# Datentypen II

# Python3 – Datentypen II

nochmal: ganze Zahlen

- Unterstrich erlaubt
- beliebige Stellen
- Übersichtlichkeit

```
>>> 2_400_000
2400000

>>> 12_34
1234
```

# Python3 – Datentypen II

Angabe von Zahlen in allen Basen 2-36

- Notation:  int( "Zahl", Basis )

- Zahl als String, weil Buchstaben möglich
- Basis als Ganzzahl

# Python3 – Datentypen II

Zahlensysteme

- beliebige Basis möglich

```
>>> int("14", 6)
10
```

- Basen 2-36, Ziffern größer 9 als Buchstaben A-Z

- nur andere Schreibweise, intern stets gleich

# Python3 – Datentypen II

Zahlensysteme mit eigener Notation

- Präfix "0b" für Binärzahlen (Basis 2)

```
>>> 0b101 + 2
7
>>> 0b101 + 0b11
8
```

- Präfix "0o" für Oktalzahlen (Basis 8)

```
>>> 0o4 + 0o4
8
```

- Präfix "0x" für Hexadezimalzahlen (Basis 16)

```
>>> 0x11
17
```

# Python3 – Datentypen II

Was ergibt folgender Ausdruck?

```
>>> 0b101 + 0o10 + 0x1A + int("11", 6) + int("D",22)
```

a) 32

b) 104

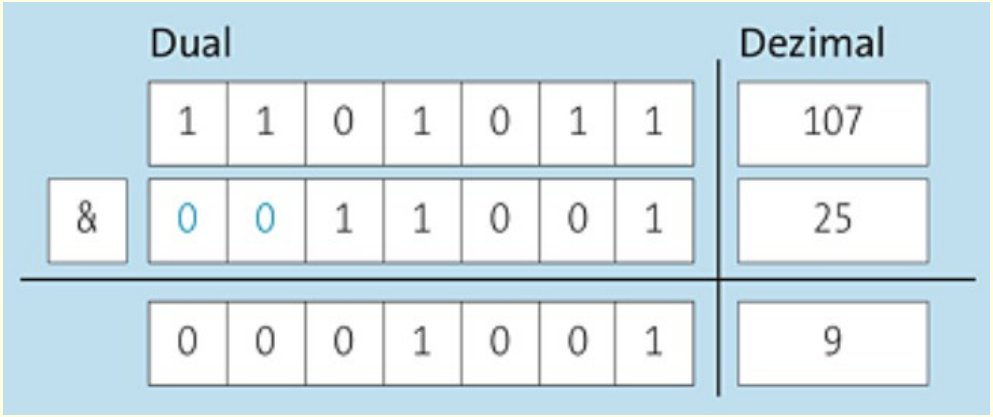
c) 72

d) 59

# Python3 – Datentypen II

Bitoperationen

- s. Tabelle 11.5 im Buch



| | Dual | | | | | | | Dezimal |
|---|---|---|---|---|---|---|---|---|
| | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 107 |
| & | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 25 |
| | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 9 |

- Beispiel  UND

- interessant für maschinennahe Programmierung

# Python3 – Datentypen II

## Kommazahlen

- Punkt, nicht Komma
- obligatorisch

```
>>> -3.
-3.0
>>> .001
0.001
```

# Python3 – Datentypen II

Gleitkommazahlen, Exponentialschreibweise

- "e" oder "E" trennt Mantisse von Exponent
- Mantisse x $10^{\text{Exponent}}$

```
>>> 7.23e3
7230.0
>>> 7.23e-3
0.00723
```

# Python3 – Datentypen II

Gleitkommazahlen, Genauigkeit

- nicht unendlich genau
- angenähert

```
>>> 1.2 + 2.2
3.4000000000000004
```

# Python3 – Datentypen II

Gleitkommazahlen, Genauigkeit

- nicht unendlich genau
- angenähert

```
>>> 1.2 + 2.2
3.4000000000000004
```

- genauere Typen in Bibliotheken verfügbar
- weitere Info
  `https://docs.python.org/2/tutorial/floatingpoint.html`

# Python3 – Datentypen II

Gleitkommazahlen, Genauigkeit

- nach oben und unten beschränkt (nicht wie int)

```
>>> import sys

>>> sys.float_info.max

1.7976931348623157e+308
```

# Python3 – Datentypen II

Gleitkommazahlen, Genauigkeit

- nach oben und unten beschränkt (nicht wie int)
- bei Überschreitung: inf

```
>>> 3.0e999
inf
>>> -3.0e999
-inf
>>> 2.0e999 < 5.0
False
>>> 2.0e999 > 5.0
True
>>> 3.0e999 == 7.0e9999999999
True
```

# Python3 – Datentypen II

Gleitkommazahlen, Genauigkeit

- bei Rechenoperationen unklar, ob Ergebnis darstellbar
- im Zweifelsfall: nan (not a number)

```
>>> -3.0e999 + 3.0e999
nan
>>> 3.0e999 + 3.0e999
inf
>>> -3.0e999 - 3.0e999
-inf
>>> -3.0e999 + 3.0e997
nan
```

# Python3 – Datentypen II

Gleitkommazahlen, Genauigkeit

- weder inf noch nan sind Konstanten
- Werte können mit

  float() erzeugt werden

```
>>> float("inf")
inf
>>> float("nan")
nan
```

# Python3 – Datentypen II

## Boolesche Werte

True oder 1

False oder 0

```
>>> True + False
1
>>> True + True
2
>>> True - True
0
```

# Python3 – Datentypen II

- Boolesche Werte  -- Verknüpfungen
- vor allem für Bedingungen
- wichtigste Operationen:

  logische
- Negierung: not

```
>>> not True
False
>>> not False
True
>>> not 1
False
```

# Python3 – Datentypen II

## Boolesche Werte -- Verknüpfungen

- and:  nur True wenn beide True
- or:  nur False wenn beide False
- Wahrheitswertetabellen bekannt?

```
>>> True and True
True

>>> True and False
False

>>> False and False
False

>>> False and True
False
```

# Python3 – Datentypen II

## Boolesche Werte -- Verknüpfungen

- Verknüpfung beliebig vieler Werte

```
1 (a or b) and (c and not d)
```

- Auswertung: links nach rechts
- Klammerung möglich

# Python3 – Datentypen II

## Boolesche Werte -- Verknüpfungen

- Verknüpfung beliebig vieler Werte

```
1 (a or b) and (c and not d)
```

- Auswertung: links nach rechts
- Klammerung möglich

# Python3 – Datentypen II

Boolesche Werte nicht boolescher Typen

- über Konvertierungsfunktion bool()
- automatisch bei Verwendung eines logischen Operators:
- stets ein Wert False
- alle anderen True
- s. Tabelle 11.10

```
>>> "" or True
True
>>> not ""
True
>>> not "abc"
False
```

# Python3 – Datentypen II

## Komplexe Zahlen

- Realteil und Imaginärteil (mit j)
- Vergleiche nur == und !=
- Attribute real und imag

```
>>> x = 3 + 4j
>>> x.real
3.0
>>> x.imag
4.0
```

# Python3 – Datentypen II

## Komplexe Zahlen

- tauchen nur selten auf

```
>>> x = 3 + 4j
>>> x.real
3.0
>>> x.imag
4.0
```