



EINFÜHRUNG IN PROGRAMMIERUNG UND DATENBANKEN

joern ploennigs

grundlagen

Motivation

Computer und
Architekturen

Programmierung
und Datentypen

Verzweigungen und
Schleifen

modellierung

Fehler und
Debugging

Objektorientierung u.
Softwareentwurf

Funktionen und
Rekursion

OPERATOREN



DALL-E 2: A sign for in and out

ANWEISUNGEN UND AUSDRÜCKE

- Eine **Anweisung (engl. statement)** stellt eine in der Syntax einer Programmiersprache formulierte *einzelne Vorschrift* dar, die im Programm nacheinander auszuführen ist
- Eine besondere Form von Anweisungen sind **Ausdrücke (engl. expression)** welche immer einen Wert zurückgeben und somit einen Datentypen haben
- Bei einer Variablenzuweisung stehen die Ausdrücke rechts von dem Zuweisungsoperator

Variablenname = Ausdruck

- Beispielausdrücke:
 - Konstante Werte wie z. B. 5 oder "ABC"
 - Der Wert einer Variablen, z. B. x
 - Das Ergebnis von Operationen (Dazu im folgenden mehr)
- Ausdrücke können über Operatoren kombiniert werden, was wieder einen Ausdruck erzeugt

ARITHMETISCHE OPERATOREN AUS DER MATHEMATIK!

- Bilden einen oder zwei Werte (Operanden) auf einen neuen Wert ab
- Arithmetische Operatoren auf zwei Werten: + - * / (Grundrechenarten)
- Operator auf einem Wert: - (Negativ)
- In der Programmierung: Nimmt ein bis zwei Ausdrücke und ist selbst ein Ausdruck
- Können verkettet werden – Berechnungsabfolge ist wie in der Mathematik

$$(6 + a) * (-b)$$

ZUWEISUNGSOOPERATOREN

- Kombinieren von arithmetischen Operatoren mit einer Zuweisung (=)
- Führt einen Operator auf den Variablenwert und den Rückgabewert des Ausdrucks aus und weist den entstehenden Wert der Variable zu
- Beispiel:

`a = 5` # a hat den Wert 5

`a += 1` # a hat nun den Wert 6

- In der Praxis meistens als Abkürzung für simple Berechnungen relevant.

VERGLEICHSOPERATOREN - GLEICH IST NICHT =

- = ist in den meisten Programmiersprachen das Symbol für das Belegen von Variablen
- Für Aussagen über die Gleichheit von Werten nutzen wir daher folgende Operatoren.

==	ist-gleich
!=	ungleich
<	kleiner
<=	kleiner-gleich
>	größer
>=	größer-gleich

LOGISCHE UND BITWEISE OPERATOREN

- In Python gibt es nicht die üblichen logischen Operatoren & (AND), | (OR), ~ (NOT)
- Stattdessen werden logische Operatoren ausgeschrieben **and**, **or**, **not**
- Angewendet auf boolesche Werte (**bool**) mit den Werten **True/False**

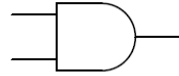
True and True

False or not False

Hörsaalfrage

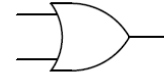
WELCHE ERGEBNISSE GIBT ES FÜR
WELCHE
LOGISCHE OPERATION?

(ENTSPRICHT DER
AUSSAGENLOGIK)



AND

A	B	Ergebnis
Falsch (0)	Falsch (0)	Falsch (0)
Falsch (0)	Wahr (1)	Falsch (0)
Wahr (1)	Falsch (0)	Falsch (0)
Wahr (1)	Wahr (1)	Wahr (1)



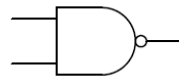
OR

A	B	Ergebnis
Falsch (0)	Falsch (0)	Falsch (0)
Falsch (0)	Wahr (1)	Wahr (1)
Wahr (1)	Falsch (0)	Wahr (1)
Wahr (1)	Wahr (1)	Wahr (1)



XOR

A	B	Ergebnis
Falsch (0)	Falsch (0)	Falsch (0)
Falsch (0)	Wahr (1)	Wahr (1)
Wahr (1)	Falsch (0)	Wahr (1)
Wahr (1)	Wahr (1)	Falsch (0)



NAND

A	B	Ergebnis
Falsch (0)	Falsch (0)	Wahr (1)
Falsch (0)	Wahr (1)	Wahr (1)
Wahr (1)	Falsch (0)	Wahr (1)
Wahr (1)	Wahr (1)	Falsch (0)



NOR

A	B	Ergebnis
Falsch (0)	Falsch (0)	Wahr (1)
Falsch (0)	Wahr (1)	Falsch (0)
Wahr (1)	Falsch (0)	Falsch (0)
Wahr (1)	Wahr (1)	Falsch (0)

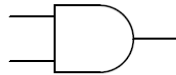


XNOR

A	B	Ergebnis
Falsch (0)	Falsch (0)	Wahr (1)
Falsch (0)	Wahr (1)	Falsch (0)
Wahr (1)	Falsch (0)	Falsch (0)
Wahr (1)	Wahr (1)	Wahr (1)

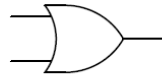
N steht für `Negation`

Eine CPU kann nur einfache logische Operationen ausführen wie NOT, AND, OR, XOR und Negationen. Aus diesen Grundoperationen lassen sich komplexere Operationen zusammen



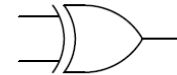
AND

A	B	Ergebnis
Falsch (0)	Falsch (0)	Falsch (0)
Falsch (0)	Wahr (1)	Falsch (0)
Wahr (1)	Falsch (0)	Falsch (0)
Wahr (1)	Wahr (1)	Wahr (1)



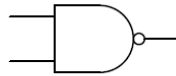
OR

A	B	Ergebnis
Falsch (0)	Falsch (0)	Falsch (0)
Falsch (0)	Wahr (1)	Wahr (1)
Wahr (1)	Falsch (0)	Wahr (1)
Wahr (1)	Wahr (1)	Wahr (1)



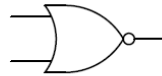
XOR

A	B	Ergebnis
Falsch (0)	Falsch (0)	Falsch (0)
Falsch (0)	Wahr (1)	Wahr (1)
Wahr (1)	Falsch (0)	Wahr (1)
Wahr (1)	Wahr (1)	Falsch (0)



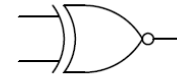
NAND

A	B	Ergebnis
Falsch (0)	Falsch (0)	Wahr (1)
Falsch (0)	Wahr (1)	Wahr (1)
Wahr (1)	Falsch (0)	Wahr (1)
Wahr (1)	Wahr (1)	Falsch (0)



NOR

A	B	Ergebnis
Falsch (0)	Falsch (0)	Wahr (1)
Falsch (0)	Wahr (1)	Falsch (0)
Wahr (1)	Falsch (0)	Falsch (0)
Wahr (1)	Wahr (1)	Falsch (0)



XNOR

A	B	Ergebnis
Falsch (0)	Falsch (0)	Wahr (1)
Falsch (0)	Wahr (1)	Falsch (0)
Wahr (1)	Falsch (0)	Falsch (0)
Wahr (1)	Wahr (1)	Wahr (1)

N steht für 'Negation'

LOGISCHE UND BITWEISE OPERATOREN

- In Python gibt es nicht die üblichen logischen Operatoren & (AND), | (OR), ~ (NOT)
- Stattdessen werden logische Operatoren ausgeschrieben **and**, **or**, **not**
- Angewendet auf boolesche Werte (**bool**) mit den Werten **True/False**

True and True

False or not False

- Bitweise Operatoren nutzen die üblichen Operatoren werden aber nur auf Bit-Werte
- Zusätzlich: (mit $a = 0011\ 1100$ als Beispielwert)

$\sim a$ = **1100 0011** # Komplement

$a \ll n$ = **1111 0000** # Linksverschiebung um n , Multiplikation mit 2^n

$a \gg n$ = **0000 1111** # Rechtsverschiebung um n , Division durch 2^n

OPERATOREN - VERBINDUNG ZUR HARDWARE

- Beachte die Ähnlichkeit zur Hardware-Funktion des ALU!
 - Aus 1 bis 2 Operanden wird mittels einer festgelegten Operation ein Resultat.
- Der ALU berechnet also Ergebnisse von Operatoren.
- Überspitzt formuliert: Alle Berechnungen, die jemals auf einem Computer ausgeführt worden sind, waren nichts als Abfolgen von Speicherbelegungen und den ebend vorgestellten Operatoren.

Hörsaalfrage

FRAGEN?



DALL·E 2: A psychedelic DJ with a question mark for a head