Python: Wahrheitswerte und Vergleiche Lösung Theorie

Wahrheitswerte

Python kennt zwei Wahrheitswerte: `True` und `False`.

Beachte die Gross- und Kleinschreibung.

Vergleichsoperatoren

Vergleichsoperatoren ergeben jeweils einen Wahrheitswert (True oder False).

Operator	Ausdruck	Wert
`==`	`x == y`	`True`, wenn $x = y$ und `False` sonst
`!=`	`x != y`	`True`, wenn $x \neq y$ und `False` sonst
`<`	`x < y`	`True`, wenn $x < y$ und `False` sonst
`<=`	`x <= y`	`True`, wenn $x \leq y$ und `False` sonst
`>`	`x > y`	`True`, wenn $x > y$ und `False` sonst
`>=`	`x >= y`	`True`, wenn $x \geq y$ und `False` sonst

Warum verwendet man in Python für den Test auf Gleichheit nicht `x = y`?

Weil `x = y` bedeutet, dass der Wert von `y` der Variablen `x` zugewiesen wird.

Beispiel 1

(a) `5 < 4` `False`

(b) `7 >= 7` `True`

(c) `4 != 4` `False`

(d) `1+8 < 5+6` `True`

Das logische NEIN (Python: `not`)

Merke: `not A` ist genau dann wahr, wenn der Operand `A` falsch ist.

A	not A
True	False
False	True

Das logische UND (Python: `and`)

Merke: `A and B` ist genau dann wahr, wenn beide Operanden wahr sind.

A	B	A and B
True	True	True
True	False	False
False	True	False
False	False	False

Das logische ODER (Python: `or`)

Merke: `A or B` ist genau dann falsch, wenn beide Operanden falsch sind.

A	B	A or B
True	True	True
True	False	True
False	True	True
False	False	False

Hierarchie der Python-Operatoren

Priorität	**Operatoren**	**Bedeutung**
1	`()`	Klammern
2	`**`	Potenzieren
3	`-`	unäres Minus (Vorzeichenwechsel)
4	`*`, `/`, `//`, `%`	Multiplikation und Division
5	`+`, `-`	Addition und Subtraktion
6	`==`, `!=`, `<`, `<=`, `>`, `>=`	Vergleichsoperatoren
7	`not`	Logische Negation
8	`and`	Logisches UND
9	`or`	Logisches ODER

Operationen gleicher Priorität werden grundsätzlich von links nach rechts ausgewertet, mit Ausnahme der Potenzierung (`**`), die von rechts nach links bindet. Klammern sind eigentlich keine Operatoren; Klammerausdrücke werden von innen nach aussen ausgewertet. Die Tabelle ist vereinfacht; einige Operatoren fehlen.

Beispiel 2

(a) `5 < 3 or 7 > 4` True

(b) `not False or False` True

(c) `not False and False` False

(d) `(5 == 7) or (5 != 5)` False

(e) `True and False and True and True and True` False

Beim letzten Beispiel kann man sich die *Kurzschlusseigenschaft* der logischen Operatoren zunutze machen. Sobald der Wert des Ausdrucks feststeht, wird er nicht mehr weiter ausgewertet.