

Informatik I: Einführung in die Programmierung

3. Werte, Typen, Variablen und Ausdrücke

Albert-Ludwigs-Universität Freiburg



Prof. Dr. Peter Thiemann

24. Oktober 2023

1 Exkursion: Datenrepräsentation



Exkursion:
Datenrepräsentation

Werte und
Typen

Variable

Ausdrücke

- Der Computer repräsentiert Daten als Folgen von **Bits**.
- Ein Bit (*binary digit*) ist die kleinste Informationseinheit.
- Zwei mögliche Werte: 0 oder 1
- Technische Realisierung: Schalter ein / Schalter aus bzw. geladen / entladen.
- Logische Interpretation: **Wahrheitswerte** 0 = falsch, 1 = wahr

Exkursion:
Datenrepräsentation

Werte und
Typen

Variable

Ausdrücke

- **Logisches Und:** $b_1 \wedge b_2$

Ergebnis ist 1, falls $b_1 = 1$ **und** $b_2 = 1$, sonst 0.

$$1 \wedge 1 = 1, \quad 1 \wedge 0 = 0, \quad 0 \wedge 1 = 0, \quad 0 \wedge 0 = 0$$

- **Logisches Oder:** $b_1 \vee b_2$

Ergebnis ist 1, falls $b_1 = 1$ **oder** $b_2 = 1$, sonst 0.

$$1 \vee 1 = 1, \quad 1 \vee 0 = 1, \quad 0 \vee 1 = 1, \quad 0 \vee 0 = 0$$

- **Logisches Nicht, Negation, Komplement:** $\neg b$

Ergebnis ist 1, falls $b = 0$. Ergebnis ist 0, falls $b = 1$.

$$\neg 1 = 0, \quad \neg 0 = 1$$

- Mit diesen drei Grundoperationen können **alle möglichen Operationen** auf Bits definiert werden.

- Die Variablen b , b_1 , b_2 stehen für Bits.

Jede Operation auf zwei Bits ist durch ihre Wertetabelle bestimmt. Es gibt 16 mögliche Wertetabellen. Die Wertetabelle umfasst vier Bits.

| b_1 | b_2 | $f(b_1, b_2)$ | f_8 | f_{11} |
|-------|-------|---------------|-------|----------|
| 0 | 0 | | 0 | 1 |
| 0 | 1 | | 0 | 1 |
| 1 | 0 | | 0 | 0 |
| 1 | 1 | | 1 | 1 |

Aufgabe

Schreibe f_8 und f_{11} mit Hilfe von Und, Oder, Nicht.

Auflösung

$$f_8(b_1, b_2) = b_1 \wedge b_2$$

Auflösung

$$f_{11}(b_1, b_2) = (b_1 \wedge b_2) \vee \neg b_1 = \neg b_1 \vee b_2$$

- Rechnen mit einem Bit ist zu ineffizient.
- Die meisten klassischen Computer rechnen mit Bitvektoren der Breite 8 (ein **Byte** auch **Octet**), 16, 32 oder 64.
- Letztere heißen auch 16-Bit (bzw. 32-Bit, 64-Bit) **Worte** (bzw. Doppelworte, Quadworte). Daher auch **Wortbreite**.
- Der Aufbau des Computers (genauer gesagt, des Prozessors) ist auf eine Wortbreite ausgerichtet, die durch Bezeichnungen wie 32-Bit-Architektur bzw. 64-Bit-Architektur zum Ausdruck kommt.

Grundoperationen auf Worten

Bitweise logische Operationen



- Definiert auf Worten gleicher Breite.
- Wendet die logischen Bit-Operationen auf die entsprechenden Positionen der Argumente an.
- **Und:** $w_1 \wedge w_2$
Beispiel: $1100 \wedge 1010 = (1 \wedge 1)(1 \wedge 0)(0 \wedge 1)(0 \wedge 0) = 1000$
- **Oder:** $w_1 \vee w_2$
Beispiel: $1100 \vee 1010 = (1 \vee 1)(1 \vee 0)(0 \vee 1)(0 \vee 0) = 1110$
- **Negation:** $\neg w$
Beispiel: $\neg 10 = (\neg 1)(\neg 0) = 01$

Exkursion:
Datenrepräsentation

Werte und
Typen

Variable

Ausdrücke

- Alle Daten werden im Computer durch Bitvektoren dargestellt
- Die Interpretation des Bitvektors hängt vom angenommenen Typ ab

Exkursion:
Datenrepräsentation

Werte und
Typen

Variable

Ausdrücke

Mensch: Dezimalsystem

- Stellenwertsystem mit Basis 10: Zehn Ziffern— 0, 1, 2, 3, 4, 5, 6, 7, 8, 9
- Dezimaldarstellung einer Zahl ist Vektor von Ziffern
- Jede Stelle in der Dezimaldarstellung einer Zahl entspricht einer 10er-Potenz
- Beginnend von rechts mit 10^0

Exkursion:
Datenrepräsentation

Werte und
Typen

Variable

Ausdrücke

Beispiel

$$\begin{aligned}\underline{4711}_{10} &= 4 * 10^3 + 7 * 10^2 + 1 * 10^1 + 1 * 10^0 \\ &= 4000 + 700 + 10 + 1 \\ &= 4711\end{aligned}$$

Computer: Dual- oder Binärsystem (Gottfried Wilhelm Leibniz ~1700)

- Stellenwertsystem mit Basis **2**: **Zwei** Ziffern— 0, 1 — **eine Ziffer = ein Bit!**
- **Binärdarstellung** einer Zahl ist **Vektor von Bits**
- Jede Stelle in der Binärdarstellung einer Zahl entspricht einer **2er-Potenz**
- Beginnend von rechts mit **2^0**

Beispiel

$$\begin{aligned} \underline{101010}_2 &= 1 * 2^5 + 0 * 2^4 + 1 * 2^3 + 0 * 2^2 + 1 * 2^1 + 0 * 2^0 \\ &= 32 + 0 + 8 + 0 + 2 + 0 \\ &= 42 \end{aligned}$$

Exkursion:
Datenrepräsentation

Werte und
Typen

Variable

Ausdrücke

Programmierer: Hexadezimalsystem

- Stellenwertsystem mit Basis **16** (4 Bit pro Stelle)
16 Ziffern— 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, a, b, c, d, e, f
- Die **Hexadezimaldarstellung** ist ein Vektor von Hexadezimalziffern
- Jede Stelle in der Hexdarstellung einer Zahl entspricht einer **16er-Potenz**
- Beginnend von rechts mit **16⁰**

Beispiel

$$\begin{aligned}\underline{beef}_{16} &= \mathbf{11} * 16^3 + \mathbf{14} * 16^2 + \mathbf{14} * 16^1 + \mathbf{15} * 16^0 \\ &= 11 * 4096 + 14 * 256 + 14 * 16 + 15 \\ &= 48879\end{aligned}$$

Exkursion:
Datenrepräsentation

Werte und
Typen

Variable

Ausdrücke

- Welche natürlichen Zahlen lassen sich mit gegebener Wortbreite darstellen?

| Wortbreite | Wertebereich |
|------------|---------------------------------|
| 1 | 0... 1 |
| 2 | 0... 3 |
| 4 | 0... 15 |
| 8 | 0... 255 |
| 16 | 0... 65.535 |
| 32 | 0... 4.294.967.295 |
| 64 | 0... 18.446.744.073.709.551.615 |
| n | 0... $2^n - 1$ |

Exkursion:
Datenrepräsentation

Werte und
Typen

Variable

Ausdrücke

- Typischerweise wird die Hälfte des Bereichs für negative Zahlen verwendet

Algorithmus: Darstellung in Basis B



- Eingabe: natürliche Zahl n
- Ausgabe: Darstellung von n im Stellenwertsystem mit Basis $B \geq 2$
- Verwende als Ziffern $0, 1, \dots, B-1$
- Schreibe von **rechts nach links** in die Ausgabe

Algorithmus

- 1 Berechne $q \leftarrow n // B$ und $r \leftarrow n \% B$ (Quotient und Divisionsrest von n/B).
- 2 Schreibe den Rest r links an die Ausgabe.
- 3 Falls $q \neq 0$, weiter bei Punkt 1 mit $n \leftarrow q$.
- 4 Sonst fertig.

Exkursion:
Datenrepräsentation

Werte und
Typen

Variable

Ausdrücke

Beispiel: Darstellung in Basis B

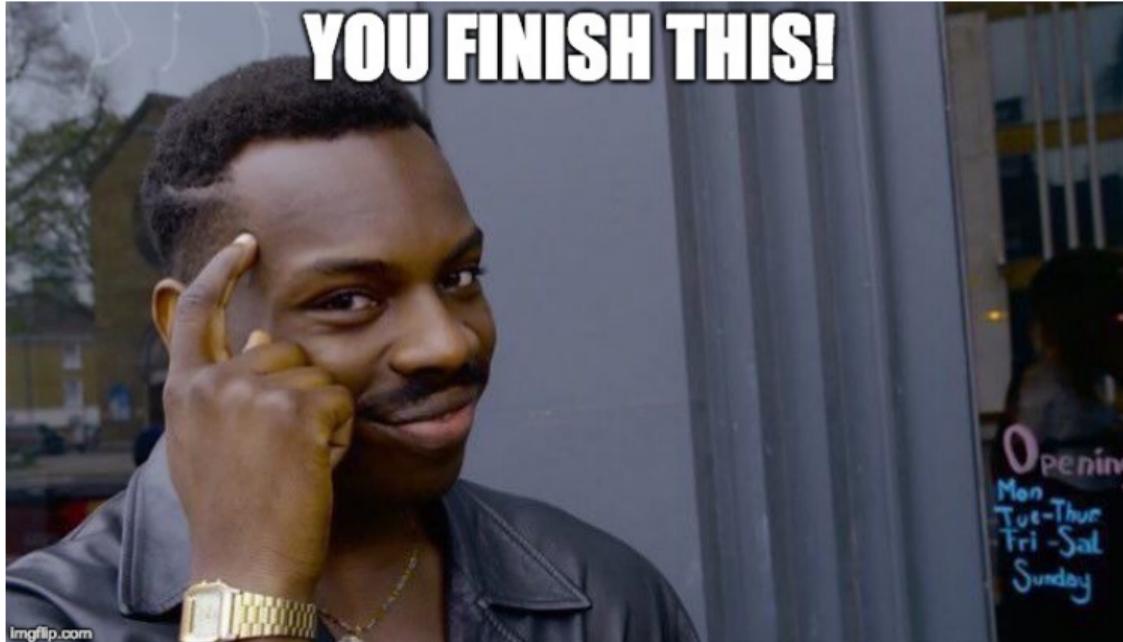
Bestimme die Binärdarstellung ($B = 2$) von $n = 42$.

- $42 // 2 = 21$ Rest **0**
- $21 // 2 = 10$ Rest **1**
- $10 // 2 = 5$ Rest **0**
- $5 // 2 = 2$ Rest **1**
- $2 // 2 = 1$ Rest **0**
- $1 // 2 = 0$ Rest **1**
- Fertig, weil $q = 0$.
- Ergebnis 101010₂
- von unten nach oben abgelesen

Addieren von Zahlen in Binärdarstellung

- Wortbreite 1: $0+0=0$; $0+1=1$; $1+0=1$; $1+1=?$
- $1+1=0$ mit **Übertrag 1**
- Damit weiter wie schriftliche Addition
- Beispiel: $42 + 6$ (in Binärdarstellung: 101010_2 und 110_2)

$$\begin{array}{r} 101010 \\ + 000110 \\ \hline 0 \end{array}$$



Exkursion:
Datenrepräsentation

Werte und
Typen

Variable

Ausdrücke

Rechnerarithmetik

- Darstellung negativer Zahlen
- Subtraktion
- Multiplikation
- Division
- und Schaltungen dafür

Zum Nachdenken

Definiere Ergebnis und Übertrag der 1-Bit Addition mit Hilfe der Grundoperationen (und, oder, nicht).

Exkursion:
Datenrepräsentation

Werte und
Typen

Variable

Ausdrücke

2 Werte und Typen



Exkursion:
Datenrepräsentation

**Werte und
Typen**

Variable

Ausdrücke

Datentypen — Syntax und Semantik

Jede Programmiersprache unterstützt verschiedene Datentypen



- **Semantik eines Datentyps**
 - Menge von **Werten** und **Operationen** auf diesen Werten.
- In einem Programmtext müssen wir diese Werte und Operationen als **Zeichenketten** aufschreiben können.
- **Syntax**
 - Ein **Literal** ist die **Darstellung** eines Wertes.
 - Ein **Operationssymbol** ist die **Darstellung** einer Operation.
- **Pragmatik**
 - Syntax und Semantik entsprechen den üblichen Konventionen bzw. mathematischen Definitionen
 - Konsistenz

Exkursion:
Datenrepräsentation

Werte und Typen

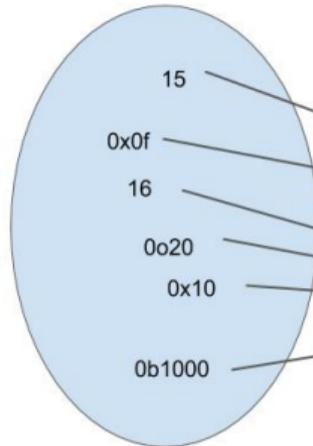
Variable

Ausdrücke

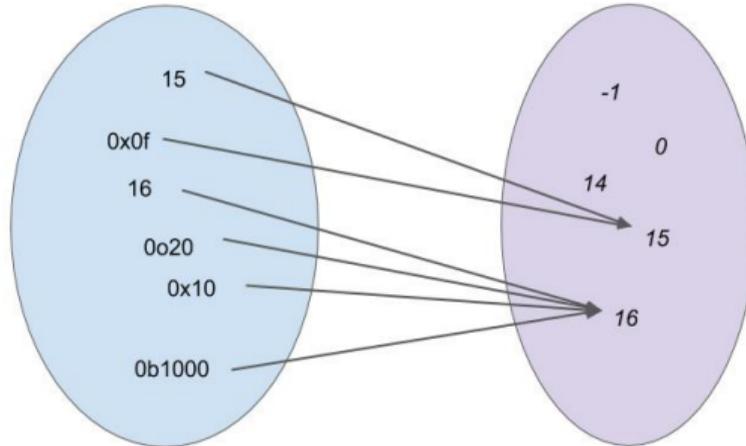
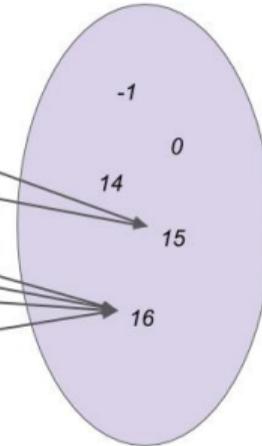
- Die ganze Zahl **16** als Wert wird z.B. durch das Literal 16 dargestellt, aber auch durch 0x10 (hexadezimale Darstellung) und 0b10000 (binäre Darstellung).
- Die Zeichenkette (der String) **nuqneH** als Wert wird durch die Literale 'nuqneH', "nuqneH" und '''nuqneH''' dargestellt.
- Die Zahl **0.2** wird durch 0.2 dargestellt, aber auch durch 2.0e-1, 0.02e1, 2000e-4 usw (Exponentialschreibweise $2.0 * 10^{-1}$).

Beispiel: int

Darstellung (Syntax)



Wertemenge (Semantik)



Exkursion:
Datenreprä-
sentation

Werte und
Typen

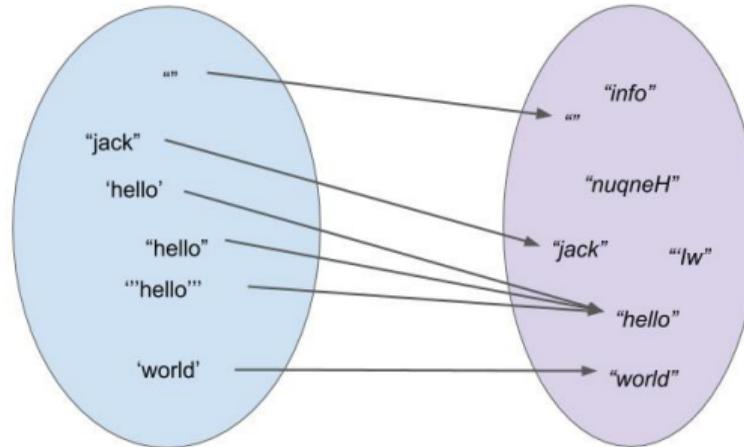
Variable

Ausdrücke

Beispiel: string

Darstellung (Syntax)

Wertemenge (Semantik)



Exkursion:
Datenrepräsentation

Werte und Typen

Variable

Ausdrücke

- In Python besteht jeder Wert aus zwei Teilen:

| Typ | Interne Repräsentation des Wertes |
|-----|-----------------------------------|
|-----|-----------------------------------|

- Die interne Repräsentation ist ein Bitvektor, der entsprechend des (Bitvektors des) Typs interpretiert wird.
- Beispiele

| | | | |
|------------|---|--------|-----------------------|
| 16 | ↔ | int | <i>0x10</i> |
| 2.24E44 | ↔ | float | <i>0x10</i> |
| 3.14159 | ↔ | float | <i>0x40490fd0</i> |
| 1078530000 | ↔ | int | <i>0x40490fd0</i> |
| "hello" | ↔ | string | <i>0x68656c6c6f00</i> |

3 Variable



Exkursion:
Datenrepräsentation

Werte und
Typen

Variable

Ausdrücke

spam.py

```
spam = 111  
print(spam)
```

Exkursion:
Datenrepräsentation

Werte und
Typen

Variable

Ausdrücke

- Eine **Zuweisung** versieht einen Wert mit einem Namen (**Variablennamen**, **Bezeichner**, **Identifier**). Dazu wird der Bezeichner (`spam`) auf der linken Seite und ein Ausdruck (`111`) auf der rechten Seite eines Gleichheitszeichens geschrieben.
- Im Beispiel: “Die *Variable* `spam` erhält den *Wert* von `111`.”
- **Syntax der Zuweisung**: *Variable = Ausdruck*
- Ausführung durch `python3 spam.py`

Variable und Speicher



UNI
FREIBURG



Exkursion:
Datenrepräsentation

Werte und
Typen

Variable

Ausdrücke

Image by https://pixabay.com/users/annaer-35513/?utm_source=link-attribution&utm_medium=referral&utm_campaign=image&utm_content=187777Anna from https://pixabay.com//?utm_source=link-attribution&utm_medium=referral&utm_campaign=image&utm_content=187777Pivahav

- Der **Zustand** eines Programms wird vollständig durch die **Belegung der Variablen** mit Werten und den aktuellen Ausführungspunkt beschrieben.

spam-egg.py

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

- Variablenbelegung nach der Ausführung:

```
Global frame  
spam | 123  
egg  | "spam"
```

Exkursion:
Datenrepräsentation

Werte und
Typen

Variable

Ausdrücke

Syntax von Bezeichnern

- Ein Bezeichner besteht aus Buchstaben, Unterstrichen und Ziffern. Das erste Zeichen darf keine Ziffer sein.

```
Brägele = 1
```

Ok

```
Kaltes Wasser = 2
```

```
~~~~~
```

```
SyntaxError: invalid syntax. Perhaps you forgot a comma?
```

```
2you = 3
```

```
^
```

```
SyntaxError: invalid decimal literal
```

```
class = 'Theory'
```

~

SyntaxError: invalid syntax

Schlüsselwörter können nicht als Bezeichner 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 | |

Exkursion:
Datenrepräsentation

Werte und
Typen

Variable

Ausdrücke

Variablennutzung vor Zuweisung



- Variablen sind erst verwendbar, nachdem ihnen ein Wert zugewiesen wurde.
- Groß-/Kleinschreibung macht einen Unterschied

```
spam = 3  
print(spam)
```

Ok. Drückt 3.

```
egg
```

```
NameError: name 'egg' is not defined
```

```
Spam
```

```
NameError: name 'Spam' is not defined. Did you mean: 'spam'?
```

Exkursion:
Datenrepräsentation

Werte und
Typen

Variable

Ausdrücke

4 Ausdrücke



Exkursion:
Datenrepräsentation

Werte und
Typen

Variable

Ausdrücke

- Wir kennen bereits **Operatoren** auf Zahlen: $+$, $-$, $*$, \dots
- **Ausdrücke** werden aus Operatoren, Literalen und Variablen zusammengesetzt.
- Die **Auswertung eines Ausdrucks** liefert einen Wert oder bricht mit einer Fehlermeldung ab:
 - Variable ist nicht definiert
 - Operator ist auf Kombination von Werten nicht definiert (Bsp: $1/0$)

Exkursion:
Datenrepräsentation

Werte und
Typen

Variable

Ausdrücke

Auswertung von Ausdrücken

Operatorpräzedenz (Operatorrangfolge)



- Die **Auswertung eines Ausdrucks** beginnt bei den Literalen und Variablen.
- Wenn die Werte der Teilausdrücke vorliegen, wird die durch den Operator bezeichnete Operation auf sie angewendet.
- Bei arithmetischen Ausdrücken gelten die üblichen **Präzedenzregeln**:
 - zuerst die Klammerung,
 - dann die Exponentiation (rechtsassoziativ!),
 - dann Multiplikation und Division,
 - dann Addition und Subtraktion,
 - bei gleicher Präzedenz wird von links nach rechts geklammert (linksassoziativ), außer bei Exponentiation

Exkursion:
Datenrepräsentation

Werte und
Typen

Variable

Ausdrücke

Arithmetische Ausdrücke: Beispiele



```
spam = 3
print (3 * 1 ** spam)
# 3
print ((3 * 1) ** spam)
# 27
print (2 * spam - 1 // 2)
# 6
print (spam ** spam ** spam)
# 7625597484987
print ((spam ** spam) ** spam)
# 19683
```

Exkursion:
Datenrepräsentation

Werte und
Typen

Variable

Ausdrücke

- Strings verketteten mit dem Operator '+' (**Konkatenation**)

```
print ('spam' + 'egg')  
# spamegg  
assert 'spam' + 'egg' == 'spamegg'
```

- Strings mit ganzen Zahlen multiplizieren (Python spezifisch)

```
print (3 * 'spam')  
# spamspamspam  
assert 3 * 'spam' == 'spamspamspam'  
print (0 * 'spam')  
#  
assert 0 * 'spam' == ''  
print (-2 * 'spam')  
#  
assert -2 * 'spam' == ''
```

Exkursion:
Datenrepräsentation

Werte und
Typen

Variable

Ausdrücke

- Auf der rechten Seite einer Zuweisung dürfen Ausdrücke auftreten:

```
spam = 42
egg = spam // 7
print (egg)
# 6
```

- Erst wird die rechte Seite ausgewertet, dann an die Variable zugewiesen:

```
spam = 42
spam = spam * 2
print (spam)
# 84
```

Exkursion:
Datenrepräsentation

Werte und
Typen

Variable

Ausdrücke

- Ein Datentyp ist bestimmt durch eine Menge von **Werten** und **Operationen** auf diesen Werten (**Semantik**).
- **Literale** sind die **Darstellung** von Werten eines Datentyps als **Zeichenkette** (**Syntax**).
- Jeder Wert hat einen bestimmten **Typ**.
- Werte erhalten durch **Zuweisung** einen Namen (**Variable**).
- Der Wert einer Variablen kann sich ändern.
- **Ausdrücke** werden aus Operatoren, Literalen und Variablen gebildet.
- Sie haben einen Wert!
- Eine Zuweisung wertet erst die rechte Seite aus und weist dann den Wert zu!

Exkursion:
Datenrepräsentation

Werte und
Typen

Variable

Ausdrücke