konkatenation**)

## Python-Interpreter

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

- Außerdem kann man Strings mit ganzen Zahlen multiplizieren:

## Python-Interpreter

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

Werte und  
Typen

Variablen

Ausdrücke

- Auf Strings gibt es nur den Operator '+' (**Konkatenation**)

## Python-Interpreter

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

- Außerdem kann man Strings mit ganzen Zahlen multiplizieren:

## Python-Interpreter

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

Werte und  
Typen

Variablen

Ausdrücke

- Auf Strings gibt es nur den Operator '+' (**Konkatenation**)

## Python-Interpreter

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

- Außerdem kann man Strings mit ganzen Zahlen multiplizieren:

## Python-Interpreter

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

Werte und  
Typen

Variablen

Ausdrücke

- Auf Strings gibt es nur den Operator '+' (**Konkatenation**)

## Python-Interpreter

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

- Außerdem kann man Strings mit ganzen Zahlen multiplizieren:

## Python-Interpreter

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

Werte und  
Typen

Variablen

Ausdrücke

- Auf Strings gibt es nur den Operator '+' (**Konkatenation**)

## Python-Interpreter

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

- Außerdem kann man Strings mit ganzen Zahlen multiplizieren:

## Python-Interpreter

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

Werte und  
Typen

Variablen

Ausdrücke



- Statt Literalen kann man (natürlich) auch Ausdrücke in Zuweisungen verwenden:

## Python-Interpreter

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

Werte und  
Typen

Variablen

Ausdrücke



- Statt Literalen kann man (natürlich) auch Ausdrücke in Zuweisungen verwenden:

## Python-Interpreter

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

Werte und  
Typen

Variablen

Ausdrücke



- Statt Literalen kann man (natürlich) auch Ausdrücke in Zuweisungen verwenden:

## Python-Interpreter

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

Werte und  
Typen

Variablen

Ausdrücke





- Statt Literalen kann man (natürlich) auch Ausdrücke in Zuweisungen verwenden:

## Python-Interpreter

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

Werte und  
Typen

Variablen

Ausdrücke



- Statt Literalen kann man (natürlich) auch Ausdrücke in Zuweisungen verwenden:

## 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
>>>
```

Werte und  
Typen

Variablen

Ausdrücke

- Statt Literalen kann man (natürlich) auch Ausdrücke in Zuweisungen verwenden:

## 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
>>>
```

Werte und  
Typen

Variablen

Ausdrücke



- Statt Literalen kann man (natürlich) auch Ausdrücke in Zuweisungen verwenden:

## 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
```

Werte und  
Typen

Variablen

Ausdrücke

- Statt Literalen kann man (natürlich) auch Ausdrücke in Zuweisungen verwenden:

## 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
```

Werte und  
Typen

Variablen

Ausdrücke



- Oft möchte man den Wert einer Variablen ändern, d.h. um einen Betrag erhöhen, multiplizieren, usw.
- $X = X + Y$ ,  $X = X * Y$ , usw.
- Dafür gibt es die **erweiterten Zuweisungen**
- $X += Y$  entspricht  $X = X + Y$
- $X *= Y$  entspricht  $X = X * Y$
- $X /= Y$  entspricht  $X = X / Y$
- Auch für:  $\&$ ,  $-$ ,  $|$ ,  $\wedge$ ,  $>>$ ,  $\%$ ,  $<<$ ,  $**$ ,  $//$



- **Werte** und ihre Darstellung als **Literale** gehören zu den Basiskomponenten von Programmiersprachen
- Werte haben alle einen bestimmten **Typ**
- Werten kann durch eine **Zuweisung** ein Name gegeben werden.
- Dieser Name wird als **Variable** bezeichnet.
- Der Wert einer Variablen kann sich ändern.
- Ausdrücke werden aus Operatoren, Literalen und Variablen gebildet.
- Sie haben einen Wert!
- Bei einer Zuweisung wird immer erst die rechte Seite ausgewertet, dann wird der Wert zugewiesen!

Werte und  
Typen

Variablen

Ausdrücke