🧭 Python Operatoren – Der Ultimative Guide

Python hat viele Arten von Operatoren, um Werte zu vergleichen, zu kombinieren oder zu verändern.
Hier lernst du die wichtigsten Gruppen:

🔢 1. Arithmetische Operatoren

Diese Operatoren werden für mathematische Berechnungen verwendet.

Operator	Bedeutung	Beispiel	Ergebnis
`+`	Addition	`5 + 2`	`7`
`-`	Subtraktion	`5 - 2`	`3`
`*`	Multiplikation	`5 * 2`	`10`
`/`	Division (mit Komma)	`5 / 2`	`2.5`
`//`	Ganzzahldivision	`5 // 2`	`2`
`%`	Modulo (Rest)	`5 % 2`	`1`
`**`	Potenzierung	`2 ** 3`	`8`

⚖️ 2. Vergleichsoperatoren (Relationale Operatoren)

Vergleichen zwei Werte und liefern `True` oder `False`.

Operator	Bedeutung	Beispiel	Ergebnis
`==`	Gleich	`5 == 5`	`True`
`!=`	Ungleich	`5 != 3`	`True`
`>`	Größer als	`5 > 2`	`True`
`<`	Kleiner als	`3 < 5`	`True`
`>=`	Größer oder gleich	`5 >= 5`	`True`
`<=`	Kleiner oder gleich	`4 <= 5`	`True`

🔮 3. Logische (Boolean) Operatoren

Diese kombinieren Wahrheitswerte (`True` / `False`).

Operator	Bedeutung	Beispiel	Ergebnis
`and`	Beide müssen wahr sein	`(5 > 2) and (3 < 4)`	`True`
`or`	Einer muss wahr sein	`(5 < 2) or (3 < 4)`	`True`
`not`	Negiert den Wert	`not (5 > 2)`	`False`

🧠 **Tipp:**
Python wertet logische Ausdrücke von links nach rechts aus – aber `not` hat die höchste Priorität, dann `and`, dann `or`.

⚙️ 4. Bitweise Operatoren

Diese arbeiten auf **Binär-Ebene** (Bit für Bit).

Operator	Bedeutung	Beispiel (a=5, b=3)	Erklärung
`&`	UND	`5 & 3 → 1`	`0101 & 0011 = 0001`
`` ` ``	`` ` ``	ODER	`` `5 ``
`^`	XOR (entweder-oder)	`5 ^ 3 → 6`	`0101 ^ 0011 = 0110`
`~`	NOT (invertiert alle Bits)	`~5 → -6`	Bitweise Negation
`<<`	Linksverschiebung	`5 << 1 → 10`	`0101 → 1010`
`>>`	Rechtsverschiebung	`5 >> 1 → 2`	`0101 → 0010`

🧠 **Tipp:** Bitweise Operatoren werden oft in System- und Hardware-Programmierung verwendet.

🧩 5. Zuweisungsoperatoren

Diese kombinieren Operation und Zuweisung.

Operator	Bedeutung	Beispiel	Ergebnis
`=`	Zuweisung	`x = 5`	`x` ist `5`
`+=`	Addieren & Zuweisen	`x += 3`	`x = x + 3`
`-=`	Subtrahieren & Zuweisen	`x -= 2`	`x = x - 2`
`*=`	Multiplizieren & Zuweisen	`x *= 2`	`x = x * 2`
`/=`	Dividieren & Zuweisen	`x /= 2`	`x = x / 2`
`//=`	Ganzzahldivision	`x //= 2`	`x = x // 2`
`%=`	Modulo	`x %= 2`	`x = x % 2`
`**=`	Potenzieren	`x **= 2`	`x = x ** 2`
`&=`, `|=`, `^=`, `<<=`, `>>=`	Bitweise Kombination	–	

📦 6. Mitgliedschaftsoperatoren

Prüfen, ob ein Element in einer Sequenz vorkommt.

Operator	Bedeutung	Beispiel	Ergebnis
`in`	Element vorhanden	`'a' in 'abc'`	`True`
`not in`	Element nicht vorhanden	`'x' not in 'abc'`	`True`

Beispiel:

```python
fruits = ["apple", "banana", "cherry"]
print("apple" in fruits)      # True
print("orange" not in fruits) # True
```

🧱 7. Identitätsoperatoren

Prüfen, ob zwei Variablen auf dasselbe Objekt im Speicher zeigen.

Operator	Bedeutung	Beispiel	Ergebnis
`is`	gleiche Identität	`a is b`	True, wenn beide dieselbe Referenz haben
`is not`	verschiedene Identität	`a is not b`	True, wenn unterschiedlich

Beispiel:

```python
a = [1, 2]
b = a
c = [1, 2]
print(a is b)   # True  (gleiche Referenz)
print(a is c)   # False (gleicher Inhalt, aber anderes Objekt)
```

💡 8. Operator-Priorität (Reihenfolge der Auswertung)

Von **höchster** zu **niedrigster** Priorität:

Priorität	Operator
1	`()` (Klammern)
2	`**`
3	`+x`, `-x`, `~x`
4	`*`, `/`, `//`, `%`
5	`+`, `-`
6	`<<`, `>>`
7	`&`
8	`^`
9	`|`
10	Vergleichsoperatoren (`<`, `<=`, `>`, `>=`, `==`, `!=`)
11	`not`
12	`and`
13	`or`

Antwort auf YouTube

🧠 Beispiel – Alles zusammen

```python
x = 5
y = 10

if (x < y and y % 2 == 0) or not (x == 5):
    print("Bedingung erfüllt!")
else:
    print("Nicht erfüllt.")
```

Erklärung:

- `(x < y and y % 2 == 0)` → True
- `not (x == 5)` → False
- `True or False` → True ✅
  ➡️ Ausgabe: `"Bedingung erfüllt!"`

✅ Zusammenfassung

Kategorie	Beispiele
Arithmetisch	`+`, `-`, `*`, `/`, `//`, `%`, `**`
Vergleich	`==`, `!=`, `<`, `>`, `<=`, `>=`
Logisch	`and`, `or`, `not`
Bitweise	`&`, `
Zuweisung	`=`, `+=`, `-=`, …
Mitgliedschaft	`in`, `not in`
Identität	`is`, `is not`
