Der Datentyp `bool` in Python

Einführung

In Python ist `bool` ein Datentyp, der einen von zwei Werten annehmen kann: `True` oder `False`. Er wird oft in Bedingungen und Kontrollstrukturen verwendet, um den Fluss eines Programms zu steuern.

Grundlegende Verwendung

```
ist_wahr = True
ist_falsch = False
```

Boolesche Operationen

Python unterstützt die grundlegenden logischen Operationen: `and`, `or`, und `not`.

Beispiel

```
a = True
b = False

print(a and b)  # False
print(a or b)  # True
print(not a)  # False
```

Verwendung in Bedingungen

`bool` wird häufig in `if`-Anweisungen und anderen Kontrollstrukturen verwendet.

Beispiel

```
if ist_wahr:
    print("Das ist wahr!")
else:
    print("Das ist falsch!")
```

Konvertierung zu `bool`

Python ermöglicht die Konvertierung anderer Datentypen zu `bool`. Generell gilt: Leere Sequenzen und der Wert `0` werden als `False` betrachtet, alles andere als `True`.

Beispiel

```
print(bool(0))  # False
print(bool(""))  # False
print(bool("Hallo"))  # True
print(bool(123))  # True
```

Boolesche Werte in logischen Ausdrücken

Logische Ausdrücke und Bedingungen evaluieren zu `True` oder `False`, was für die Steuerung des Programmflusses genutzt wird.

Beispiel

```
zahl = 5
print(zahl > 0)  # True
print(zahl == 0)  # False
```

Arithmetische Vergleichsoperatoren und Boolesche Werte in Python

Einführung

In Python werden arithmetische Vergleichsoperatoren verwendet, um zwei Werte zu vergleichen. Das Ergebnis eines solchen Vergleichs ist ein Boolescher Wert: `True` oder `False`. Diese Operatoren sind grundlegend für die Erstellung von Bedingungen und die Steuerung des Programmflusses.

Arithmetische Vergleichsoperatoren

Operator	Beschreibung
`==`	Gleich
`!=`	Ungleich
`>`	Größer als
`<`	Kleiner als
`>=`	Größer als oder gleich
`<=`	Kleiner als oder gleich

Beispiele

Gleichheit und Ungleichheit

```
print(5 == 5)  # True
print(5 != 5)  # False
```

Größer als und Kleiner als

```
print(10 > 5)  # True
print(2 < 3)   # True
```

Größer als oder gleich und Kleiner als oder gleich

```
print(5 >= 5)  # True
print(4 <= 3)  # False
```

Verwendung in Bedingungen

Vergleichsoperatoren werden oft in `if`-Anweisungen verwendet, um zu entscheiden, welcher Codeblock ausgeführt werden soll.

Beispiel

```
alter = 18
if alter >= 18:
    print("Du bist volljährig.")
else:
    print("Du bist minderjährig.")
```

Boolesche Logik in Vergleichen

Boolesche Operatoren (`and`, `or`, `not`) können mit Vergleichsoperatoren kombiniert werden, um komplexe Bedingungen zu erstellen.

Beispiel

```
alter = 25
name = "Anna"
if alter > 18 and name == "Anna":
    print("Anna ist älter als 18.")
```