

Programmierung in Python

Inhaltsverzeichnis



- ▼ 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Verarbeitungsschlüsselelemente
- ▼ 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
- ▼ 3 Grundlegende Datenstrukturen
 - ▼ 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können verändert werden
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutablen Objekte
 - 3.3 Ausgabe mit `print`
 - ▼ 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`$\inf`)
 - 3.4.5 Not-a-Number (`NaN`)
 - ▼ 3.5 Mathematische Operationen
 - 3.5.1 Arithmetische Operationen
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
 - 3.6 Umwandlung des Datentyps
 - ▼ 3.7 Rundung
 - 3.7.1 Runden mit `round`

· Prof. Dr. Martin Hepp, Universität der Bundeswehr München

heit 1: Erste Schritte in Python

on: 2019-11-13

[/www.ebusiness-unibw.org/wiki/Teaching/PIP](http://www.ebusiness-unibw.org/wiki/Teaching/PIP)

Martin Hepp, martin.hepp@unibw.de

1 Syntaktische Konventionen

1.1 Keine Zeilenummern

Inhaltsverzeichnis

- ☞ 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Verweisungssyntax
- ☞ 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
- ☞ 3 Grundlegende Datenstrukturen
 - 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können verändert werden
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutablen Objekte
 - 3.3 Ausgabe mit `print`
 - 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`$\inf`)
 - 3.4.5 Not-a-Number (`NaN`)
 - 3.5 Mathematische Operationen
 - 3.5.1 Arithmetische Operationen
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
 - 3.6 Umwandlung des Datentyps
 - 3.7 Rundung
 - 3.7.1 Runden mit `round()`

Martin Hepp, martin.hepp@unibw.de

1 Syntaktische Konventionen

1.1 Keine Zeilenummern

Ohne Zeilenummern: (Python etc.)

```
a = 1  
b = 2  
print(a)
```

Inhaltsverzeichnis

- » 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Verweisungssyntax
- » 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
- » 3 Grundlegende Datenstrukturen
 - » 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können verändert werden
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutablen Objekte
 - 3.3 Ausgabe mit `print`
 - » 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`$\inf`)
 - 3.4.5 Not-a-Number (`NaN`)
 - » 3.5 Mathematische Operationen
 - 3.5.1 Arithmetische Operationen
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
 - 3.6 Umwandlung des Datentyps
 - » 3.7 Rundung
 - 3.7.1 Runden mit `round`

1 Syntaktische Konventionen

1.1 Keine Zeilenummern

Ohne Zeilenummern: (Python etc.)

```
a = 1
b = 2
print(a)
```

Mit Zeilenummern: (nur in älteren Sprachen)

```
10    a = 1
20    b = 2
30    print(a)
```

1.2 Groß-/Kleinschreibung

Inhaltsverzeichnis

- ⌚ 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Verweisungsschlüsselwörter
- ⌚ 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
- ⌚ 3 Grundlegende Datenstrukturen
 - ⌚ 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können verändert werden
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutablen Objekte
 - 3.3 Ausgabe mit `print`
 - ⌚ 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`$\inf`)
 - 3.4.5 Not-a-Number (`NaN`)
 - ⌚ 3.5 Mathematische Operationen
 - 3.5.1 Arithmetische Operationen
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
 - 3.6 Umwandlung des Datentyps
 - ⌚ 3.7 Rundung
 - 3.7.1 Runden mit `round`

Groß-/Kleinschreibung muss beachtet werden:

```
a = 10    # die Variable a wird definiert
print(A)  # die Variable A gibt es aber nicht
```

1.3 Reservierte Namen

Inhaltsverzeichnis

☞ *	
▼ 1	Syntaktische Konventionen
1.1	Keine Zeilenummern
1.2	Groß-/Kleinschreibung
1.3	Reservierte Namen
1.4	Zuweisungs- und Ver-
▼ 2	Stil und Formatierung
2.1	Namen
2.2	Leerzeichen
▼ 3	Grundlegende Datenstrukturen
3.1	Namen und Objekte
3.1.1	Alles in Python ist ein Objekt
3.1.2	Objekte können verändert werden
3.1.3	Mehrfachzuweisung
3.2	Mutable und Immutablen Objekte
3.3	Ausgabe mit <code>print</code>
3.4	Numerische Werte
3.4.1	Ganze Zahlen
3.4.2	Gleitkommazahl
3.4.3	Dezimalzahlen
3.4.4	Unendlich (<code>\$inf</code>)
3.4.5	Not-a-Number (<code>NaN</code>)
3.5	Mathematische Operationen
3.5.1	Arithmetische Operationen
3.5.2	Potenz
3.5.3	Wurzel
3.5.4	Ganzzahlige Division
3.5.5	Divisionsrest (modulo)
3.6	Umwandlung des Datentyps
3.7	Rundung
3.7.1	Runden mit <code>round()</code>

Namen, die für Befehle etc. verwendet werden, dürfen nicht als Namen für Werte oder Objekte genutzt werden.

Es gibt

- **Schlüsselwörter** für echte Sprachelemente ("keywords") und
- **Namen für vordefinierte Objekte und Methoden** ("Built-ins")

Schlüsselwörter für echte Sprachelemente ("keywords")

Inhaltsverzeichnis

- ⌚⚙️
- ▼ 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Verweisungssyntax
- ▼ 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
- ▼ 3 Grundlegende Datenstrukturen
 - ▼ 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können verschachtelt werden
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutablen Objekte
 - 3.3 Ausgabe mit `print`
- ▼ 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`inf`)
 - 3.4.5 Not-a-Number (`NaN`)
- ▼ 3.5 Mathematische Operationen
 - 3.5.1 Arithmetische Operationen
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
- 3.6 Umwandlung des Datentyps
- ▼ 3.7 Rundung
 - 3.7.1 Runden mit `round()`

In [2]: `help('keywords')`

Here is a list of the Python keywords. Enter any keyword to get more help.

False	class	from	or
None	continue	global	pass
True	def	if	raise
and	del	import	return
as	elif	in	try
assert	else	is	while
async	except	lambda	with
await	finally	nonlocal	yield
break	for	not	

Namen für vordefinierte Objekte und Methoden ("Built-ins")

Inhaltsverzeichnis

- ⌚ 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Verweisungssyntax
- ⌚ 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
- ⌚ 3 Grundlegende Datenstrukturen
 - ⌚ 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können mehrfach zuweisen
 - 3.1.3 Mehrfachzuweisung
 - ⌚ 3.2 Mutable und Immutablen Objekte
 - ⌚ 3.3 Ausgabe mit `print`
 - ⌚ 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`$inf`)
 - 3.4.5 Not-a-Number (`NaN`)
 - ⌚ 3.5 Mathematische Operationen
 - 3.5.1 Arithmetische Operationen
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
 - ⌚ 3.6 Umwandlung des Datentyps
 - ⌚ 3.7 Rundung
 - 3.7.1 Runden mit `round`

In [57]:

```
import builtins
seq = list(dir(builtins))
seq.sort()
max_len = len(max(seq, key=len))
chunks = [seq[pos:pos + 4] for pos in range(0, len(seq), 4)]
for chunk in chunks:
    print("\n".join([item.ljust(max_len + 1) for item in chunk]))
```

ArithmeticError	AssertionError	AttributeError
BaseException		
BlockingIOError	BrokenPipeError	BufferError
BytesWarning		
ChildProcessError	ConnectionAbortedError	ConnectionError
ConnectionRefusedError		
ConnectionResetError	DeprecationWarning	EOFError
Ellipsis		
EnvironmentError	Exception	False
FileExistsError		
FileNotFoundException	FloatingPointError	FutureWarning
GeneratorExit		
IOError	ImportError	ImportWarning

1.4 Zuweisungs- und Vergleichsoperator

Inhaltsverzeichnis

- ⌚ *
 - ▼ 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Vergleichsoperatoren
 - ▼ 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
 - ▼ 3 Grundlegende Datenstrukturen
 - ▼ 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können verändert werden
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutablen Objekte
 - 3.3 Ausgabe mit `print`
 - ▼ 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`inf`)
 - 3.4.5 Not-a-Number (`nan`)
 - ▼ 3.5 Mathematische Operatoren
 - 3.5.1 Arithmetische Operatoren
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
 - 3.6 Umwandlung des Datentyps
 - ▼ 3.7 Rundung
 - 3.7.1 Runden mit `round()`

Die meisten Programmiersprachen unterscheiden zwischen

- **Zuweisung** ("a soll den Wert 5 erhalten") und
- **Vergleich** ("Entspricht a dem Wert 5?") von Werten und Ausdrücken.

1.4 Zuweisungs- und Vergleichsoperator

Inhaltsverzeichnis

- ⌚ *
 - ▼ 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Vergleichsoperatoren
 - ▼ 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
 - ▼ 3 Grundlegende Datenstrukturen
 - ▼ 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können verändert werden
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutablen Objekte
 - 3.3 Ausgabe mit `print`
 - ▼ 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`inf`)
 - 3.4.5 Not-a-Number (`nan`)
 - ▼ 3.5 Mathematische Operatoren
 - 3.5.1 Arithmetische Operatoren
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
 - 3.6 Umwandlung des Datentyps
 - ▼ 3.7 Rundung
 - 3.7.1 Runden mit `round`

Die meisten Programmiersprachen unterscheiden zwischen

- **Zuweisung** ("a soll den Wert 5 erhalten") und
- **Vergleich** ("Entspricht a dem Wert 5?") von Werten und Ausdrücken.

```
In [3]: # Python  
  
# Zuweisung  
a = 5  
# Vergleich  
# Entspricht a dem Wert 5?  
print(a == 5)
```

True

2 Stil und Formatierung

2.1 Namen

Namen für Werte (in anderen Programmiersprachen "Variablen") sollten aussagekräftig und ohne Umlaute gewählt werden.

```
dauer = 5  
zins = 0.01
```

Inhaltsverzeichnis

- » 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Verweisungsschlüsselwörter
- » 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
- » 3 Grundlegende Datenstrukturen
 - » 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können verschachtelt sein
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutablen Objekte
 - 3.3 Ausgabe mit `print`
- » 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`$\inf`)
 - 3.4.5 Not-a-Number (`$nan`)
- » 3.5 Mathematische Operationen
 - 3.5.1 Arithmetische Operationen
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
- 3.6 Umwandlung des Datentyps
- » 3.7 Rundung
 - 3.7.1 Runden mit `round()`

Wenn der Name aus mehreren Wörtern besteht, werden diese durch einen Unterstrich (_) verbunden:

```
dauer_in_jahren = 5
```

Variablenamen sollten stets in **Kleinbuchstaben** sein.

Inhaltsverzeichnis



- ▼ 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Verweisungsschlüsselwörter
- ▼ 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
- ▼ 3 Grundlegende Datenstrukturen
 - ▼ 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können verändert werden
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutablen Objekte
 - 3.3 Ausgabe mit `print`
- ▼ 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`inf`)
 - 3.4.5 Not-a-Number (`nan`)
- ▼ 3.5 Mathematische Operationen
 - 3.5.1 Arithmetische Operationen
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
- 3.6 Umwandlung des Datentyps
- ▼ 3.7 Rundung
 - 3.7.1 Runden mit `round()`

Wenn der Name aus mehreren Wörtern besteht, werden diese durch einen Unterstrich (_) verbunden:

```
dauer_in_jahren = 5
```

Variablenamen sollten stets in **Kleinbuchstaben** sein.

Für Konstanten verwendet man dagegen Namen in Großbuchstaben

```
PI = 3.1415
```

```
ABSOLUTER_NULLPUNKT = -273.15 # Grad Celsius
```

2.2 Leerzeichen

Inhaltsverzeichnis

- ☞ *
 - ▼ 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Ver-
 - ▼ 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
 - ▼ 3 Grundlegende Datenstrukturen
 - ▼ 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können benannt werden
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutablen Objekte
 - 3.3 Ausgabe mit `print`
 - ▼ 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`inf`)
 - 3.4.5 Not-a-Number (`nan`)
 - ▼ 3.5 Mathematische Operationen
 - 3.5.1 Arithmetische Operationen
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
 - 3.6 Umwandlung des Datentyps
 - ▼ 3.7 Rundung
 - 3.7.1 Runden mit `round`

Vor und nach Operanden wie + oder - gehört jeweils ein Leerzeichen

```
zins = 1 + 0.02
```

Unnötige Einrückungen sind nicht erlaubt:

```
zins = 1 + 0.02
```

```
zinseszins = guthaben * (1 + zins) ** 4
```

-----^
 ^
 ^
 ^

Stilistische Konventionen

Inhaltsverzeichnis

☞ *	
▼ 1	Syntaktische Konventionen
1.1	Keine Zeilenummern
1.2	Groß-/Kleinschreibung
1.3	Reservierte Namen
1.4	Zuweisungs- und Ver-
▼ 2	Stil und Formatierung
2.1	Namen
2.2	Leerzeichen
▼ 3	Grundlegende Datenstrukturen
3.1	Namen und Objekte
3.1.1	Alles in Python ist ein Objekt
3.1.2	Objekte können verschachtelt werden
3.1.3	Mehrfachzuweisung
3.2	Mutable und Immutablen Objekte
3.3	Ausgabe mit <code>print</code>
3.4	Numerische Werte
3.4.1	Ganze Zahlen
3.4.2	Gleitkommazahl
3.4.3	Dezimalzahlen
3.4.4	Unendlich (<code>\$inf</code>)
3.4.5	Not-a-Number (<code>NaN</code>)
3.5	Mathematische Operationen
3.5.1	Arithmetische Operationen
3.5.2	Potenz
3.5.3	Wurzel
3.5.4	Ganzzahlige Division
3.5.5	Divisionsrest (modulo)
3.6	Umwandlung des Datentyps
3.7	Rundung
3.7.1	Runden mit <code>round()</code>

- Keine sonstigen unnötigen Leerzeichen, besonders nicht am Zeilenende.
- Unnötige Leerzeilen nur sparsam verwenden.
- Es gibt noch weitere stilistische Konventionen:
 - [PEP 8](#)
 - [Google Python Styleguide](#)

3 Grundlegende Datenstrukturen

Inhaltsverzeichnis



- ▼ 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Verarbeitungsschlüsselwörter
- ▼ 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
- ▼ 3 Grundlegende Datenstrukturen
 - ▼ 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können mehrfachzuweisen
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutablen Objekte
 - 3.3 Ausgabe mit `print`
- ▼ 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`$\inf`)
 - 3.4.5 Not-a-Number (`NaN`)
- ▼ 3.5 Mathematische Operationen
 - 3.5.1 Arithmetische Operationen
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
- 3.6 Umwandlung des Datentyps
- ▼ 3.7 Rundung
 - 3.7.1 Runden mit `round()`

- Alles in Python ist genaugenommen ein Objekt - jeder Wert, jede Funktion, jedes Unterprogramm etc.
- Alle Objekte, also auch Werte liegen irgendwo im Arbeitsspeicher des Computers.
- Die Position nennt man die **Adresse**. Sie entspricht der Nummer der Speicherzelle, an der die Daten abgelegt sind, die das Objekt repräsentieren.

3.1 Namen und Objekte

3.1.1 Alles in Python ist ein Objekt

- Objekte können, müssen aber keinen Namen haben.

```
print("Hallo Welt")
```

```
print(42)
```

- Hier haben die Zeichenfolge "Hallo Welt" und die Zahl 42 keine Namen, sind aber trotzdem Objekte mit einer Adresse.

Die Adresse eines Objektes im Speicher kann man mit der Funktion `id(name)` zeigen:

Inhaltsverzeichnis

- ⌚⚙️
- ▼ 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Verweis
- ▼ 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
- ▼ 3 Grundlegende Datenstrukturen
 - ▼ 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können mehrfach zuweisen
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutablen Objekte
 - 3.3 Ausgabe mit `print`
- ▼ 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`$inf`)
 - 3.4.5 Not-a-Number (`NaN`)
- ▼ 3.5 Mathematische Operationen
 - 3.5.1 Arithmetische Operationen
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
- 3.6 Umwandlung des Datentyps
- ▼ 3.7 Rundung
 - 3.7.1 Runden mit `round()`

Die Adresse eines Objektes im Speicher kann man mit der Funktion `id(name)` zeigen:

```
In [4]: print(type("Hallo Welt"), type(42))  
print(id("Hallo Welt"), id(42))
```

```
<class 'str'> <class 'int'>  
4579809392 4541229456
```

Inhaltsverzeichnis

- ⚙️
- ▼ 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Verweisungssyntax
- ▼ 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
- ▼ 3 Grundlegende Datenstrukturen
 - ▼ 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können mehrfach zuweisen
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutablen Objekte
 - 3.3 Ausgabe mit `print`
 - ▼ 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`$inf`)
 - 3.4.5 Not-a-Number (`$nan`)
 - ▼ 3.5 Mathematische Operationen
 - 3.5.1 Arithmetische Operationen
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
 - 3.6 Umwandlung des Datentyps
 - ▼ 3.7 Rundung
 - 3.7.1 Runden mit `round`

Die Adresse eines Objektes im Speicher kann man mit der Funktion `id(name)` zeigen:

```
In [4]: print(type("Hallo Welt"), type(42))
print(id("Hallo Welt"), id(42))

<class 'str'> <class 'int'>
4579809392 4541229456
```

- `str` und `int` sind die Typen der Objekte
- `str/String` = Zeichenkette und `int/Integer` = Ganzzahl
- Die Zahlen darunter sind die Adressen des Objektes.

3.1.2 Objekte *können* Namen haben

Inhaltsverzeichnis

- ⌚⚙️
- ▼ 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Verweisungsschlüsselwörter
- ▼ 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
- ▼ 3 Grundlegende Datenstrukturen
 - ▼ 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte *können* Namen haben
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutablen Objekte
 - 3.3 Ausgabe mit `print`
- ▼ 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`$\inf`)
 - 3.4.5 Not-a-Number (`NaN`)
- ▼ 3.5 Mathematische Operationen
 - 3.5.1 Arithmetische Operationen
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
- 3.6 Umwandlung des Datentyps
- ▼ 3.7 Rundung
 - 3.7.1 Runden mit `round()`

```
In [5]: mein_text = "Hallo Welt"  
meine_zahl = 42
```

3.1.2 Objekte *können* Namen haben

Inhaltsverzeichnis

- ⌚⚙️
- ▼ 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Verweisungsschlüsselwörter
- ▼ 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
- ▼ 3 Grundlegende Datenstrukturen
 - ▼ 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte *können* Namen haben
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutablen Objekte
 - 3.3 Ausgabe mit `print`
- ▼ 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`$\inf`)
 - 3.4.5 Not-a-Number (`NaN`)
- ▼ 3.5 Mathematische Operationen
 - 3.5.1 Arithmetische Operationen
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
- 3.6 Umwandlung des Datentyps
- ▼ 3.7 Rundung
 - 3.7.1 Runden mit `round()`

```
In [5]: mein_text = "Hallo Welt"  
meine_zahl = 42
```

Diese Namen verweisen auf die **Adresse** des Objektes:

3.1.2 Objekte können Namen haben

Inhaltsverzeichnis

- ⌚ 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Verweisungsschlüsselwörter
- ⌚ 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
- ⌚ 3 Grundlegende Datenstrukturen
 - ⌚ 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können benannt werden
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutablen Objekte
 - 3.3 Ausgabe mit `print`
- ⌚ 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`$\inf`)
 - 3.4.5 Not-a-Number (`NaN`)
- ⌚ 3.5 Mathematische Operationen
 - 3.5.1 Arithmetische Operationen
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
- ⌚ 3.6 Umwandlung des Datentyps
- ⌚ 3.7 Rundung
 - 3.7.1 Runden mit `round`

```
In [5]: mein_text = "Hallo Welt"  
meine_zahl = 42
```

Diese Namen verweisen auf die **Adresse** des Objektes:

```
In [117]: print(id(mein_text))  
print(id(meine_zahl))
```

```
4579625328  
4541229456
```

Inhaltsverzeichnis

- ▼ 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Verarbeitungsschlüsselwörter
- ▼ 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
- ▼ 3 Grundlegende Datenstrukturen
 - ▼ 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können verändert werden
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutablen Objekte
 - 3.3 Ausgabe mit `print`
- ▼ 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`$\inf`)
 - 3.4.5 Not-a-Number (`NaN`)
- ▼ 3.5 Mathematische Operationen
 - 3.5.1 Arithmetische Operationen
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
- 3.6 Umwandlung des Datentyps
- ▼ 3.7 Rundung
 - 3.7.1 Runden mit `round()`

Das ist ein wesentlicher Unterschied zu anderen Programmiersprachen. In Python führt eine Anweisung wie

```
variable = 1234
```

nicht dazu, dass eine Variable **erzeugt** wird, die mit dem Wert 1234 **initial gefüllt** wird.

Inhaltsverzeichnis

- ☞ * **1** Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Verweisoperator
- ☞ **2** Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
- ☞ **3** Grundlegende Datenstrukturen
 - 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können verändert werden
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutablen Objekte
 - 3.3 Ausgabe mit `print`
 - 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`inf`)
 - 3.4.5 Not-a-Number (`NaN`)
 - 3.5 Mathematische Operationen
 - 3.5.1 Arithmetische Operationen
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
 - 3.6 Umwandlung des Datentyps
 - 3.7 Rundung
 - 3.7.1 Runden mit `round()`

Stattdessen wird geprüft, ob es das Objekt der Zahl 1234 schon im Speicher gibt. Falls nicht, wird eines im Speicher erzeugt. Dann wird die Adresse dieses Objektes als Verweis dem Namen `variable` zugewiesen, also damit verbunden.

Der Name `variable` wird also mit dem Objekt/Wert verbunden.

[vgl. Fredrik Lundh: Call by Object]

Mehrere Anweisungen wie

```
zahl_1 = 42  
zahl_2 = 42  
zahl_3 = 42
```

führen in der Regel (*) daher nicht dazu, dass drei Variablen erzeugt werden, sondern dass drei Namen definiert werden, über die man die Ganzzahl 42 ansprechen kann.

(*) Im Detail hängt das davon ab, ob Python schnell feststellen kann, ob es diesen Wert schon im Speicher gibt.

Inhaltsverzeichnis

- ⌚ ⓘ ▾ 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Verarbeitungsoperatoren
- ▴ 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
- ▴ 3 Grundlegende Datenstrukturen
 - ▴ 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können verändert werden
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutable Objekte
 - 3.3 Ausgabe mit `print`
- ▴ 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`inf`)
 - 3.4.5 Not-a-Number (`nan`)
- ▴ 3.5 Mathematische Operationen
 - 3.5.1 Arithmetische Operationen
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
- 3.6 Umwandlung des Datentyps
- ▴ 3.7 Rundung
 - 3.7.1 Runden mit `round`

3.1.3 Mehrfachzuweisung

Inhaltsverzeichnis

- ☞ 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Verarbeitungsoperatoren
- ☞ 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
- ☞ 3 Grundlegende Datenstrukturen
 - 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können mehrfach zugeordnet werden
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutablen Objekte
 - 3.3 Ausgabe mit `print`
- ☞ 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`$inf`)
 - 3.4.5 Not-a-Number (`NaN`)
- ☞ 3.5 Mathematische Operationen
 - 3.5.1 Arithmetische Operationen
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
- 3.6 Umwandlung des Datentyps
- ☞ 3.7 Rundung
 - 3.7.1 Runden mit `round()`

```
In [7]: a = b = c = 3
```

Man kann übrigens auch in einer Anweisung mehrere Namen für dasselbe Objekt definieren:

3.1.3 Mehrfachzuweisung

Inhaltsverzeichnis

- » 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Ver-
- » 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
- » 3 Grundlegende Datenstrukturen
 - » 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können mehrfach definiert werden
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutab - 3.3 Ausgabe mit `print`
- » 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`inf`)
 - 3.4.5 Not-a-Number (`nan`)
- » 3.5 Mathematische Operationen
 - 3.5.1 Arithmetische Operatoren
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
- 3.6 Umwandlung des Datentyps
- » 3.7 Rundung
 - 3.7.1 Runden mit `round()`

Man kann übrigens auch in einer Anweisung mehrere Namen für dasselbe Objekt definieren:

```
In [7]: a = b = c = 3
```

Verständnischeck: Wenn wir nun

```
b = 4
```

ausführen, was passiert?

In [8]:

```
a = b = c = 3
print(a, b, c)
b = 4
print(a, b, c)
```

3 3 3

3 4 3

Nur der Wert von `b` ändert sich, weil die Verweise der anderen Namen nicht berührt werden.

Inhaltsverzeichnis

- ⚙️
- ▼ 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Verweisoperatoren
- ▼ 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
- ▼ 3 Grundlegende Datenstrukturen
 - ▼ 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können verändert werden
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutablen Objekte
 - 3.3 Ausgabe mit `print`
- ▼ 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`$\inf`)
 - 3.4.5 Not-a-Number (`NaN`)
- ▼ 3.5 Mathematische Operationen
 - 3.5.1 Arithmetische Operationen
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
- 3.6 Umwandlung des Datentyps
- ▼ 3.7 Rundung
 - 3.7.1 Runden mit `round`

3.2 Mutable und Immutable Objects

Inhaltsverzeichnis

- ⌚ ⓘ
 - ▼ 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Verweisungssyntax
 - ▼ 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
 - ▼ 3 Grundlegende Datenstrukturen
 - ▼ 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können verändert werden
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutable Objekte
 - 3.3 Ausgabe mit `print`
 - ▼ 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`$\inf`)
 - 3.4.5 Not-a-Number (`NaN`)
 - ▼ 3.5 Mathematische Operationen
 - 3.5.1 Arithmetische Operationen
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
 - 3.6 Umwandlung des Datentyps
 - ▼ 3.7 Rundung
 - 3.7.1 Runden mit `round()`

Es gibt in Python Objekte,

- die man verändern kann ("**Mutable Objects**"), und
- solche, die unveränderlich sind ("**Immutable Objects**").

Zahlen und Zeichenketten sind unveränderlich.

Das heißt aber nicht, dass man den Wert von Variablen dieser Typen nicht ändern könnte:

Inhaltsverzeichnis

- ⌚⚙️
- ▼ 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Verweis
- ▼ 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
- ▼ 3 Grundlegende Datenstrukturen
 - ▼ 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können verändert werden
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutablen Objekte
 - 3.3 Ausgabe mit `print`
- ▼ 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`$\inf`)
 - 3.4.5 Not-a-Number (`NaN`)
- ▼ 3.5 Mathematische Operationen
 - 3.5.1 Arithmetische Operationen
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
- 3.6 Umwandlung des Datentyps
- ▼ 3.7 Rundung
 - 3.7.1 Runden mit `round()`

Das heißt aber nicht, dass man den Wert von Variablen dieser Typen nicht ändern könnte:

Inhaltsverzeichnis

- ⚙️
- ▼ 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Verweisungssyntax
- ▼ 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
- ▼ 3 Grundlegende Datenstrukturen
 - ▼ 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können verändert werden
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutablen Objekte
 - 3.3 Ausgabe mit `print`
- ▼ 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`$\inf`)
 - 3.4.5 Not-a-Number (`NaN`)
- ▼ 3.5 Mathematische Operationen
 - 3.5.1 Arithmetische Operationen
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
- 3.6 Umwandlung des Datentyps
- ▼ 3.7 Rundung
 - 3.7.1 Runden mit `round`

```
In [9]: text = "Uni"  
        print(text)  
        text = "FH"  
        print(text)
```

Uni

FH

Das heißt aber nicht, dass man den Wert von Variablen dieser Typen nicht ändern könnte:

Inhaltsverzeichnis

- ⌚⚙️
- ▼ 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Verknüpfungsoperatoren
- ▼ 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
- ▼ 3 Grundlegende Datenstrukturen
 - ▼ 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können verändert werden
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutablen Objekte
 - 3.3 Ausgabe mit `print`
- ▼ 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`$\inf`)
 - 3.4.5 Not-a-Number (`NaN`)
- ▼ 3.5 Mathematische Operationen
 - 3.5.1 Arithmetische Operationen
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
- 3.6 Umwandlung des Datentyps
- ▼ 3.7 Rundung
 - 3.7.1 Runden mit `round`

```
In [9]: text = "Uni"  
        print(text)  
        text = "FH"  
        print(text)
```

Uni

FH

```
In [10]: zahl = 1  
        print(zahl)  
        zahl = zahl + 1  
        print(zahl)
```

1

2

Das heißt aber nicht, dass man den Wert von Variablen dieser Typ nicht ändern könnte:

Inhaltsverzeichnis

- ⌚⚙️
- ▼ 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Verarbeitungsschlüsselwörter
- ▼ 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
- ▼ 3 Grundlegende Datenstrukturen
 - ▼ 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können verändert werden
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutablen Objekte
 - 3.3 Ausgabe mit `print`
 - ▼ 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`inf`)
 - 3.4.5 Not-a-Number (`nan`)
 - ▼ 3.5 Mathematische Operationen
 - 3.5.1 Arithmetische Operationen
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
 - 3.6 Umwandlung des Datentyps
 - ▼ 3.7 Rundung
 - 3.7.1 Runden mit `round`

```
In [9]: text = "Uni"  
        print(text)  
        text = "FH"  
        print(text)
```

Uni
FH

```
In [10]: zahl = 1  
        print(zahl)  
        zahl = zahl + 1  
        print(zahl)
```

1
2

Hier wird jeweils nicht die Variable mit einem neuen **Wert** überschrieben, sondern der neue Wert als neues Objekt erzeugt und die Variable (der Name) **mit der Adresse des neuen Objektes verbunden**.

3.3 Ausgabe mit `print`

Inhaltsverzeichnis

- ⌚ *
 - ▼ 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Verweisungsschlüsselwörter
 - ▼ 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
 - ▼ 3 Grundlegende Datenstrukturen
 - ▼ 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können verändert werden
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutablen Objekte
 - 3.3 Ausgabe mit `print`
 - ▼ 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`$\inf`)
 - 3.4.5 Not-a-Number (`$\NaN`)
 - ▼ 3.5 Mathematische Operationen
 - 3.5.1 Arithmetische Operationen
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
 - 3.6 Umwandlung des Datentyps
 - ▼ 3.7 Rundung
 - 3.7.1 Runden mit `round()`

Man kann den Wert jeder Variable und jeden mathematischen Ausdruck mit dem Befehl `print(<ausdruck>)` auf dem Bildschirm anzeigen lassen:

3.3 Ausgabe mit `print`

Inhaltsverzeichnis

- ⌚ *
 - ▼ 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Ver-
 - ▼ 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
 - ▼ 3 Grundlegende Datenstrukturen
 - ▼ 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können mehrfach zuweisen
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutab
 - 3.3 Ausgabe mit `print`
 - ▼ 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`inf`)
 - 3.4.5 Not-a-Number (`nan`)
 - ▼ 3.5 Mathematische Operationen
 - 3.5.1 Arithmetische Operat
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
 - 3.6 Umwandlung des Datentyps
 - ▼ 3.7 Rundung
 - 3.7.1 Runden mit `round`

Man kann den Wert jeder Variable und jeden mathematischen Ausdruck mit dem Befehl `print(<ausdruck>)` auf dem Bildschirm anzeigen lassen:

```
In [3]: print(1 + 4)
          print('Hallo')
          print('Toll!')
```

```
5
Hallo
Toll!
```

```
In [1]: # Mehrere Werte werden
          # durch Komma getrennt
          print(1, 5 * 3, 'Hallo', 10.)
```

```
1 15 Hallo 10.3
```

3.4 Numerische Werte

Inhaltsverzeichnis

- ⌚ ⓘ ▾ 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Ver-
- ▴ 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
- ▴ 3 Grundlegende Datenstrukturen
 - ▴ 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können mehrfachzuweisen
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutab
 - 3.3 Ausgabe mit `print`
- ▴ 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`$\inf`)
 - 3.4.5 Not-a-Number (`NaN`)
- ▴ 3.5 Mathematische Operationen
 - 3.5.1 Arithmetische Operat
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
- 3.6 Umwandlung des Datentyps
- ▴ 3.7 Rundung
 - 3.7.1 Runden mit `round`

Numerische Werte, wie

- Zahlen wie 5 oder -1.23
- Mathematische Konstanten wie π oder e
- Unendlich (∞ / $-\infty$) und Not-a-Number sind die häufigsten Arten von Objekten in den meisten Programmen.

3.4.1 Ganze Zahlen

Inhaltsverzeichnis

- » 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Ver-
- » 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
- » 3 Grundlegende Datenstrukturen
 - » 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können mehrfach zuweisbar sein
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutablen
 - 3.3 Ausgabe mit `print`
- » 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`$inf`)
 - 3.4.5 Not-a-Number (`NaN`)
- » 3.5 Mathematische Operationen
 - 3.5.1 Arithmetische Operationen
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
- 3.6 Umwandlung des Datentyps
- » 3.7 Rundung
 - 3.7.1 Runden mit `round()`

Ganze Zahlen werden in Python durch den Datentyp `int` repräsentiert und können beliebig große Werte annehmen (vgl. Numeric Types — `int, float, complex`).

```
In [13]: a = 15
          b = -7
          c = 240
```

Man kann auch eine andere Basis als 10 wählen und dadurch elegant **Binärzahlen** und **Hexadezimalzahlen** erzeugen:

```
In [14]: # Binärzahlen  
a = 0b00001111  
# Hexadezimalzahlen  
c = 0xF0  
print(a, c)
```

```
15 240
```

Inhaltsverzeichnis

- ⚙️
- ▼ 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Verweisungsschlüsselwörter
- ▼ 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
- ▼ 3 Grundlegende Datenstrukturen
 - ▼ 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können mehrfach zuweisen
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutablen Objekte
 - 3.3 Ausgabe mit `print`
- ▼ 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`$\inf`)
 - 3.4.5 Not-a-Number (`$\NaN`)
- ▼ 3.5 Mathematische Operationen
 - 3.5.1 Arithmetische Operationen
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
- 3.6 Umwandlung des Datentyps
- ▼ 3.7 Rundung
 - 3.7.1 Runden mit `round`

3.4.2 Gleitkommazahlen

Inhaltsverzeichnis

- » 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Ver-
- » 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
- » 3 Grundlegende Datenstrukturen
 - » 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können mehrfach zuweisen
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutab
 - 3.3 Ausgabe mit `print`
- » 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`$\inf`)
 - 3.4.5 Not-a-Number (`NaN`)
- » 3.5 Mathematische Operationen
 - 3.5.1 Arithmetische Operat
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
- 3.6 Umwandlung des Datentyps
- » 3.7 Rundung
 - 3.7.1 Runden mit `round`

Wenn ein übergebener Wert einen Dezimalpunkt oder einen Exponenten enthält, wird daraus in Python ein Objekt vom Typ `float` erzeugt.

```
In [118]: wert_1 = float(1/3)
          print(wert_1)
          print(wert_1 * 3)
```

```
0.3333333333333333
1.0
```

3.4.2 Gleitkommazahlen

Inhaltsverzeichnis

- ⌚ 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Ver-
- ⌚ 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
- ⌚ 3 Grundlegende Datenstrukturen
 - ⌚ 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können mehrfach zuweisen
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutab
 - 3.3 Ausgabe mit `print`
- ⌚ 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`inf`)
 - 3.4.5 Not-a-Number (`nan`)
- ⌚ 3.5 Mathematische Operationen
 - 3.5.1 Arithmetische Operat
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
- ⌚ 3.6 Umwandlung des Datentyps
- ⌚ 3.7 Rundung
 - 3.7.1 Runden mit `round`

Wenn ein übergebener Wert einen Dezimalpunkt oder einen Exponenten enthält, wird daraus in Python ein Objekt vom Typ `float` erzeugt.

```
In [118]: wert_1 = float(1/3)
          print(wert_1)
          print(wert_1 * 3)
```

```
0.3333333333333333
1.0
```

```
In [120]: # Achtung, Genauigkeitprobleme
          wert_2 = wert_1 / 10000
          print((wert_2 * 10000 * 3))
```

```
0.9999999999999998
```

Bei einem Python-Objekt vom Typ `float` handelt es sich (auf fast jedem Computersystem) um eine Gleitkommazahl mit 64 Bit.

Inhaltsverzeichnis

- ⚙️
- ▼ 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Verweisungssyntax
- ▼ 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
- ▼ 3 Grundlegende Datenstrukturen
 - ▼ 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können verändert werden
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutablen Objekte
 - 3.3 Ausgabe mit `print`
- ▼ 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`$\inf`)
 - 3.4.5 Not-a-Number (`NaN`)
- ▼ 3.5 Mathematische Operationen
 - 3.5.1 Arithmetische Operationen
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
- 3.6 Umwandlung des Datentyps
- ▼ 3.7 Rundung
 - 3.7.1 Runden mit `round()`

Bei einem Python-Objekt vom Typ `float` handelt es sich (auf fast jedem Computersystem) um eine Gleitkommazahl mit 64 Bit.

Inhaltsverzeichnis

- ⌚⚙️
- ▼ 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Verweisungsschlüsselelemente
- ▼ 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
- ▼ 3 Grundlegende Datenstrukturen
 - ▼ 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können mehrfach zuweisen
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutablen Objekte
 - 3.3 Ausgabe mit `print`
- ▼ 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`$\inf`)
 - 3.4.5 Not-a-Number (`NaN`)
- ▼ 3.5 Mathematische Operationen
 - 3.5.1 Arithmetische Operationen
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
- 3.6 Umwandlung des Datentyps
- ▼ 3.7 Rundung
 - 3.7.1 Runden mit `round()`

Die Genauigkeit und der Wertebereich entsprechen daher dem, was in anderen Programmiersprachen der Typ `double` bietet.

Inhaltsverzeichnis

- ☞ * ▼ 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Verweisungssyntax
- ▼ 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
- ▼ 3 Grundlegende Datenstrukturen
 - 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können verändert werden
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutablen Objekte
 - 3.3 Ausgabe mit `print`
 - ▼ 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`$\inf`)
 - 3.4.5 Not-a-Number (`NaN`)
 - ▼ 3.5 Mathematische Operationen
 - 3.5.1 Arithmetische Operationen
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
 - 3.6 Umwandlung des Datentyps
 - ▼ 3.7 Rundung
 - 3.7.1 Runden mit `round()`

Man muss dazu wissen, dass Python in den neueren Versionen versucht, die Beschränkungen von Gleitkommazahlen bei der Ausgabe durch geschickte Rundungsregeln zu verbergen. So wird z.B. $\frac{1}{3}$ intern als eine Gleitkommazahl mit einer begrenzten Anzahl an Stellen gespeichert.

Zu den Beschränkungen und Fehlerquellen beim Rechnen mit Gleitkommazahlen vgl. [Floating Point Arithmetic: Issues and Limitations](#).

3.4.3 Dezimalzahlen

Inhaltsverzeichnis

- » 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Ver-
- » 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
- » 3 Grundlegende Datenstrukturen
 - » 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können verändert werden
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutab
 - 3.3 Ausgabe mit `print`
- » 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`inf`)
 - 3.4.5 Not-a-Number (`nan`)
- » 3.5 Mathematische Operationen
 - 3.5.1 Arithmetische Operat
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
- 3.6 Umwandlung des Datentyps
- » 3.7 Rundung
 - 3.7.1 Runden mit `round`

Wenn es wichtig ist, dass Zahlen genau in der gegebenen Genauigkeit gespeichert und verarbeitet werden, sind Dezimalzahlen mit einer festen Stellenzahl besser geeignet.

Dies betrifft insbesondere Geldbeträge.

Weitere Informationen: <https://docs.python.org/3/library/decimal.html>

3.4.4 Unendlich (∞)

Inhaltsverzeichnis

- ☞ 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Verweisungssyntax
- ☞ 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
- ☞ 3 Grundlegende Datenstrukturen
 - 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können mehrfach zuweisen
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutablen Objekte
 - 3.3 Ausgabe mit `print`
- ☞ 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (∞)
 - 3.4.5 Not-a-Number (NaN)
- ☞ 3.5 Mathematische Operationen
 - 3.5.1 Arithmetische Operationen
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
- 3.6 Umwandlung des Datentyps
- ☞ 3.7 Rundung
 - 3.7.1 Runden mit `round()`

Der Wert unendlich kann in Python auf zwei Wegen erhalten werden:

In []:

```
positive_infinity = float('inf')
negative_infinity = float('-inf')
```

3.4.4 Unendlich (∞)

Inhaltsverzeichnis

- ☞ *
 - ▼ 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Verweisungssyntax
 - ▼ 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
 - ▼ 3 Grundlegende Datenstrukturen
 - ▼ 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können verschachtelt sein
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutablen Objekte
 - 3.3 Ausgabe mit `print`
 - ▼ 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`float('inf')`)
 - 3.4.5 Not-a-Number (`float('nan')`)
 - ▼ 3.5 Mathematische Operationen
 - 3.5.1 Arithmetische Operationen
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
 - 3.6 Umwandlung des Datentyps
 - ▼ 3.7 Rundung
 - 3.7.1 Runden mit `round()`

Der Wert unendlich kann in Python auf zwei Wegen erhalten werden:

```
In [ ]: positive_infinity = float('inf')
negative_infinity = float('-inf')
```

```
In [ ]: import math
positive_infinity_2 = math.inf
negative_infinity2 = -math.inf
```

3.4.5 Not-a-Number (NaN)

Inhaltsverzeichnis

»	1 Syntaktische Konventionen
1.1 Keine Zeilenummern	
1.2 Groß-/Kleinschreibung	
1.3 Reservierte Namen	
1.4 Zuweisungs- und Ver-	
» 2 Stil und Formatierung	
2.1 Namen	
2.2 Leerzeichen	
» 3 Grundlegende Datenstrukturen	
3.1 Namen und Objekte	
3.1.1 Alles in Python ist ein Objekt	
3.1.2 Objekte können verändert werden	
3.1.3 Mehrfachzuweisung	
3.2 Mutable und Immutab	
3.3 Ausgabe mit <code>print</code>	
» 3.4 Numerische Werte	
3.4.1 Ganze Zahlen	
3.4.2 Gleitkommazahl	
3.4.3 Dezimalzahlen	
3.4.4 Unendlich (<code>inf</code>)	
3.4.5 Not-a-Number (<code>NaN</code>)	
» 3.5 Mathematische Operationen	
3.5.1 Arithmetische Operat	
3.5.2 Potenz	
3.5.3 Wurzel	
3.5.4 Ganzzahlige Division	
3.5.5 Divisionsrest (modulo)	
3.6 Umwandlung des Datentyps	
» 3.7 Rundung	
3.7.1 Runden mit <code>round</code>	

Es gibt Operationen, bei denen sich das Ergebnis nicht als reelle Zahl abspeichern lässt. Ferner kann bei der Verarbeitung eigentlich numerischer Werte durch Datenqualitätsprobleme der Fall eintreten, dass einzelne Werte keine Zahlen sind. Für diesen Fall gibt es einen besonderen Wert, der sich **NaN** für "Not a number" nennt. Beispiele:

- ∞/∞
- Quadratwurzel aus negativen Werten

```
In [ ]: not_a_number = float('NaN')
         print(100 * not_a_number)
```

Inhaltsverzeichnis



- ▼ 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Verweisungssyntax
- ▼ 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
- ▼ 3 Grundlegende Datenstrukturen
 - ▼ 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können verändert werden
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutablen Objekte
 - 3.3 Ausgabe mit `print`
 - ▼ 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`$\inf`)
 - 3.4.5 Not-a-Number (`NaN`)
 - ▼ 3.5 Mathematische Operationen
 - 3.5.1 Arithmetische Operationen
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
 - 3.6 Umwandlung des Datentyps
 - ▼ 3.7 Rundung
 - 3.7.1 Runden mit `round`

```
In [ ]: not_a_number = float('NaN')
         print(100 * not_a_number)
```

Inhaltsverzeichnis



- ▼ 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Verweisungssyntax
- ▼ 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
- ▼ 3 Grundlegende Datenstrukturen
 - ▼ 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können verändert werden
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutablen Objekte
 - 3.3 Ausgabe mit `print`
- ▼ 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`$\inf`)
 - 3.4.5 Not-a-Number (`NaN`)
- ▼ 3.5 Mathematische Operationen
 - 3.5.1 Arithmetische Operationen
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
- 3.6 Umwandlung des Datentyps
- ▼ 3.7 Rundung
 - 3.7.1 Runden mit `round`

Der wesentliche Nutzen dieses Wertes besteht darin, dass man die Ungültigkeit einer Berechnung erkennen kann.

```
In [ ]: not_a_number = float('NaN')
         print(100 * not_a_number)
```

Inhaltsverzeichnis

- ⌚ ⓘ **1** Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Verweisungssyntax
- ⌚ ⓘ **2** Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
- ⌚ ⓘ **3** Grundlegende Datenstrukturen
 - 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können verändert werden
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutablen Objekte
 - 3.3 Ausgabe mit `print`
 - 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`$\inf`)
 - 3.4.5 Not-a-Number (`NaN`)
 - 3.5 Mathematische Operationen
 - 3.5.1 Arithmetische Operationen
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
 - 3.6 Umwandlung des Datentyps
 - 3.7 Rundung
 - 3.7.1 Runden mit `round`

Der wesentliche Nutzen dieses Wertes besteht darin, dass man die Ungültigkeit einer Berechnung erkennen kann.

Hinweis: Es gibt auch einen Datentyp `None`, der immer dann zurückgeliefert wird, wenn eine Operation 'nichts' ergibt.

3.5 Mathematische Operationen

Inhaltsverzeichnis

- ⌚⚙️
- ▼ 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Ver-
- ▼ 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
- ▼ 3 Grundlegende Datenstrukturen
 - ▼ 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können verändert werden
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutab
 - 3.3 Ausgabe mit `print`
- ▼ 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`$inf`)
 - 3.4.5 Not-a-Number (`NaN`)
- ▼ 3.5 Mathematische Operationen
 - 3.5.1 Arithmetische Operat
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
- 3.6 Umwandlung des Datentyps
- ▼ 3.7 Rundung
 - 3.7.1 Runden mit `round`

Martin Hepp, martin.hepp@unibw.de

3.5.1 Arithmetische Operationen

Inhaltsverzeichnis

- ⌚⚙️
- ▼ 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Ver-
- ▼ 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
- ▼ 3 Grundlegende Datenstrukturen
 - ▼ 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können verändert werden
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutab
 - 3.3 Ausgabe mit `print`
- ▼ 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`$\inf`)
 - 3.4.5 Not-a-Number (`NaN`)
- ▼ 3.5 Mathematische Operationen
 - 3.5.1 Arithmetische Operat
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
- 3.6 Umwandlung des Datentyps
- ▼ 3.7 Rundung
 - 3.7.1 Runden mit `round`

```
In [5]: a = 1  
        b = 2  
        c = 3
```

```
In [6]: # Grundrechenarten  
        d = a + b  
        print(d)
```

3

```
In [7]: e = c - a  
       print(e)
```

2

Inhaltsverzeichnis

- ⌚⚙️
- ▼ 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Verweisungssyntax
- ▼ 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
- ▼ 3 Grundlegende Datenstrukturen
 - ▼ 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können verändert werden
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutablen Objekte
 - 3.3 Ausgabe mit `print`
 - ▼ 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`$\inf`)
 - 3.4.5 Not-a-Number (`NaN`)
 - ▼ 3.5 Mathematische Operationen
 - 3.5.1 Arithmetische Operationen
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
 - 3.6 Umwandlung des Datentyps
 - ▼ 3.7 Rundung
 - 3.7.1 Runden mit `round`

In [7]:
e = c - a
print(e)

2

In [8]: f = b * e
print(f)

4

Inhaltsverzeichnis



- ▼ 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Verweisungssyntax
- ▼ 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
- ▼ 3 Grundlegende Datenstrukturen
 - ▼ 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können verändert werden
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutablen Objekte
 - 3.3 Ausgabe mit `print`
- ▼ 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`$\inf`)
 - 3.4.5 Not-a-Number (`NaN`)
- ▼ 3.5 Mathematische Operationen
 - 3.5.1 Arithmetische Operationen
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
- 3.6 Umwandlung des Datentyps
- ▼ 3.7 Rundung
 - 3.7.1 Runden mit `round`

In [9]:

```
g = f / b
print(g)
print(5 / 2)
```

2.0

2.5

Inhaltsverzeichnis

- ⚙️
- ▼ 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Verweisungssyntax
- ▼ 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
- ▼ 3 Grundlegende Datenstrukturen
 - ▼ 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können verändert werden
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutablen Objekte
 - 3.3 Ausgabe mit `print`
- ▼ 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`$\inf`)
 - 3.4.5 Not-a-Number (`$\NaN`)
- ▼ 3.5 Mathematische Operationen
 - 3.5.1 Arithmetische Operationen
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
- 3.6 Umwandlung des Datentyps
- ▼ 3.7 Rundung
 - 3.7.1 Runden mit `round`

In [9]:

```
g = f / b
print(g)
print(5 / 2)
```

2.0

2.5

Achtung: Seit Python 3.0 ist die Standard-Division eine Gleitkommadivision, $5 / 2$ ist also 2.5. In früheren Versionen wurde standardmäßig eine ganzzahlige Division durchgeführt, also $5/2 = 2$ (Rest 1).

Inhaltsverzeichnis



- ▼ 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Ver-
- ▼ 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
- ▼ 3 Grundlegende Datenstrukturen
 - ▼ 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können verändert werden
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutab
 - 3.3 Ausgabe mit `print`
- ▼ 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`$\inf`)
 - 3.4.5 Not-a-Number (`NaN`)
- ▼ 3.5 Mathematische Operationen
 - 3.5.1 Arithmetische Operat
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
 - 3.6 Umwandlung des Datentyps
- ▼ 3.7 Rundung
 - 3.7.1 Runden mit `round`

3.5.2 Potenz

Inhaltsverzeichnis

- ☞ 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Verweisungssyntax
- ☞ 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
- ☞ 3 Grundlegende Datenstrukturen
 - 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können verschachtelt werden
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutablen Objekte
 - 3.3 Ausgabe mit `print`
- ☞ 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`$\inf`)
 - 3.4.5 Not-a-Number (`NaN`)
- ☞ 3.5 Mathematische Operationen
 - 3.5.1 Arithmetische Operationen
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
- 3.6 Umwandlung des Datentyps
- ☞ 3.7 Rundung
 - 3.7.1 Runden mit `round`

x^y in Python als `x**y`

```
In [22]: # Potenzfunktionen
          h = 2**7    # 2^7 = 128
          print(h)
```

128

3.5.3 Wurzel

Inhaltsverzeichnis

- ☞ 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Ver-
- ☞ 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
- ☞ 3 Grundlegende Datenstrukturen
 - 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können mehrfach zuweisen
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutab
 - 3.3 Ausgabe mit `print`
- ☞ 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`inf`)
 - 3.4.5 Not-a-Number (`nan`)
- ☞ 3.5 Mathematische Operationen
 - 3.5.1 Arithmetische Operatoren
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
 - 3.6 Umwandlung des Datentyps
- ☞ 3.7 Rundung
 - 3.7.1 Runden mit `round`

$$\sqrt{x} = x^{\frac{1}{2}}$$

$$\sqrt[3]{x} = x^{\frac{1}{3}}$$

Direkt in Python gibt es keine Funktion für die Quadratwurzel, wenn man dies einfach als Potenzfunktion mit einem Bruch als Exponenten ausdrücken kann:

```
In [2]: # Quadratwurzel
a = 16
print(a**(1/2))
print(a**0.5)

4.0
4.0
```

Es gibt auch ein spezielles Modul `math` mit zusätzlichen mathematischen Methoden.

Inhaltsverzeichnis



- ▼ 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Verweisungssyntax
- ▼ 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
- ▼ 3 Grundlegende Datenstrukturen
 - ▼ 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können verändert werden
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutablen Objekte
 - 3.3 Ausgabe mit `print`
- ▼ 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`inf`)
 - 3.4.5 Not-a-Number (`nan`)
- ▼ 3.5 Mathematische Operationen
 - 3.5.1 Arithmetische Operationen
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
- 3.6 Umwandlung des Datentyps
- ▼ 3.7 Rundung
 - 3.7.1 Runden mit `round`

```
In [24]: import math  
  
a = 16  
print(math.sqrt(a))
```

4.0

3.5.4 Ganzzahlige Division

Inhaltsverzeichnis

- ⌚⚙️
- ▼ 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Ver-
- ▼ 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
- ▼ 3 Grundlegende Datenstrukturen
 - ▼ 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können verändert werden
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutab
 - 3.3 Ausgabe mit `print`
- ▼ 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`$\inf`)
 - 3.4.5 Not-a-Number (`NaN`)
- ▼ 3.5 Mathematische Operationen
 - 3.5.1 Arithmetische Operat
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
 - 3.6 Umwandlung des Datentyps
- ▼ 3.7 Rundung
 - 3.7.1 Runden mit `round`

In [25]:

```
a = 5  
b = 2  
print(a // b)
```

2

3.5.5 Divisionsrest (modulo)

Inhaltsverzeichnis

- ☞ *
 - ▼ 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Ver-
 - ▼ 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
 - ▼ 3 Grundlegende Datenstrukturen
 - ▼ 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können mehrfach zuweisen
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutab
 - 3.3 Ausgabe mit `print`
 - ▼ 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`inf`)
 - 3.4.5 Not-a-Number (`nan`)
 - ▼ 3.5 Mathematische Operationen
 - 3.5.1 Arithmetische Operat
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
 - 3.6 Umwandlung des Datentyps
 - ▼ 3.7 Rundung
 - 3.7.1 Runden mit `round`

- **Tip 1:** Nützlich, um zu prüfen, ob eine Zahl gerade ist.
- **Tip 2:** Auch nützlich, wenn man den Wertebereich einer Zahl begrenzen will.

```
In [26]: # Modulo / Divisionsrest
          print(a % b)
```

1

3.6 Umwandlung des Datentyps numerischer Werte

Inhaltsverzeichnis

- ⌚⚙️
- ▼ 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Ver-
- ▼ 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
- ▼ 3 Grundlegende Datenstrukturen
 - ▼ 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können mehrfach zuweisen
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutab
 - 3.3 Ausgabe mit `print`
- ▼ 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`$\inf`)
 - 3.4.5 Not-a-Number (`NaN`)
- ▼ 3.5 Mathematische Operationen
 - 3.5.1 Arithmetische Operat
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
- 3.6 Umwandlung des Datentyps
- ▼ 3.7 Rundung
 - 3.7.1 Runden mit `round`

```
In [27]: # float als int
# Was passiert?
print(int(3.1))
print(int(3.5))
print(int(3.8))
```

3

3

3

3.6 Umwandlung des Datentyps numerischer Werte

Inhaltsverzeichnis

- ⌚⚙️
- ▼ 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Ver-
- ▼ 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
- ▼ 3 Grundlegende Datenstrukturen
 - ▼ 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können mehrfach zuweisen
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutab
 - 3.3 Ausgabe mit `print`
- ▼ 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`$\inf`)
 - 3.4.5 Not-a-Number (`NaN`)
- ▼ 3.5 Mathematische Operationen
 - 3.5.1 Arithmetische Operat
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
- 3.6 Umwandlung des Datentyps
- ▼ 3.7 Rundung
 - 3.7.1 Runden mit `round`

```
In [27]: # float als int
          # Was passiert?
          print(int(3.1))
          print(int(3.5))
          print(int(3.8))
```

3
3
3

```
In [28]: # int als float
          print(float(7))
```

7.0

Inhaltsverzeichnis

- ⌚⚙️
- ▼ 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Verweisungssyntax
- ▼ 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
- ▼ 3 Grundlegende Datenstrukturen
 - ▼ 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können verändert werden
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutablen Objekte
 - 3.3 Ausgabe mit `print`
- ▼ 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`$\inf`)
 - 3.4.5 Not-a-Number (`NaN`)
- ▼ 3.5 Mathematische Operationen
 - 3.5.1 Arithmetische Operationen
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
- 3.6 Umwandlung des Datentyps
- ▼ 3.7 Rundung
 - 3.7.1 Runden mit `round`

In [14]:

```
# int als Binärzahl
print(bin(255))

zahl_als_binaerzahl = bin(255)
print(zahl_als_binaerzahl[2:])
print(type(zahl_als_binaerzahl))
```

```
0b11111111
11111111
11111111
<class 'str'>
```

In [14]:

```
# int als Binärzahl
print(bin(255))

zahl_als_binaerzahl = bin(255)
print(zahl_als_binaerzahl[2:])
print(type(zahl_als_binaerzahl))
```

```
0b11111111
11111111
11111111
<class 'str'>
```

In [30]:

```
# int als Hexadezimalzahl
print(hex(255))
```

```
0xff
```

Inhaltsverzeichnis

- ⚙️
- ▼ 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Verweisungssyntax
- ▼ 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
- ▼ 3 Grundlegende Datenstrukturen
 - ▼ 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können mehrfach zuweisen
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutablen Objekte
 - 3.3 Ausgabe mit `print`
 - ▼ 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`inf`)
 - 3.4.5 Not-a-Number (`nan`)
 - ▼ 3.5 Mathematische Operationen
 - 3.5.1 Arithmetische Operationen
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
 - 3.6 Umwandlung des Datentyps
 - ▼ 3.7 Rundung
 - 3.7.1 Runden mit `round`

3.7 Rundung

Inhaltsverzeichnis

- ☞ * **1** Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Ver-
- ▼ **2** Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
- ▼ **3** Grundlegende Datenstrukturen
 - ▼ **3.1** Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können verändert werden
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutab - 3.3 Ausgabe mit `print`
- ▼ **3.4** Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`$inf`)
 - 3.4.5 Not-a-Number (`NaN`)
- ▼ **3.5** Mathematische Operationen
 - 3.5.1 Arithmetische Operationen
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
- 3.6 Umwandlung des Datentyps
- ▼ **3.7** Rundung
 - 3.7.1 Runden mit `round()`

Bei der Umwandlung einer Gleitkommazahl in eine Ganzzahl mit `int()` ist die Art der Rundung nicht eindeutig.

3.7.1 Runden mit `round()`

Mit der Funktion `round(<wert>, <anzahl_stellen>)` kann man mathematisch korrekt runden.

Wenn keine Stellenanzahl angegeben wird, wird auf die nächste ganze Zahl gerundet.

Beispiel:

Inhaltsverzeichnis

- ⌚⚙️
- ▼ 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Verweisungssyntax
- ▼ 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
- ▼ 3 Grundlegende Datenstrukturen
 - ▼ 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können verändert werden
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutablen Objekte
 - 3.3 Ausgabe mit `print`
- ▼ 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`$\inf`)
 - 3.4.5 Not-a-Number (`NaN`)
- ▼ 3.5 Mathematische Operationen
 - 3.5.1 Arithmetische Operationen
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
- 3.6 Umwandlung des Datentyps
- ▼ 3.7 Rundung
 - 3.7.1 Runden mit `round()`

In [31]:

```
# Runden  
# round(value[, n])  
print(round(3.5))
```

4

Beispiel:

Inhaltsverzeichnis

- ⌚⚙️
- ▼ 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Verweisungssyntax
- ▼ 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
- ▼ 3 Grundlegende Datenstrukturen
 - ▼ 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können verschachtelt werden
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutablen Objekte
 - 3.3 Ausgabe mit `print`
- ▼ 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`$\inf`)
 - 3.4.5 Not-a-Number (`NaN`)
- ▼ 3.5 Mathematische Operationen
 - 3.5.1 Arithmetische Operationen
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
- 3.6 Umwandlung des Datentyps
- ▼ 3.7 Rundung
 - 3.7.1 Runden mit `round()`

In [31]:

```
# Runden  
# round(value[, n])  
print(round(3.5))
```

4

Der optionale zweite Parameter gibt an, wieviele Nachkommastellen gewünscht werden:

Beispiel:

Inhaltsverzeichnis

- ⌚⚙️
- ▼ 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Ver-
- ▼ 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
- ▼ 3 Grundlegende Datenstrukturen
 - 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können mehrfach zuweisen
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutab
 - 3.3 Ausgabe mit `print`
- ▼ 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`inf`)
 - 3.4.5 Not-a-Number (`nan`)
- ▼ 3.5 Mathematische Operationen
 - 3.5.1 Arithmetische Operat
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
- 3.6 Umwandlung des Datentyps
- ▼ 3.7 Rundung
 - 3.7.1 Runden mit `round`

```
In [31]: # Runden  
# round(value[, n])  
print(round(3.5))
```

4

Der optionale zweite Parameter gibt an, wieviele Nachkommastellen gewünscht werden:

```
In [32]: # Wir runden Pi auf drei Stellen nach dem Komma  
print(round(3.1415926, 3))
```

3.142

3.7.2 Abrunden mit `math.floor()`

Inhaltsverzeichnis

- ☞ 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Ver-
- ☞ 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
- ☞ 3 Grundlegende Datenstrukturen
 - 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können mehrfach zuweisen
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutab
 - 3.3 Ausgabe mit `print`
- ☞ 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`$inf`)
 - 3.4.5 Not-a-Number (`NaN`)
- ☞ 3.5 Mathematische Operationen
 - 3.5.1 Arithmetische Operat
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
- 3.6 Umwandlung des Datentyps
- ☞ 3.7 Rundung
 - 3.7.1 Runden mit `round()`

Mit der Funktion `math.floor(<wert>)` kann auf die nächstkleinere ganze Zahl abgerundet werden.

```
In [64]: import math

zahl = 3.8
print(math.floor(zahl))
negative_zahl = -3.8
print(math.floor(negative_zahl))
```

3

-4

3.7.3 Aufrunden mit `math.ceil()`

Inhaltsverzeichnis

- ☞ 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Ver-
- ▼ 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
- ▼ 3 Grundlegende Datenstrukturen
 - ▼ 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können mehrfach zuweisen
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutab
 - 3.3 Ausgabe mit `print`
- ▼ 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`$inf`)
 - 3.4.5 Not-a-Number (`NaN`)
- ▼ 3.5 Mathematische Operationen
 - 3.5.1 Arithmetische Operat
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
- 3.6 Umwandlung des Datentyps
- ▼ 3.7 Rundung
 - 3.7.1 Runden mit `round`

Mit der Funktion `math.ceil(<wert>)` kann auf die nächstgrößte ganze Zahl aufgerundet werden.

```
In [65]: import math

zahl = 3.8
print(math.ceil(zahl))
negative_zahl = -3.8
print(math.ceil(negative_zahl))
```

4

-3

3.8 Wahrheitswerte (Boolesche Werte)

Inhaltsverzeichnis

- ⌚ *
 - ▼ 1 Syntaktische Konvention
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Ver-
 - ▼ 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
 - ▼ 3 Grundlegende Datenstrukturen
 - 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können verändert werden
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutab - 3.3 Ausgabe mit `print`
 - ▼ 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`$\inf`)
 - 3.4.5 Not-a-Number (`NaN`)
 - ▼ 3.5 Mathematische Operatoren
 - 3.5.1 Arithmetische Operatoren
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
 - 3.6 Umwandlung des Datentyps
 - ▼ 3.7 Rundung
 - 3.7.1 Runden mit `round()`

Ähnlich wie wir in der elementaren Algebra mit Zahlen arbeiten, kann man in der sogenannten Booleschen Algebra mit den Wahrheitswerten `wahr` (`true`) und `unwahr` (`false`) arbeiten. Operatoren stehen uns **AND** (Konjunktion), **OR** (Disjunktion), **XOR** (Kontravalenz) und **NOT** (Negation) zur Verfügung.

Inhaltsverzeichnis

- ☞ * ▼ 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Verzweigungsoperatoren
- ▼ 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
- ▼ 3 Grundlegende Datenstrukturen
 - ▼ 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können verändert werden
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutablen Objekte
 - 3.3 Ausgabe mit `print`
- ▼ 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`inf`)
 - 3.4.5 Not-a-Number (`nan`)
- ▼ 3.5 Mathematische Operatoren
 - 3.5.1 Arithmetische Operatoren
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
 - 3.6 Umwandlung des Datentyps
- ▼ 3.7 Rundung
 - 3.7.1 Runden mit `round()`

Zwei (oder mehr) Boolesche Werte kann man mit den Operatoren AND, OR oder XOR verknüpfen. Mit NOT kann man einen Booleschen Wert invertieren:

a	b	a AND b	a OR b	NOT a	a XOR b
False	False	False	False	True	False
True	False	False	True	False	True
False	True	False	True	True	True
True	True	True	True	False	False

Praktisch relevant ist dies z.B. bei Suchmaschinen

```
"finde alle Bücher, die entweder 'Informatik' oder 'BWL' im Titel enthalten"
```

und bei Bedingungen in Geschäftsprozessen

```
"Kreditkarte_gültig AND Produkt_lieferbar".
```

3.8.1 Boolesche Werte und Operatoren in Python

Inhaltsverzeichnis

- ⌚⚙️
- ▼ 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Ver-
- ▼ 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
- ▼ 3 Grundlegende Datenstrukturen
 - ▼ 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können verändert werden
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutab
 - 3.3 Ausgabe mit `print`
- ▼ 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`inf`)
 - 3.4.5 Not-a-Number (`nan`)
- ▼ 3.5 Mathematische Operationen
 - 3.5.1 Arithmetische Operationen
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
- 3.6 Umwandlung des Datentyps
- ▼ 3.7 Rundung
 - 3.7.1 Runden mit `round`

```
In [15]: # Wahr und Falsch sind vordefiniert  
# Achtung: Schreibweise!  
a = True  
b = False
```

3.8.1 Boolesche Werte und Operatoren in Python

Inhaltsverzeichnis



- ▼ 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Verweisungssyntax
- ▼ 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
- ▼ 3 Grundlegende Datenstrukturen
 - ▼ 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können verändert werden
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutablen Objekte
 - 3.3 Ausgabe mit `print`
- ▼ 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`$inf`)
 - 3.4.5 Not-a-Number (`NaN`)
- ▼ 3.5 Mathematische Operationen
 - 3.5.1 Arithmetische Operationen
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
- 3.6 Umwandlung des Datentyps
- ▼ 3.7 Rundung
 - 3.7.1 Runden mit `round`

In [15]: `# Wahr und Falsch sind vordefiniert
Achtung: Schreibweise!
a = True
b = False`

In [16]: `# Logische Operatoren
print(a and b)
print(a or b)
print(not a)
Work-around für XOR
print(bool(a) ^ bool(b))`

False
True
False
True

3.8.2 Boolesche Werte lassen sich in Ganzzahlen umwandeln

Inhaltsverzeichnis

- ⌚⚙️
- ▼ 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Ver-
- ▼ 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
- ▼ 3 Grundlegende Datenstrukturen
 - ▼ 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können verändert werden
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutab
 - 3.3 Ausgabe mit `print`
- ▼ 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`$\inf`)
 - 3.4.5 Not-a-Number (`NaN`)
- ▼ 3.5 Mathematische Operationen
 - 3.5.1 Arithmetische Operat
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
- 3.6 Umwandlung des Datentyps
- ▼ 3.7 Rundung
 - 3.7.1 Runden mit `round`

```
In [35]: print(int(True))  
print(int(False))
```

1

0

3.8.2 Boolesche Werte lassen sich in Ganzzahlen umwandeln

Inhaltsverzeichnis

- ⌚ 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Verweisungsschlüssel
- ⌚ 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
- ⌚ 3 Grundlegende Datenstrukturen
 - ⌚ 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können mehrfach zuweisen
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutablen Objekte
 - ⌚ 3.3 Ausgabe mit `print`
 - ⌚ 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`inf`)
 - 3.4.5 Not-a-Number (`nan`)
 - ⌚ 3.5 Mathematische Operationen
 - 3.5.1 Arithmetische Operationen
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
 - ⌚ 3.6 Umwandlung des Datentyps
 - ⌚ 3.7 Rundung
 - 3.7.1 Runden mit `round`

```
In [35]: print(int(True))  
print(int(False))
```

1

0

```
In [36]: # Ziemlich nützlich bei Berechnungen  
versandkostenpflichtig = True  
versandkosten = 5.95  
nettobetrag = 135.00  
bruttobetrag = nettobetrag + versandkosten * versandkostenpflichtig  
print(bruttobetrag)
```

140.95

3.9 Vergleichsoperatoren

Inhaltsverzeichnis

- ⌚ 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Verweisoperatoren
- ⌚ 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
- ⌚ 3 Grundlegende Datenstrukturen
 - ⌚ 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können verändert werden
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutablen Objekte
 - 3.3 Ausgabe mit `print`
- ⌚ 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`$inf`)
 - 3.4.5 Not-a-Number (`$nan`)
- ⌚ 3.5 Mathematische Operatoren
 - 3.5.1 Arithmetische Operatoren
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
- ⌚ 3.6 Umwandlung des Datentyps
- ⌚ 3.7 Rundung
 - 3.7.1 Runden mit `round()`

In einem Programm muss man oft den Wert von Objekten vergleichen, z.B. den Lagerbestand mit einer Mindestmenge. Dazu gibt es sogenannte **Vergleichsoperatoren**. Das Ergebnis ist immer ein Boole'scher Wert, also `True` oder `False`.

In [37]:

```
a = 90  
b = 60  
c = 60
```

Inhaltsverzeichnis



- ▼ 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Verweisungssyntax
- ▼ 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
- ▼ 3 Grundlegende Datenstrukturen
 - ▼ 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können verändert werden
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutablen Objekte
 - 3.3 Ausgabe mit `print`
 - ▼ 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`$\inf`)
 - 3.4.5 Not-a-Number (`NaN`)
 - ▼ 3.5 Mathematische Operationen
 - 3.5.1 Arithmetische Operationen
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
 - 3.6 Umwandlung des Datentyps
 - ▼ 3.7 Rundung
 - 3.7.1 Runden mit `round()`

In [37]:

```
a = 90  
b = 60  
c = 60
```

In [38]:

```
print(a < b)
```

False

Inhaltsverzeichnis



- ▼ 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Ver-
- ▼ 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
- ▼ 3 Grundlegende Datenstrukturen
 - ▼ 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können verändert werden
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutab
 - 3.3 Ausgabe mit `print`
- ▼ 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`$\inf`)
 - 3.4.5 Not-a-Number (`NaN`)
- ▼ 3.5 Mathematische Operationen
 - 3.5.1 Arithmetische Operat
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
- 3.6 Umwandlung des Datentyps
- ▼ 3.7 Rundung
 - 3.7.1 Runden mit `round`

In [39]: `print(a > b)`

True

Inhaltsverzeichnis



- ▼ 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Verweisungssyntax
- ▼ 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
- ▼ 3 Grundlegende Datenstrukturen
 - ▼ 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können verändert werden
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutablen Objekte
 - 3.3 Ausgabe mit `print`
 - ▼ 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`$\inf`)
 - 3.4.5 Not-a-Number (`NAN`)
 - ▼ 3.5 Mathematische Operationen
 - 3.5.1 Arithmetische Operationen
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
 - 3.6 Umwandlung des Datentyps
 - ▼ 3.7 Rundung
 - 3.7.1 Runden mit `round`

In [39]: `print(a > b)`

True

In [40]: `print(a < a)`

False

Inhaltsverzeichnis

- ⌚⚙️
- ▼ 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Ver-
- ▼ 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
- ▼ 3 Grundlegende Datenstrukturen
 - ▼ 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können verändert werden
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutab
 - 3.3 Ausgabe mit `print`
- ▼ 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`$\inf`)
 - 3.4.5 Not-a-Number (`NaN`)
- ▼ 3.5 Mathematische Operationen
 - 3.5.1 Arithmetische Operat
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
- 3.6 Umwandlung des Datentyps
- ▼ 3.7 Rundung
 - 3.7.1 Runden mit `round`

```
In [41]: print(a <= a)
```

```
True
```

Inhaltsverzeichnis



- ▼ 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Verweisungssyntax
- ▼ 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
- ▼ 3 Grundlegende Datenstrukturen
 - ▼ 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können verändert werden
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutablen Objekte
 - 3.3 Ausgabe mit `print`
 - ▼ 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`$\inf`)
 - 3.4.5 Not-a-Number (`NAN`)
 - ▼ 3.5 Mathematische Operationen
 - 3.5.1 Arithmetische Operationen
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
 - 3.6 Umwandlung des Datentyps
 - ▼ 3.7 Rundung
 - 3.7.1 Runden mit `round`

```
In [41]: print(a <= a)
```

True

```
In [42]: print(a >= a)
```

True

Inhaltsverzeichnis



- ▼ 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Ver-
- ▼ 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
- ▼ 3 Grundlegende Datenstrukturen
 - ▼ 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können verändert werden
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutab
 - 3.3 Ausgabe mit `print`
- ▼ 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`$\inf`)
 - 3.4.5 Not-a-Number (`NaN`)
- ▼ 3.5 Mathematische Operationen
 - 3.5.1 Arithmetische Operat
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
- 3.6 Umwandlung des Datentyps
- ▼ 3.7 Rundung
 - 3.7.1 Runden mit `round`

3.10 Wertevergleich oder Identitätsvergleich?

Inhaltsverzeichnis

☞ *	
▼ 1	Syntaktische Konventionen
1.1	Keine Zeilenummern
1.2	Groß-/Kleinschreibung
1.3	Reservierte Namen
1.4	Zuweisungs- und Verweisungsschlüssel
▼ 2	Stil und Formatierung
2.1	Namen
2.2	Leerzeichen
▼ 3	Grundlegende Datenstrukturen
3.1	Namen und Objekte
3.1.1	Alles in Python ist ein Objekt
3.1.2	Objekte können mehrfach zugegriffen werden
3.1.3	Mehrfachzuweisung
3.2	Mutable und Immutablen Objekte
3.3	Ausgabe mit <code>print</code>
3.4	Numerische Werte
3.4.1	Ganze Zahlen
3.4.2	Gleitkommazahl
3.4.3	Dezimalzahlen
3.4.4	Unendlich (<code>\$\inf</code>)
3.4.5	Not-a-Number (<code>NaN</code>)
3.5	Mathematische Operationen
3.5.1	Arithmetische Operationen
3.5.2	Potenz
3.5.3	Wurzel
3.5.4	Ganzzahlige Division
3.5.5	Divisionsrest (modulo)
3.6	Umwandlung des Datentyps
3.7	Rundung
3.7.1	Runden mit <code>round()</code>

Wenn man Ausdrücke oder Objekte vergleicht, muss man sich überlegen, ob man

1. den **Wert** der Ausdrücke vergleichen will, oder
2. ob geprüft werden soll, ob es sich um **dasselbe Objekt** handelt

Wertevergleich mit `a == b`

Identitätsvergleich mit `a is b`

Bei numerischen Ausdrücken gibt es i.d.R. keinen Unterschied

Inhaltsverzeichnis

- ⟳ ↘ 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Ver-
- ⟳ ↘ 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
- ⟳ ↘ 3 Grundlegende Datenstrukturen
 - 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können mehrfach zuweisen
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutab
 - 3.3 Ausgabe mit `print`
- ⟳ ↘ 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`$\inf`)
 - 3.4.5 Not-a-Number (`NaN`)
- ⟳ ↘ 3.5 Mathematische Operationen
 - 3.5.1 Arithmetische Operat
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
- 3.6 Umwandlung des Datentyps
- 3.7 Rundung
 - 3.7.1 Runden mit `round`

```
In [43]: print(3 * 5 == 15)
          print(3 * 5 is 15)
```

True

True

Bei numerischen Ausdrücken gibt es i.d.R. keinen Unterschied

Inhaltsverzeichnis

- ⌚ ⓘ **1** Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Verweisungssyntax
- ⌚ ⓘ **2** Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
- ⌚ ⓘ **3** Grundlegende Datenstrukturen
 - 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können mehrfach zuweisen
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutablen Objekte
 - 3.3 Ausgabe mit `print`
- ⌚ ⓘ **3.4** Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`$\inf`)
 - 3.4.5 Not-a-Number (`NaN`)
- ⌚ ⓘ **3.5** Mathematische Operationen
 - 3.5.1 Arithmetische Operationen
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
- ⌚ ⓘ **3.6** Umwandlung des Datentyps
- ⌚ ⓘ **3.7** Rundung
 - 3.7.1 Runden mit `round`

```
In [43]: print(3 * 5 == 15)
          print(3 * 5 is 15)
```

True

True

Allerdings sollte man sich nicht darauf verlassen, dass dieselbe Wahrheit auch durch dasselbe Objekt repräsentiert wird und daher stets Wahrheit vergleichen, wenn man numerische Größen vergleicht, und nicht die Identität der Objekte.

Inhaltsverzeichnis

- ⌚⚙️
- ▼ 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Verweisungssyntax
- ▼ 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
- ▼ 3 Grundlegende Datenstrukturen
 - ▼ 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können mehrfach zuweisen
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutablen Objekte
 - 3.3 Ausgabe mit `print`
- ▼ 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`$\inf`)
 - 3.4.5 Not-a-Number (`NaN`)
- ▼ 3.5 Mathematische Operationen
 - 3.5.1 Arithmetische Operationen
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
- 3.6 Umwandlung des Datentyps
- ▼ 3.7 Rundung
 - 3.7.1 Runden mit `round`

In [44]:

```
# Dito bei Strings
a = "Text"
b = "Text"
print(a == b)
print(a is b)
# Warum?
```

True

True

Bei änderbaren Objekten (Mutables) sieht es aber anders aus

Inhaltsverzeichnis

- ⌚ ⓘ **1** Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Verweisungssyntax
- ⌚ ⓘ **2** Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
- ⌚ ⓘ **3** Grundlegende Datenstrukturen
 - ⌚ ⓘ **3.1** Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können verändert werden
 - 3.1.3 Mehrfachzuweisung
 - ⌚ ⓘ **3.2** Mutable und Immutablen Objekte
 - ⌚ ⓘ **3.3** Ausgabe mit `print`
 - ⌚ ⓘ **3.4** Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`$\inf`)
 - 3.4.5 Not-a-Number (`NaN`)
 - ⌚ ⓘ **3.5** Mathematische Operationen
 - 3.5.1 Arithmetische Operationen
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
 - ⌚ ⓘ **3.6** Umwandlung des Datentyps
 - ⌚ ⓘ **3.7** Rundung
 - 3.7.1 Runden mit `round`

In [20]:

```
a = [1, 2]
b = [1, 2]
c = b
print(a == b)
print(a is b)
print(c == b)
print(c is b)
```

True

False

True

True

Bei änderbaren Objekten (Mutables) sieht es aber anders aus

Inhaltsverzeichnis

- ⌚ ⓘ
 - ▼ 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Verweisungssyntax
 - ▼ 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
 - ▼ 3 Grundlegende Datenstrukturen
 - ▼ 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können mehrfach zuweisen
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutablen Objekte
 - 3.3 Ausgabe mit `print`
 - ▼ 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`$\inf`)
 - 3.4.5 Not-a-Number (`NaN`)
 - ▼ 3.5 Mathematische Operationen
 - 3.5.1 Arithmetische Operationen
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
 - 3.6 Umwandlung des Datentyps
 - ▼ 3.7 Rundung
 - 3.7.1 Runden mit `round`

In [20]:

```
a = [1, 2]
b = [1, 2]
c = b
print(a == b)
print(a is b)
print(c == b)
print(c is b)
```

```
True
False
True
True
```

Das liegt daran, dass änderbare Objekte im Speicher eigene Plätze einnehmen, weil der Computer ja nicht wissen kann, ob sie immer identisch bleiben.

Beim Wertevergleich mit == wird automatisch eine Typumwandlung versucht:

```
In [46]: print(5 == 5.0)
```

```
True
```

Inhaltsverzeichnis

- ⚙️
- ▼ 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Verweisungsschlüssel
- ▼ 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
- ▼ 3 Grundlegende Datenstrukturen
 - ▼ 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können verändert werden
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutablen Objekte
 - 3.3 Ausgabe mit `print`
- ▼ 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`$\inf`)
 - 3.4.5 Not-a-Number (`$\NaN`)
- ▼ 3.5 Mathematische Operationen
 - 3.5.1 Arithmetische Operationen
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
- 3.6 Umwandlung des Datentyps
- ▼ 3.7 Rundung
 - 3.7.1 Runden mit `round`

Beim Identitätsvergleich sind verschiedene Datentypen verschiedene Objekte, selbst wenn sich ihre Werte umwande ließen:

Inhaltsverzeichnis

- ⌚ ⓘ
 - ▼ 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Verweisungsschlüssel
 - ▼ 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
 - ▼ 3 Grundlegende Datenstrukturen
 - ▼ 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können mehrfach zuweisen
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutablen Objekte
 - 3.3 Ausgabe mit `print`
 - ▼ 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`$inf`)
 - 3.4.5 Not-a-Number (`NaN`)
 - ▼ 3.5 Mathematische Operationen
 - 3.5.1 Arithmetische Operationen
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
 - 3.6 Umwandlung des Datentyps
 - ▼ 3.7 Rundung
 - 3.7.1 Runden mit `round()`

Beim Identitätsvergleich sind verschiedene Datentypen verschiedene Objekte, selbst wenn sich ihre Werte umwande ließen:

```
In [47]: print(5 is 5.0)
print(True is 1)
```

```
False
```

```
False
```

Inhaltsverzeichnis

- ⌚ ↗
 - ▼ 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Verweisungssyntax
 - ▼ 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
 - ▼ 3 Grundlegende Datenstrukturen
 - ▼ 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können gleich sein
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutablen Objekte
 - 3.3 Ausgabe mit `print`
 - ▼ 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`$\inf`)
 - 3.4.5 Not-a-Number (`NaN`)
 - ▼ 3.5 Mathematische Operationen
 - 3.5.1 Arithmetische Operationen
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
 - 3.6 Umwandlung des Datentyps
 - ▼ 3.7 Rundung
 - 3.7.1 Runden mit `round`

Beim Identitätsvergleich sind verschiedene Datentypen verschiedene Objekte, selbst wenn sich ihre Werte umwande ließen:

```
In [47]: print(5 is 5.0)
print(True is 1)
```

False

False

```
In [48]: # Aber:
print(5 is int(5.0))
print(int(True) is 1)
print(True is bool(1))
```

True

True

True

3.11 Trigonometrische und sonstige mathematische Funktionen

Siehe auch <https://docs.python.org/3/library/math.html>.

```
In [49]: import math

# Pi
print(math.pi)
# Eulersche Zahl
print(math.e)
```

3.141592653589793

2.718281828459045

Inhaltsverzeichnis



- ▼ 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Verweisungsschlüsselelemente
- ▼ 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
- ▼ 3 Grundlegende Datenstrukturen
 - ▼ 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können verändert werden
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutablen Objekte
 - 3.3 Ausgabe mit `print`
- ▼ 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`$\inf`)
 - 3.4.5 Not-a-Number (`NaN`)
- ▼ 3.5 Mathematische Operationen
 - 3.5.1 Arithmetische Operationen
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
- 3.6 Umwandlung des Datentyps
- ▼ 3.7 Rundung
 - 3.7.1 Runden mit `round`

In [50]:

```
# Quadratwurzel
print(math.sqrt(16))
# Sinus
print(math.sin(90))
# Cosinus
print(math.cos(math.pi))
# Tangens
print(math.tan(math.pi))
# Log2
print(math.log2(256))
```

```
4.0
0.8939966636005579
-1.0
-1.2246467991473532e-16
8.0
```

3.12 Komplexe Datentypen

Inhaltsverzeichnis

- ☞ *
 - ▼ 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Ver-
 - ▼ 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
 - ▼ 3 Grundlegende Datenstrukturen
 - ▼ 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können benannt werden
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutablen Objekte
 - 3.3 Ausgabe mit `print`
 - ▼ 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`inf`)
 - 3.4.5 Not-a-Number (`NaN`)
 - ▼ 3.5 Mathematische Operationen
 - 3.5.1 Arithmetische Operationen
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
 - 3.6 Umwandlung des Datentyps
 - ▼ 3.7 Rundung
 - 3.7.1 Runden mit `round()`

Als komplexe Datentypen bezeichnet man solche, die eine adressierbare Struktur an Unterelementen haben.

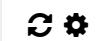
- Zeichenketten

0	1	2
W	O	W

- Listen
- Dictionaries
- Tuples
- Mengen / Sets
- sonstige, auch benutzerdefinierte Objekte

3.12.1 Zeichenketten

Inhaltsverzeichnis



- ▼ 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Verweisungsschlüsselwörter
- ▼ 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
- ▼ 3 Grundlegende Datenstrukturen
 - 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können mehrfach zuweisen
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutablen Objekte
 - 3.3 Ausgabe mit `print`
 - 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`$\inf`)
 - 3.4.5 Not-a-Number (`NaN`)
 - 3.5 Mathematische Operationen
 - 3.5.1 Arithmetische Operationen
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
 - 3.6 Umwandlung des Datentyps
 - 3.7 Rundung
 - 3.7.1 Runden mit `round()`

```
In [51]: # Zeichenkette
my_string_1 = 'UniBwM'
my_string_2 = "UniBwM"
```

```
In [52]: # Die Wahl zwischen einfachen und doppelten Anführungszeichen erlaubt es einem Programmierer, die jeweils andere Form innerhalb der Zeichenkette zu verwenden:
my_string_3 = "Die Abkürzung für unsere Universität lautet 'UniBwM'."  
my_string_3 = 'Die Abkürzung für unsere Universität lautet "UniBwM".'
```

Inhaltsverzeichnis



- ▼ 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Verweisungssyntax
- ▼ 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
- ▼ 3 Grundlegende Datenstrukturen
 - ▼ 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können verändert werden
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutablen Objekte
 - 3.3 Ausgabe mit `print`
- ▼ 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`$inf`)
 - 3.4.5 Not-a-Number (`NaN`)
- ▼ 3.5 Mathematische Operationen
 - 3.5.1 Arithmetische Operationen
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
- 3.6 Umwandlung des Datentyps
- ▼ 3.7 Rundung
 - 3.7.1 Runden mit `round()`

In [12]:

```
# Mehrzeilige Zeichenketten erfordern DREI Anführungszeichen:  
my_long_string_1 = """Herr von Ribbeck auf Ribbeck im Havelland,  
Ein Birnbaum in seinem Garten stand,  
Und kam die goldene Herbsteszeit,  
Und die Birnen leuchteten weit und breit,  
Da stopfte, wenn's Mittag vom Thurme scholl,  
Der von Ribbeck sich beide Taschen voll,  
Und kam in Pantinen ein Junge daher,  
So rief er: 'Junge, wist' ne Beer?'  
Und kam ein Mädel, so rief er: 'Lütt Dirn'  
Kumm man röwer, ick hebb' ne Birn."""
```

Inhaltsverzeichnis



- ▼ 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Verweisungssyntax
- ▼ 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
- ▼ 3 Grundlegende Datenstrukturen
 - ▼ 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können verändert werden
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutablen Objekte
 - 3.3 Ausgabe mit `print`
- ▼ 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`$inf`)
 - 3.4.5 Not-a-Number (`NaN`)
- ▼ 3.5 Mathematische Operationen
 - 3.5.1 Arithmetische Operationen
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
- 3.6 Umwandlung des Datentyps
- ▼ 3.7 Rundung
 - 3.7.1 Runden mit `round()`

In [54]:

```
my_long_string_2 = '''Herr von Ribbeck auf Ribbeck im Havelland,  
Ein Birnbaum in seinem Garten stand,  
Und kam die goldene Herbsteszeit,  
Und die Birnen leuchteten weit und breit,  
Da stopfte, wenn's Mittag vom Thurme scholl,  
Der von Ribbeck sich beide Taschen voll,  
Und kam in Pantinen ein Junge daher,  
So rief er: "Junge, wist' ne Beer?"  
Und kam ein Mädel, so rief er: "Lütt Dirn"  
Kumm man röwer, ick hebb' ne Birn.'''
```

3.12.1.1 Addition von Zeichenketten

Inhaltsverzeichnis

- ⌚⚙️
- ▼ 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Verweisungssyntax
- ▼ 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
- ▼ 3 Grundlegende Datenstrukturen
 - ▼ 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können mehrfach zuweisen
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutablen Objekte
 - 3.3 Ausgabe mit `print`
- ▼ 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`$\inf`)
 - 3.4.5 Not-a-Number (`NaN`)
- ▼ 3.5 Mathematische Operationen
 - 3.5.1 Arithmetische Operationen
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
- 3.6 Umwandlung des Datentyps
- ▼ 3.7 Rundung
 - 3.7.1 Runden mit `round`

In [21]:

```
my_string_1 = "UniBwM"
print('Ich studiere an der ' + my_string_1)

# Addition mit einem Nicht-String
print('Text 1' + str(5 * 7))
```

Ich studiere an der UniBwM

Text 135

Text 135

3.12.1.2 Multiplikation von Zeichenketten

Inhaltsverzeichnis

- ⟳ ⚙
 - ▼ 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Ver-
 - ▼ 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
 - ▼ 3 Grundlegende Datenstrukturen
 - ▼ 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können verändert werden
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutab - 3.3 Ausgabe mit `print`
 - ▼ 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`$\inf`)
 - 3.4.5 Not-a-Number (`NaN`)
 - ▼ 3.5 Mathematische Operationen
 - 3.5.1 Arithmetische Operatoren
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
 - 3.6 Umwandlung des Datentyps
 - ▼ 3.7 Rundung
 - 3.7.1 Runden mit `round()`

In [56]: `print('ABCD' * 3)`

ABCDABCDABCD

3.12.1.2 Multiplikation von Zeichenketten

Inhaltsverzeichnis



- ▼ 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Ver-
- ▼ 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
- ▼ 3 Grundlegende Datenstrukturen
 - ▼ 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können mehrfach zuweisen
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutab
 - 3.3 Ausgabe mit `print`
- ▼ 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`$\inf`)
 - 3.4.5 Not-a-Number (`NaN`)
- ▼ 3.5 Mathematische Operationen
 - 3.5.1 Arithmetische Operat
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
- 3.6 Umwandlung des Datentyps
- ▼ 3.7 Rundung
 - 3.7.1 Runden mit `round`

In [56]: `print('ABCD' * 3)`

ABCDABCDABCD

In [57]: *# Nützlich z.B. für*

```
print('=' * 60)
print('Programm ABCD Version 1.0')
print('=' * 60)
```

=====

Programm ABCD Version 1.0

=====

Aber man kann keine Zeichenketten *miteinander* multiplizieren

Inhaltsverzeichnis



- ▼ 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Verweisungssyntax
- ▼ 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
- ▼ 3 Grundlegende Datenstrukturen
 - ▼ 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können mehrfach zuweisen
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutablen Objekte
 - 3.3 Ausgabe mit `print`
- ▼ 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`$inf`)
 - 3.4.5 Not-a-Number (`NaN`)
- ▼ 3.5 Mathematische Operationen
 - 3.5.1 Arithmetische Operationen
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
- 3.6 Umwandlung des Datentyps
- ▼ 3.7 Rundung
 - 3.7.1 Runden mit `round()`

In [58]:

```
my_string_test = '11'  
second_string = '2'  
print(my_string_test * second_string)
```

```
-----  
TypeError
```

```
Traceback (most recent call last)
```

```
<ipython-input-58-1f4b1a3739c9> in <module>
```

```
    1 my_string_test = '11'  
    2 second_string = '2'  
----> 3 print(my_string_test * second_string)
```

```
TypeError: can't multiply sequence by non-int of type 'str'
```

3.12.1.3 Länge ermitteln

Inhaltsverzeichnis



- ▼ 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Verweisungsschlüsselwörter
- ▼ 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
- ▼ 3 Grundlegende Datenstrukturen
 - ▼ 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können verändert werden
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutablen Objekte
 - 3.3 Ausgabe mit `print`
 - ▼ 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`$\inf`)
 - 3.4.5 Not-a-Number (`NaN`)
 - ▼ 3.5 Mathematische Operationen
 - 3.5.1 Arithmetische Operationen
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
 - 3.6 Umwandlung des Datentyps
 - ▼ 3.7 Rundung
 - 3.7.1 Runden mit `round`

In [60]:

```
my_string = "LOTTO"  
print(len(my_string))
```

5

3.12.1.4 Sonderzeichen in Zeichenketten (Escaping)

Inhaltsverzeichnis

- » 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Ver-
- » 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
- » 3 Grundlegende Datenstrukturen
 - » 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können verändert werden
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutab
 - 3.3 Ausgabe mit `print`
- » 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`$\inf`)
 - 3.4.5 Not-a-Number (`NaN`)
- » 3.5 Mathematische Operationen
 - 3.5.1 Arithmetische Operat
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
- 3.6 Umwandlung des Datentyps
- » 3.7 Rundung
 - 3.7.1 Runden mit `round`

Für eine vollständige Liste siehe z.B. [List of Python Escape sequence characters with examples](#).

```
In [1]: # Zeilenumbruch
      print('text\nneue Zeile')
```

```
text
neueZeile
```

In [2]:

```
# Tabulator
print('wert 1\twert2\twert3')
```

wert 1 wert2 wert3

Inhaltsverzeichnis



- ▼ 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Verweisungsschlüssel
- ▼ 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
- ▼ 3 Grundlegende Datenstrukturen
 - ▼ 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können verändert werden
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutablen Objekte
 - 3.3 Ausgabe mit `print`
 - ▼ 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`$\inf`)
 - 3.4.5 Not-a-Number (`$\NaN`)
 - ▼ 3.5 Mathematische Operationen
 - 3.5.1 Arithmetische Operationen
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
 - 3.6 Umwandlung des Datentyps
 - ▼ 3.7 Rundung
 - 3.7.1 Runden mit `round`

In [2]:

```
# Tabulator
print('wert 1\twert2\twert3')
```

wert 1 wert2 wert3

In [3]:

```
# Backslash
print('a \\ b')
```

a \ b

Inhaltsverzeichnis



- ▼ 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Ver-
- ▼ 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
- ▼ 3 Grundlegende Datenstrukturen
 - ▼ 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können verändert werden
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutabile Objekte
 - 3.3 Ausgabe mit `print`
- ▼ 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`$\inf`)
 - 3.4.5 Not-a-Number (`$\NaN`)
- ▼ 3.5 Mathematische Operationen
 - 3.5.1 Arithmetische Operationen
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
- 3.6 Umwandlung des Datentyps
- ▼ 3.7 Rundung
 - 3.7.1 Runden mit `round`

Inhaltsverzeichnis

- ⌚⚙️
- ▼ 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Verweisungsschlüsselelemente
- ▼ 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
- ▼ 3 Grundlegende Datenstrukturen
 - ▼ 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können verändert werden
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutablen Objekte
 - 3.3 Ausgabe mit `print`
- ▼ 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`$\inf`)
 - 3.4.5 Not-a-Number (`NaN`)
- ▼ 3.5 Mathematische Operationen
 - 3.5.1 Arithmetische Operationen
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
- 3.6 Umwandlung des Datentyps
- ▼ 3.7 Rundung
 - 3.7.1 Runden mit `round`

In [5]:

```
# Anführungszeichen
print('Er sagte \'Hallo\'')
print("Er sagte \"Hallo\"")
```

Er sagte 'Hallo'
Er sagte "Hallo"

3.12.1.5 *f-Strings (nicht klausurrelevant)*

Inhaltsverzeichnis

- » 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Verweisungssyntax
- » 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
- » 3 Grundlegende Datenstrukturen
 - 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können verändert werden
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutablen Objekte
 - 3.3 Ausgabe mit `print`
- » 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`$\inf`)
 - 3.4.5 Not-a-Number (`NaN`)
- » 3.5 Mathematische Operationen
 - 3.5.1 Arithmetische Operationen
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
- 3.6 Umwandlung des Datentyps
- » 3.7 Rundung
 - 3.7.1 Runden mit `round`

Schon immer gab es in Python die Möglichkeit, Werte in eine Zeichenkette einzubetten und zu formatieren, damit man einen Ergebnisstring nicht aufwändig zusammenfügen muss.

Seit der Version 3.6 existiert ein deutlich besserer Mechanismus, der sich 'f-Strings' nennt.

Wenn man vor eine Zeichenkette den Buchstaben 'f' setzt, kann innerhalb geschweifter Klammern beliebige Python-Ausdrücke einfügen:

Beispiel für f-Strings: Die Ausdrücke innerhalb der geschweiften Klammern werden durch ihren Wert ersetzt.

Inhaltsverzeichnis

- ⚙️
- ▼ 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Verweisungssyntax
- ▼ 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
- ▼ 3 Grundlegende Datenstrukturen
 - ▼ 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können verändert werden
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutablen Objekte
 - 3.3 Ausgabe mit `print`
 - ▼ 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`$inf`)
 - 3.4.5 Not-a-Number (`$nan`)
 - ▼ 3.5 Mathematische Operationen
 - 3.5.1 Arithmetische Operationen
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
 - 3.6 Umwandlung des Datentyps
 - ▼ 3.7 Rundung
 - 3.7.1 Runden mit `round`

Martin Hepp, martin.hepp@unibw.de

In [66]:

```
import math

name = 'Franz'
print(f'Hello {name}!')
print(f'Der Umfang eines Kreises mit dem Radius r=2 ist {math.pi * 2}.')
```

Hello Franz!

Der Umfang eines Kreises mit dem Radius r=2 ist 6.283185307179586.

Inhaltsverzeichnis



- ▼ 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Verweisungssyntax
- ▼ 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
- ▼ 3 Grundlegende Datenstrukturen
 - ▼ 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können verändert werden
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutablen Objekte
 - 3.3 Ausgabe mit `print`
- ▼ 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`$\inf`)
 - 3.4.5 Not-a-Number (`NaN`)
- ▼ 3.5 Mathematische Operationen
 - 3.5.1 Arithmetische Operationen
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
- 3.6 Umwandlung des Datentyps
- ▼ 3.7 Rundung
 - 3.7.1 Runden mit `round`

Formattierung von Werten innerhalb von f-Strings (nicht klausurrelevant)

Inhaltsverzeichnis



- ▼ 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Verweisungssyntax
- ▼ 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
- ▼ 3 Grundlegende Datenstrukturen
 - ▼ 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können benannt werden
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutablen Objekte
 - 3.3 Ausgabe mit `print`
- ▼ 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`inf`)
 - 3.4.5 Not-a-Number (`nan`)
- ▼ 3.5 Mathematische Operationen
 - 3.5.1 Arithmetische Operationen
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
- 3.6 Umwandlung des Datentyps
- ▼ 3.7 Rundung
 - 3.7.1 Runden mit `round`

Man kann die Werte in der Ausgabe auch formattieren. Dazu setzt man hinter den Ausdruck einen Doppelpunkt und dann verschiedene Angaben, wie

- die gesamte Breite in Zeichen **inklusive des Dezimaltrenners** (Punkt oder Komma),
- die Anzahl Nachkommastellen,
- ob fehlende Stellen vor dem Dezimalpunkt mit Leerzeichen, Nullen oder einem anderen Zeichen aufgefüllt werden sollen, sowie

Formatierung von Werten innerhalb von f-Strings (nicht klausurrelevant)

Inhaltsverzeichnis



- ▼ 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Verweisungssyntax
- ▼ 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
- ▼ 3 Grundlegende Datenstrukturen
 - ▼ 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können verschachtelt werden
 - 3.1.3 Mehrfachzuweisungen
 - 3.2 Mutable und Immutablen Objekte
 - 3.3 Ausgabe mit `print`
- ▼ 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`$inf`)
 - 3.4.5 Not-a-Number (`NaN`)
- ▼ 3.5 Mathematische Operationen
 - 3.5.1 Arithmetische Operationen
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
- 3.6 Umwandlung des Datentyps
- ▼ 3.7 Rundung
 - 3.7.1 Runden mit `round()`

```
f' {<wert>:<breite>.<nachkommastellen>f}'
```

Mit führender Null:

```
f' {<wert>:0<breite>.<nachkommastellen>f}'
```

[Link zur vollständigen Dokumentation der Formatierungsanweisungen.](#)

Beispiel

Inhaltsverzeichnis

- ⌚⚙️
- ▼ 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Ver-
- ▼ 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
- ▼ 3 Grundlegende Datenstrukturen
 - ▼ 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können mehrfach zuweisen
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutab
 - 3.3 Ausgabe mit `print`
- ▼ 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`$\inf`)
 - 3.4.5 Not-a-Number (`NaN`)
- ▼ 3.5 Mathematische Operationen
 - 3.5.1 Arithmetische Operat
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
- 3.6 Umwandlung des Datentyps
- ▼ 3.7 Rundung
 - 3.7.1 Runden mit `round`

```
In [69]: print(f'Pi ohne Nachkommastellen: {math.pi:.0f} ')
print(f'Pi mit zwei Nachkommastellen: {math.pi:.2f} ')
print(f'Pi mit vier Nachkommastellen: {math.pi:.4f} ')
```

```
Pi ohne Nachkommastellen: 3
Pi mit zwei Nachkommastellen: 3.14
Pi mit vier Nachkommastellen: 3.1416
```

Beispiel

Inhaltsverzeichnis



- ▼ 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Verweisungssyntax
- ▼ 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
- ▼ 3 Grundlegende Datenstrukturen
 - ▼ 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können verändert werden
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutablen Objekte
 - 3.3 Ausgabe mit `print`
- ▼ 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`$\inf`)
 - 3.4.5 Not-a-Number (`NaN`)
- ▼ 3.5 Mathematische Operationen
 - 3.5.1 Arithmetische Operationen
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
- 3.6 Umwandlung des Datentyps
- ▼ 3.7 Rundung
 - 3.7.1 Runden mit `round`

In [94]:

```
a = 3.5678
b = 345.7
# Fünf Stellen Gesamtlänge, eine Nachkommastelle
# Fehlende Stellen vor dem Wert werden mit Leerzeichen aufgefüllt.
print(f'Wert 1:{a:5.1f} Wert 2:{b:5.1f}')
# Dito, aber Auffüllung mit Nullen
print(f'Wert 1:{a:05.1f} Wert 2:{b:05.1f}'')
```

Wert 1: 3.6 Wert 2:345.7

Wert 1:003.6 Wert 2:345.7

3.12.1.6 Weitere Hilfsfunktionen für Strings

Inhaltsverzeichnis

- ⌚⚙️
- ▼ 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Verweisungssyntax
- ▼ 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
- ▼ 3 Grundlegende Datenstrukturen
 - ▼ 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können verändert werden
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutablen Objekte
 - 3.3 Ausgabe mit `print`
- ▼ 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`inf`)
 - 3.4.5 Not-a-Number (`NaN`)
- ▼ 3.5 Mathematische Operationen
 - 3.5.1 Arithmetische Operationen
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
- 3.6 Umwandlung des Datentyps
- ▼ 3.7 Rundung
 - 3.7.1 Runden mit `round`

```
In [122]: # https://docs.python.org/3/library/stdtypes.html
text = "UniBwM ist toll"
print(text.lower())
print(text.upper())
print(text.split(" "))
```

```
unibwm ist toll
UNIBWM IST TOLL
['UniBwM', 'ist', 'toll']
```

Aufsplitten mit `.split()`

Inhaltsverzeichnis



- ▼ 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Verweisungssyntax
- ▼ 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
- ▼ 3 Grundlegende Datenstrukturen
 - ▼ 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können verändert werden
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutablen Objekte
 - 3.3 Ausgabe mit `print`
 - ▼ 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`$\inf`)
 - 3.4.5 Not-a-Number (`NaN`)
 - ▼ 3.5 Mathematische Operationen
 - 3.5.1 Arithmetische Operationen
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
 - 3.6 Umwandlung des Datentyps
 - ▼ 3.7 Rundung
 - 3.7.1 Runden mit `round`

```
In [123]: text_2 = "Der erste Satz. Und nun der zweite Satz."  
          print(text_2.split("."))
```

```
'Der erste Satz', ' Und nun der zweite Satz', ''
```

Aufsplitten mit `.split()`

Inhaltsverzeichnis

- ⌚ 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Verweisungssyntax
- ⌚ 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
- ⌚ 3 Grundlegende Datenstrukturen
 - 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können verändert werden
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutablen Objekte
 - 3.3 Ausgabe mit `print`
- ⌚ 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`$\inf`)
 - 3.4.5 Not-a-Number (`NaN`)
- ⌚ 3.5 Mathematische Operationen
 - 3.5.1 Arithmetische Operationen
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
- ⌚ 3.6 Umwandlung des Datentyps
- ⌚ 3.7 Rundung
 - 3.7.1 Runden mit `round`

```
In [123]: text_2 = "Der erste Satz. Und nun der zweite Satz."  
          print(text_2.split("."))
```

```
['Der erste Satz', ' Und nun der zweite Satz', '']
```

Whitespace (Leerzeichen etc.) entfernen mit `.strip()`

```
In [124]: text_3 = " = Hallo = "  
          print(text_3.strip())
```

```
= Hallo =
```

endswith () und startswith () für Zeichenketten.

Inhaltsverzeichnis

- ☞ *
 - ▼ 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Ver-
 - ▼ 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
 - ▼ 3 Grundlegende Datenstrukturen
 - ▼ 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können mehrfach zuweisen
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutab
 - 3.3 Ausgabe mit `print`
 - ▼ 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`$\inf`)
 - 3.4.5 Not-a-Number (`NaN`)
 - ▼ 3.5 Mathematische Operationen
 - 3.5.1 Arithmetische Operat
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
 - 3.6 Umwandlung des Datentyps
 - ▼ 3.7 Rundung
 - 3.7.1 Runden mit `round`

```
In [111]: text = "Universität der Bundeswehr"
          print(text.startswith('Uni'))
          print(text.endswith('Bundeswehr'))
```

True

True

3.12.2 Listen

Inhaltsverzeichnis

- » 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Verweisungssyntax
- » 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
- » 3 Grundlegende Datenstrukturen
 - 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können mehrfach zuweisen
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutablen Objekte
 - 3.3 Ausgabe mit `print`
 - 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`$inf`)
 - 3.4.5 Not-a-Number (`NaN`)
 - 3.5 Mathematische Operationen
 - 3.5.1 Arithmetische Operationen
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
 - 3.6 Umwandlung des Datentyps
 - 3.7 Rundung
 - 3.7.1 Runden mit `round()`

Listen sind komplexe Variablen aus mehreren Unterelementen beliebigen Typs. Die Unterelemente können einzeln adressiert und auch geändert werden.

```
In [61]: # Liste
my_list = [1, 2, 5, 9]
my_list_mixed = [1, True, 'UniBwM']
print(my_list_mixed)
```

```
[1, True, 'UniBwM']
```

3.12.2.1 Adressierung von Listenelementen und Listenbereichen

Inhaltsverzeichnis

- » 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Verweisungssyntax
- » 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
- » 3 Grundlegende Datenstrukturen
 - » 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können mehrfach zuweisen
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutablen Objekte
 - 3.3 Ausgabe mit `print`
- » 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`$inf`)
 - 3.4.5 Not-a-Number (`NaN`)
- » 3.5 Mathematische Operationen
 - 3.5.1 Arithmetische Operationen
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
- 3.6 Umwandlung des Datentyps
- » 3.7 Rundung
 - 3.7.1 Runden mit `round()`

Unterelemente können einzeln adressiert und auch geändert werden.
Das Format ist dabei

`<name>[<start>:<end>:<step>]`

`<name>` – Name der Liste

`<start>` – Index des ersten Listenelements

`<end>` – Index des ersten Elements, das nicht mehr in der Liste sein soll

`<step>` – Schrittweite (-1 für rückwärts)

3.12.2.2 Einzelnes Listenelement

Inhaltsverzeichnis

- » 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Verweisungssyntax
- » 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
- » 3 Grundlegende Datenstrukturen
 - » 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können mehrfach verwendet werden
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutablen Objekte
 - 3.3 Ausgabe mit `print`
- » 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`$\inf`)
 - 3.4.5 Not-a-Number (`NaN`)
- » 3.5 Mathematische Operationen
 - 3.5.1 Arithmetische Operationen
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
- 3.6 Umwandlung des Datentyps
- » 3.7 Rundung
 - 3.7.1 Runden mit `round()`

```
In [62]: my_list = [1, 2, 5, 9]
          print(my_list[0])
          print(my_list[1])
          print(my_list[2])
```

```
1
2
5
```

Listenelemente können auch einzeln geändert werden:

Inhaltsverzeichnis

- ⌚ ⚙
 - ▼ 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Verweisungssyntax
 - ▼ 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
 - ▼ 3 Grundlegende Datenstrukturen
 - ▼ 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können verändert werden
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutablen Objekte
 - 3.3 Ausgabe mit `print`
 - ▼ 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`$\inf`)
 - 3.4.5 Not-a-Number (`NaN`)
 - ▼ 3.5 Mathematische Operationen
 - 3.5.1 Arithmetische Operationen
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
 - 3.6 Umwandlung des Datentyps
 - ▼ 3.7 Rundung
 - 3.7.1 Runden mit `round`

In [63]:

```
my_list_mixed = [1, True, 'UniBwM']
my_list_mixed[2] = 'LMU München'
print(my_list_mixed)
```

[1, True, 'LMU München']

3.12.2.3 Bereiche

Inhaltsverzeichnis

- » 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Verweisoperatoren
- » 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
- » 3 Grundlegende Datenstrukturen
 - 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können mehrfach zuweisen
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutablen Objekte
 - 3.3 Ausgabe mit `print`
 - 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`inf`)
 - 3.4.5 Not-a-Number (`NaN`)
 - 3.5 Mathematische Operationen
 - 3.5.1 Arithmetische Operationen
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
 - 3.6 Umwandlung des Datentyps
 - 3.7 Rundung
 - 3.7.1 Runden mit `round`

Man kann auch Bereiche adressieren. Dazu gibt man den Index des ersten Elements und das erste nicht mehr gewünschte Element an.

Wenn man einen der beiden Werte wegläßt, wird der Anfang bzw. das Ende der Liste verwendet.

```
In [5]: my_list = ['one', 'two', 'three',
               'four', 'five']
        # Alle ab dem zweiten Element
        print(my_list[1:])

['two', 'three', 'four', 'five']
```

```
In [6]: # Alle bis zum zweiten Element
        print(my_list[:2])
        # Alle vom zweiten bis zum dritten
        print(my_list[1:3])

['one', 'two']
['two', 'three']
```

Alle Elemente ohne die letzten beiden:

Inhaltsverzeichnis

- ⚙️
- ▼ 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Ver-
- ▼ 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
- ▼ 3 Grundlegende Datenstrukturen
 - ▼ 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können verändert werden
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutab
 - 3.3 Ausgabe mit `print`
- ▼ 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`$\inf`)
 - 3.4.5 Not-a-Number (`NaN`)
- ▼ 3.5 Mathematische Operationen
 - 3.5.1 Arithmetische Operat
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
- 3.6 Umwandlung des Datentyps
- ▼ 3.7 Rundung
 - 3.7.1 Runden mit `round`

```
In [114]: my_list = ['one', 'two', 'three', 'four', 'five']
          print(my_list[:-2])

['one', 'two', 'three']
```

Bereiche ersetzen

Inhaltsverzeichnis

- » 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Verweisungssyntax
- » 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
- » 3 Grundlegende Datenstrukturen
 - » 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können mehrfach zuweisen
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutablen Objekte
 - 3.3 Ausgabe mit `print`
- » 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`$\inf`)
 - 3.4.5 Not-a-Number (`NaN`)
- » 3.5 Mathematische Operationen
 - 3.5.1 Arithmetische Operationen
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
- 3.6 Umwandlung des Datentyps
- » 3.7 Rundung
 - 3.7.1 Runden mit `round()`

Man kann auch Bereiche einer Liste ändern oder die Liste dadurch verkürzen oder verlängern.

```
In [97]: my_list = ['one', 'two', 'three', 'four', 'five']
          my_list[1:3] = ['zwei', 'drei']
          print(my_list)
```

```
['one', 'zwei', 'drei', 'four', 'five']
```

Bereiche ersetzen

Inhaltsverzeichnis

- » 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Verweisungssyntax
- » 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
- » 3 Grundlegende Datenstrukturen
 - » 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können mehrfach verwendet werden
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutablen Objekte
 - 3.3 Ausgabe mit `print`
- » 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`inf`)
 - 3.4.5 Not-a-Number (`nan`)
- » 3.5 Mathematische Operationen
 - 3.5.1 Arithmetische Operationen
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
- » 3.6 Umwandlung des Datentyps
- » 3.7 Rundung
 - 3.7.1 Runden mit `round()`

Man kann auch Bereiche einer Liste ändern oder die Liste dadurch verkürzen oder verlängern.

```
In [97]: my_list = ['one', 'two', 'three', 'four', 'five']
          my_list[1:3] = ['zwei', 'drei']
          print(my_list)
```

```
['one', 'zwei', 'drei', 'four', 'five']
```

```
In [66]: my_list = ['one', 'two', 'three', 'four', 'five']
          my_list[0:2] = ['one_and_two']
          print(my_list)
```

```
['one_and_two', 'three', 'four', 'five']
```

Achtung: Wenn man einen Listen**BEREICH** ändert, muss man eine Liste übergeben.

Inhaltsverzeichnis

- ⌚⚙️
- ▼ 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Verweisungen
- ▼ 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
- ▼ 3 Grundlegende Datenstrukturen
 - ▼ 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können verändert werden
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutablen Objekte
 - 3.3 Ausgabe mit `print`
- ▼ 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`$\inf`)
 - 3.4.5 Not-a-Number (`NaN`)
- ▼ 3.5 Mathematische Operationen
 - 3.5.1 Arithmetische Operationen
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
- 3.6 Umwandlung des Datentyps
- ▼ 3.7 Rundung
 - 3.7.1 Runden mit `round()`

In [98]:

```
my_list = ['one', 'two', 'three', 'four', 'five']
my_list[0:2] = ['one_and_two']
print(my_list)
```

```
['one_and_two', 'three', 'four', 'five']
```

Sonst versucht Python, den Wert **als Liste seiner Unterelemente** verstehen, zum Beispiel eine Zeichenkette in eine Liste von Buchstaben zu zerlegen.

In [7]:

```
my_list = ['one', 'two', 'three', 'four', 'five']
my_list[0:2] = 'ABC' # ABC ist hier eine Zeichenkette
# Python versucht, den übergebenen Wert in Unterlemente zu zerlegen
# und diese einzufügen.
# Daher werden hier die drei Buchstaben A, B und C als neue Listenelemente
# eingefügt.
print(my_list)
```

```
['A', 'B', 'C', 'three', 'four', 'five']
```

Wenn man **ein** Listenelement ändert, muss man **ein Element übergeben**:

Inhaltsverzeichnis

- ⌚⚙️
- ▼ 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Verweisungssyntax
- ▼ 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
- ▼ 3 Grundlegende Datenstrukturen
 - ▼ 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können verändert werden
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutablen Objekte
 - 3.3 Ausgabe mit `print`
 - ▼ 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`$\inf`)
 - 3.4.5 Not-a-Number (`NaN`)
 - ▼ 3.5 Mathematische Operationen
 - 3.5.1 Arithmetische Operationen
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
 - 3.6 Umwandlung des Datentyps
 - ▼ 3.7 Rundung
 - 3.7.1 Runden mit `round`

```
In [69]: my_list = ['one', 'two', 'three', 'four', 'five']
my_list[2] = 'drei'
print(my_list)

['one', 'two', 'drei', 'four', 'five']
```

Wenn man an einer Position **MEHRERE** neue Elemente einfügen muss man diese Position **als Bereich der Länge 1** adressieren. Das geschieht im folgenden durch `my_list[2:3]`.

```
In [70]: my_list = ['one', 'two', 'three', 'four', 'five']
          my_list[2:3] = ['drei_a', 'drei_b']
          print(my_list)

['one', 'two', 'drei_a', 'drei_b', 'four', 'five']
```

Inhaltsverzeichnis

- ☞ *
 - ▼ 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Verweisungssyntax
 - ▼ 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
 - ▼ 3 Grundlegende Datenstrukturen
 - ▼ 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können verändert werden
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutablen Objekte
 - 3.3 Ausgabe mit `print`
 - ▼ 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`$\inf`)
 - 3.4.5 Not-a-Number (`NaN`)
 - ▼ 3.5 Mathematische Operationen
 - 3.5.1 Arithmetische Operationen
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
 - 3.6 Umwandlung des Datentyps
 - ▼ 3.7 Rundung
 - 3.7.1 Runden mit `round`

Inhaltsverzeichnis

- ☞ *
 - ▼ 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Verweisungssyntax
 - ▼ 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
 - ▼ 3 Grundlegende Datenstrukturen
 - ▼ 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können mehrfach zuweisen
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutablen
 - 3.3 Ausgabe mit `print`
 - ▼ 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`inf`)
 - 3.4.5 Not-a-Number (`nan`)
 - ▼ 3.5 Mathematische Operationen
 - 3.5.1 Arithmetische Operationen
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
 - 3.6 Umwandlung des Datentyps
 - ▼ 3.7 Rundung
 - 3.7.1 Runden mit `round`

Wenn man an einer Position **MEHRERE** neue Elemente einfügen muss man diese Position **als Bereich der Länge 1** adressieren. Das geschieht im folgenden durch `my_list[2:3]`.

```
In [70]: my_list = ['one', 'two', 'three', 'four', 'five']
          my_list[2:3] = ['drei_a', 'drei_b']
          print(my_list)

['one', 'two', 'drei_a', 'drei_b', 'four', 'five']
```

```
In [71]: # Sonst würde an dieser Stelle eine Liste als Element eingefügt:
          my_list = ['one', 'two', 'three', 'four', 'five']
          my_list[2] = ['drei_a', 'drei_b']
          print(my_list)

['one', 'two', ['drei_a', 'drei_b'], 'four', 'five']
```

Schrittweite

Inhaltsverzeichnis

- » 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Verweisungssyntax
- » 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
- » 3 Grundlegende Datenstrukturen
 - » 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können mehrfach zuweisbar sein
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutablen Objekte
 - 3.3 Ausgabe mit `print`
- » 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`$\inf`)
 - 3.4.5 Not-a-Number (`NaN`)
- » 3.5 Mathematische Operationen
 - 3.5.1 Arithmetische Operationen
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
- 3.6 Umwandlung des Datentyps
- » 3.7 Rundung
 - 3.7.1 Runden mit `round()`

```
In [100]: my_list = ['one', 'two', 'three', 'four', 'five', 'six']
          # Jedes zweite Element zwischen 0 und 4
          print(my_list[0:5:2])

['one', 'three', 'five']
```

Schrittweite

Inhaltsverzeichnis

- » 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Verweisoperatoren
- » 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
- » 3 Grundlegende Datenstrukturen
 - 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können mehrfach zuweisbar sein
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutablen Objekte
 - 3.3 Ausgabe mit `print`
- » 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`$inf`)
 - 3.4.5 Not-a-Number (`NaN`)
- » 3.5 Mathematische Operationen
 - 3.5.1 Arithmetische Operationen
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
- » 3.6 Umwandlung des Datentyps
- » 3.7 Rundung
 - 3.7.1 Runden mit `round()`

Man kann auch eine Schrittweite angeben und damit erreichen, dass nur jedes n-te Element aus der Liste zurückgeliefert wird.

```
In [100]: my_list = ['one', 'two', 'three', 'four', 'five', 'six']
          # Jedes zweite Element zwischen 0 und 4
          print(my_list[0:5:2])

['one', 'three', 'five']
```

```
In [103]: # Jedes dritte Element
          print(my_list[::-3])

['one', 'four']
```

Negative Schrittweite für rückwärts:

Inhaltsverzeichnis

- ⌚⚙️
- ▼ 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Verweisungssyntax
- ▼ 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
- ▼ 3 Grundlegende Datenstrukturen
 - ▼ 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können verändert werden
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutablen Objekte
 - 3.3 Ausgabe mit `print`
 - ▼ 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`$\inf`)
 - 3.4.5 Not-a-Number (`NaN`)
 - ▼ 3.5 Mathematische Operationen
 - 3.5.1 Arithmetische Operationen
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
 - 3.6 Umwandlung des Datentyps
 - ▼ 3.7 Rundung
 - 3.7.1 Runden mit `round`

```
In [8]: my_list = ['one', 'two', 'three', 'four', 'five', 'six']

# Alle Elemente in umgekehrter Reihenfolge
print(my_list[::-1])

['six', 'five', 'four', 'three', 'two', 'one']
```

3.12.2.4 Verlängern der Liste mit `append()`

Inhaltsverzeichnis

- ☞ 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Verweisungssyntax
- ☞ 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
- ☞ 3 Grundlegende Datenstrukturen
 - 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können mehrfach verwendet werden
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutablen Objekte
 - 3.3 Ausgabe mit `print`
 - 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`$\inf`)
 - 3.4.5 Not-a-Number (`NaN`)
 - 3.5 Mathematische Operationen
 - 3.5.1 Arithmetische Operationen
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
 - 3.6 Umwandlung des Datentyps
 - 3.7 Rundung
 - 3.7.1 Runden mit `round()`

Mit der Funktion `append(<wert>)` kann man einen Wert am Ende einer Liste hinzufügen.

```
In [105]: my_list = ['one', 'two']
my_list.append('three')
print(my_list)
```

```
['one', 'two', 'three']
```

3.12.2.5 Erweitern der Liste mit `extend()`

Inhaltsverzeichnis

- ☞ 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Verweisungssyntax
- ☞ 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
- ☞ 3 Grundlegende Datenstrukturen
 - 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können mehrfach zugeordnet werden
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutablen Objekte
 - 3.3 Ausgabe mit `print`
 - 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`$inf`)
 - 3.4.5 Not-a-Number (`NaN`)
 - 3.5 Mathematische Operationen
 - 3.5.1 Arithmetische Operationen
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
 - 3.6 Umwandlung des Datentyps
 - 3.7 Rundung
 - 3.7.1 Runden mit `round()`

Mit der Funktion `extend(<neue_liste>)` kann man eine Liste an das Ende der Liste hinzufügen.

```
In [106]: my_list = ['one', 'two']
my_list.extend(['three', 'four'])
print(my_list)

['one', 'two', 'three', 'four']
```

Kontrollfrage:

Was passiert, wenn Sie der Methode `append()` als Parameter eine **LISTE** übergeben?

```
In [ ]: my_list = ['one', 'two']
         my_list.append(['three', 'four'])
```

Inhaltsverzeichnis

- ☞ 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Verweisungssyntax
- ☞ 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
- ☞ 3 Grundlegende Datenstrukturen
 - 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können verändert werden
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutablen Objekte
 - 3.3 Ausgabe mit `print`
 - 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`$\inf`)
 - 3.4.5 Not-a-Number (`NaN`)
 - 3.5 Mathematische Operationen
 - 3.5.1 Arithmetische Operationen
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
 - 3.6 Umwandlung des Datentyps
 - 3.7 Rundung
 - 3.7.1 Runden mit `round()`

Kontrollfrage:

Was passiert, wenn Sie der Methode `append()` als Parameter eine **LISTE** übergeben?

```
In [ ]: my_list = ['one', 'two']
         my_list.append(['three', 'four'])
```

```
In [74]: print(my_list)
['one', 'two', ['three', 'four']]
```

Wie Sie sehen, wird in diesem Fall die Liste `['three', 'four']` als Element an dritter Stelle eingefügt. Das dritte Element ist dann also selbst eine Liste.

Kontrollfrage

Inhaltsverzeichnis

- » 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Verweisungssyntax
- » 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
- » 3 Grundlegende Datenstrukturen
 - » 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können mehrfach zuweisen
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutablen Objekte
 - 3.3 Ausgabe mit `print`
 - » 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`$\inf`)
 - 3.4.5 Not-a-Number (`NaN`)
 - » 3.5 Mathematische Operationen
 - 3.5.1 Arithmetische Operationen
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
 - 3.6 Umwandlung des Datentyps
 - » 3.7 Rundung
 - 3.7.1 Runden mit `round()`

Was passiert, wenn Sie der Methode `extend()` als Parameter einen einzelnen Wert übergeben?

```
In [ ]: my_list = ['one', 'two']
          my_list.extend('three')
```

Kontrollfrage

Inhaltsverzeichnis

- » 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Verweisungssyntax
- » 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
- » 3 Grundlegende Datenstrukturen
 - » 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können mehrfach zuweisen
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutablen Objekte
 - 3.3 Ausgabe mit `print`
- » 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`inf`)
 - 3.4.5 Not-a-Number (`NaN`)
- » 3.5 Mathematische Operationen
 - 3.5.1 Arithmetische Operationen
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
- 3.6 Umwandlung des Datentyps
- » 3.7 Rundung
 - 3.7.1 Runden mit `round`

Was passiert, wenn Sie der Methode `extend()` als Parameter einen einzelnen Wert übergeben?

```
In [ ]: my_list = ['one', 'two']
         my_list.extend('three')
```

```
In [75]: print(my_list)
['one', 'two', 't', 'h', 'r', 'e', 'e']
```

Wie Sie sehen, versucht Python, den Wert **als Liste seiner Unterelemente** zu verstehen, zum Beispiel eine Zeichenkette in eine Liste von Buchstaben zu zerlegen.

Wenn eine atomare Variable übergeben wird und diese Zerlegung nicht möglich ist, gibt es eine Fehlermeldung:

Inhaltsverzeichnis



- ▼ 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Verweis
- ▼ 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
- ▼ 3 Grundlegende Datenstrukturen
 - 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können mehrfach zuweisen
 - 3.1.3 Mehrfachzuweisungen sind problematisch
 - 3.2 Mutable und Immutablen Objekte
 - 3.3 Ausgabe mit `print`
 - 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`$inf`)
 - 3.4.5 Not-a-Number (`NaN`)
 - 3.5 Mathematische Operationen
 - 3.5.1 Arithmetische Operationen
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
 - 3.6 Umwandlung des Datentyps
 - 3.7 Rundung
 - 3.7.1 Runden mit `round()`

```
In [107]: my_list = ['one', 'two']
          my_list.extend(1)
          print(my_list)
```

```
-----  
TypeError
```

```
Traceback (most recent call last)
```

```
<ipython-input-107-f117dde7810b> in <module>
      1 my_list = ['one', 'two']
      ----> 2 my_list.extend(1)
            3 print(my_list)
```

```
TypeError: 'int' object is not iterable
```

3.12.2.6 Entfernen von Elementen aus einer Liste

Inhaltsverzeichnis

- » 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Ver-
- » 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
- » 3 Grundlegende Datenstrukturen
 - » 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können mehrfach zuweisen
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutab
 - 3.3 Ausgabe mit `print`
- » 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`$inf`)
 - 3.4.5 Not-a-Number (`NaN`)
- » 3.5 Mathematische Operationen
 - 3.5.1 Arithmetische Operat
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
- 3.6 Umwandlung des Datentyps
- » 3.7 Rundung
 - 3.7.1 Runden mit `round`

Wenn man ein Element aus einer Liste entfernen möchte, kann man dies über seinen **Wert** mit der Funktion `remove(<wert>)` erreichen. Wenn stattdessen die **Position** des Elementes bekannt ist, gibt es eine Funktion `pop(<position>)`.

```
In [85]: liste = ['Peter', 'Paul', 'Mary']
liste.remove('Paul')
print(liste)
```

```
['Peter', 'Mary']
```

3.12.2.6 Entfernen von Elementen aus einer Liste

Inhaltsverzeichnis

- » 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Verweisungssyntax
- » 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
- » 3 Grundlegende Datenstrukturen
 - » 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können mehrfach zuweisen
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutablen Objekte
 - 3.3 Ausgabe mit `print`
- » 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`inf`)
 - 3.4.5 Not-a-Number (`NaN`)
- » 3.5 Mathematische Operationen
 - 3.5.1 Arithmetische Operationen
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
- 3.6 Umwandlung des Datentyps
- » 3.7 Rundung
 - 3.7.1 Runden mit `round()`

Wenn man ein Element aus einer Liste entfernen möchte, kann man dies über seinen **Wert** mit der Funktion `remove(<wert>)` erreichen. Wenn stattdessen die **Position** des Elementes bekannt ist, gibt es eine Funktion `pop(<position>)`.

In [85]:

```
liste = ['Peter', 'Paul', 'Mary']
liste.remove('Paul')
print(liste)
```

```
['Peter', 'Mary']
```

In [87]:

```
liste = ['Peter', 'Paul', 'Mary']
liste.pop(0)
print(liste)
```

```
['Paul', 'Mary']
```

pop () (ohne Parameter) entfernt das Element am Ende der Liste

Inhaltsverzeichnis



- ▼ 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Verweisungssyntax
- ▼ 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
- ▼ 3 Grundlegende Datenstrukturen
 - ▼ 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können verändert werden
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutablen Objekte
 - 3.3 Ausgabe mit `print`
 - ▼ 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`$\inf`)
 - 3.4.5 Not-a-Number (`NaN`)
 - ▼ 3.5 Mathematische Operationen
 - 3.5.1 Arithmetische Operationen
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
 - 3.6 Umwandlung des Datentyps
 - ▼ 3.7 Rundung
 - 3.7.1 Runden mit `round()`

```
In [115]: liste = ['Peter', 'Paul', 'Mary']
liste.pop()
print(liste)

['Peter', 'Paul']
```

pop () (ohne Parameter) entfernt das Element am Ende der Liste

Inhaltsverzeichnis



- ▼ 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Verweisungssyntax
- ▼ 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
- ▼ 3 Grundlegende Datenstrukturen
 - ▼ 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können verändert werden
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutablen Objekte
 - 3.3 Ausgabe mit `print`
- ▼ 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`inf`)
 - 3.4.5 Not-a-Number (`nan`)
- ▼ 3.5 Mathematische Operationen
 - 3.5.1 Arithmetische Operationen
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
- 3.6 Umwandlung des Datentyps
- ▼ 3.7 Rundung
 - 3.7.1 Runden mit `round()`

```
In [115]: liste = ['Peter', 'Paul', 'Mary']
          liste.pop()
          print(liste)

          ['Peter', 'Paul']
```

`pop ()` mit oder ohne Parameter liefert das entfernte Element als Ergebnis zurück.

```
In [88]: liste = ['Peter', 'Paul', 'Mary']
          print(liste.pop(0))

          Peter
```

Stapel (Stack) und Last in, first out

Inhaltsverzeichnis

- ☞ 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Ver-
- ☞ 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
- ☞ 3 Grundlegende Datenstrukturen
 - 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können verändert werden
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutab
 - 3.3 Ausgabe mit `print`
 - 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`$inf`)
 - 3.4.5 Not-a-Number (`NaN`)
 - 3.5 Mathematische Operationen
 - 3.5.1 Arithmetische Operat
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
 - 3.6 Umwandlung des Datentyps
 - 3.7 Rundung
 - 3.7.1 Runden mit `round`

`pop()` kann man sehr verwenden, um einen Stapel (Stack) zu implementieren oder in anderen Zusammenhängen das LIFO-Prinzip ("last in, first out") anzuwenden.

Position	Name
2	Paul
1	Mary
0	Peter

Ein neues Element Linda wird oben auf den Stapel gelegt:

Position	Name
3	Linda
2	Paul
1	Mary
0	Peter

Inhaltsverzeichnis



- ▼ 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Verweisungssyntax
- ▼ 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
- ▼ 3 Grundlegende Datenstrukturen
 - ▼ 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können verändert werden
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutablen Objekte
 - 3.3 Ausgabe mit `print`
- ▼ 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`$\inf`)
 - 3.4.5 Not-a-Number (`NaN`)
- ▼ 3.5 Mathematische Operationen
 - 3.5.1 Arithmetische Operationen
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
- 3.6 Umwandlung des Datentyps
- ▼ 3.7 Rundung
 - 3.7.1 Runden mit `round`

In [11]:

```
mitarbeiter = ['Peter', 'Mary', 'Paul', 'Linda']
print('Unser Team: ' + str(mitarbeiter))
mitarbeiter.append('Frank')
print('Unser Team: ' + str(mitarbeiter))
# Wer zuletzt eingestellt wurde, wird zuerst wieder entlassen.
name = mitarbeiter.pop()
print('Leider müssen wir ' + str(name) + ' wieder entlassen.')
print('Unser Team: ' + str(mitarbeiter))
```

```
Unser Team: ['Peter', 'Mary', 'Paul', 'Linda']
Unser Team: ['Peter', 'Mary', 'Paul', 'Linda', 'Frank']
Leider müssen wir Frank wieder entlassen.
Unser Team: ['Peter', 'Mary', 'Paul', 'Linda']
```

3.12.2.7 Sortieren von Listen

Inhaltsverzeichnis

- » 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Verweisungssyntax
- » 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
- » 3 Grundlegende Datenstrukturen
 - » 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können verändert werden
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutablen Objekte
 - 3.3 Ausgabe mit `print`
- » 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`inf`)
 - 3.4.5 Not-a-Number (`nan`)
- » 3.5 Mathematische Operationen
 - 3.5.1 Arithmetische Operationen
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
- 3.6 Umwandlung des Datentyps
- » 3.7 Rundung
 - 3.7.1 Runden mit `round()`

Man kann Listen einfach sortieren. Dazu gibt es die Funktion `sort()`. Sie sortiert die Elemente in der ursprünglichen Liste um.

```
In [117]: my_list = [1, 6, 5, 3, 2, 4]
my_list.sort()
print(my_list)

woerter_liste = ['Peter', 'Mary', 'Zoe', 'Anton']
woerter_liste.sort()
print(woerter_liste)

[1, 2, 3, 4, 5, 6]
['Anton', 'Mary', 'Peter', 'Zoe']
```

Achtung: `sort()` ist eine Funktion, die das Objekt verändert. wird keine sortierte Version zurückgeliefert, sondern das Objekt bisherigen Ort sortiert.

```
In [28]: meine_liste = [1, 2, 3, 0, 7, 4, 13]
          print(meine_liste.sort())
```

```
None
```

Inhaltsverzeichnis

- ☞ 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Verweisungen
- ☞ 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
- ☞ 3 Grundlegende Datenstrukturen
 - 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können mehrfach zuweisen
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutablen Objekte
 - 3.3 Ausgabe mit `print`
 - 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`$\inf`)
 - 3.4.5 Not-a-Number (`$\NaN`)
 - 3.5 Mathematische Operationen
 - 3.5.1 Arithmetische Operationen
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
 - 3.6 Umwandlung des Datentyps
 - 3.7 Rundung
 - 3.7.1 Runden mit `round`

Inverse Sortierfolge

Inhaltsverzeichnis

- » 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Verweisungssyntax
- » 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
- » 3 Grundlegende Datenstrukturen
 - » 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können mehrfach verwendet werden
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutablen Objekte
 - 3.3 Ausgabe mit `print`
- » 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`$\inf`)
 - 3.4.5 Not-a-Number (`NaN`)
- » 3.5 Mathematische Operationen
 - 3.5.1 Arithmetische Operationen
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
- 3.6 Umwandlung des Datentyps
- » 3.7 Rundung
 - 3.7.1 Runden mit `round()`

Mit dem Parameter `reverse=True` kann man die Sortierreihenfolge umkehren.

```
In [79]: my_list = [1, 6, 5, 3, 2, 4]
my_list.sort(reverse=True)
print(my_list)
```

```
[6, 5, 4, 3, 2, 1]
```

Inhaltsverzeichnis

- ▼ 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Verweisoperator
- ▼ 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
- ▼ 3 Grundlegende Datenstrukturen
 - ▼ 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können verschachtelt sein
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutablen Objekte
 - 3.3 Ausgabe mit `print`
- ▼ 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`$\inf`)
 - 3.4.5 Not-a-Number (`NaN`)
- ▼ 3.5 Mathematische Operationen
 - 3.5.1 Arithmetische Operationen
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
- 3.6 Umwandlung des Datentyps
- ▼ 3.7 Rundung
 - 3.7.1 Runden mit `round()`

Es ist möglich, **Listen mit verschiedenen Datentypen** zu sortieren, sofern für jedes mögliche Wertepaar ein Vergleichsoperator definiert ist.

In [118]:

```
# Gemischte Liste
gemischte_liste_1 = [1, 1.5, 2, 7.2]
gemischte_liste_1.sort()
print(gemischte_liste_1)
```

```
[1, 1.5, 2, 7.2]
```

Die Sortierung funktioniert aber nicht, wenn eine Liste Elemente enthält, für die kein Vergleichsoperator definiert ist.

Inhaltsverzeichnis

- ⚙️
- ▼ 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Verweisoperatoren
- ▼ 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
- ▼ 3 Grundlegende Datenstrukturen
 - 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können mehrfach zuweisen
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutablen Objekte
 - 3.3 Ausgabe mit `print`
 - 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`$inf`)
 - 3.4.5 Not-a-Number (`NaN`)
 - 3.5 Mathematische Operationen
 - 3.5.1 Arithmetische Operationen
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
 - 3.6 Umwandlung des Datentyps
 - 3.7 Rundung
 - 3.7.1 Runden mit `round()`

In [110]:

```
gemischte_liste_2 = [1, 'Zoe', False]
gemischte_liste_2.sort()
print(gemischte_liste_2)
```

TypeError

Traceback (most recent call last)

```
<ipython-input-110-054ac3c452ca> in <module>
      1 gemischte_liste_2 = [1, 'Zoe', False]
----> 2 gemischte_liste_2.sort()
      3 print(gemischte_liste_2)
```

TypeError: '<' not supported between instances of 'str' and 'int'

3.12.2.8 Suchen in Listen

Inhaltsverzeichnis



- ▼ 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Verweisungssyntax
- ▼ 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
- ▼ 3 Grundlegende Datenstrukturen
 - ▼ 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können verändert werden
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutablen Objekte
 - 3.3 Ausgabe mit `print`
 - ▼ 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`$\inf`)
 - 3.4.5 Not-a-Number (`NaN`)
 - ▼ 3.5 Mathematische Operationen
 - 3.5.1 Arithmetische Operationen
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
 - 3.6 Umwandlung des Datentyps
 - ▼ 3.7 Rundung
 - 3.7.1 Runden mit `round`

```
In [80]: # Prüfen, ob Element in Liste enthalten
my_liste_3 = [1, 4, 9, 7]
print(2 in my_liste_3)
```

False

3.12.2.8 Suchen in Listen

Inhaltsverzeichnis

- ⌚ ↘ 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Verweisungssyntax
- ⌚ ↘ 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
- ⌚ ↘ 3 Grundlegende Datenstrukturen
 - 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können mehrfach verwendet werden
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutablen Objekte
 - 3.3 Ausgabe mit `print`
 - 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`$\inf`)
 - 3.4.5 Not-a-Number (`NaN`)
 - 3.5 Mathematische Operationen
 - 3.5.1 Arithmetische Operationen
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
 - 3.6 Umwandlung des Datentyps
 - 3.7 Rundung
 - 3.7.1 Runden mit `round`

```
In [80]: # Prüfen, ob Element in Liste enthalten  
my_liste_3 = [1, 4, 9, 7]  
print(2 in my_liste_3)
```

False

```
In [83]: my_liste_4 = ['Hepp', 'Mueller', 'Meier']  
if 'Mueller' in my_liste_4:  
    print('Täter gefunden!')
```

Täter gefunden!

index (<wert>) liefert die erste Position eines passenden Wertes

Inhaltsverzeichnis



- ▼ 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Verweisungssyntax
- ▼ 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
- ▼ 3 Grundlegende Datenstrukturen
 - ▼ 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können verändert werden
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutablen Objekte
 - 3.3 Ausgabe mit `print`
 - ▼ 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`$\inf`)
 - 3.4.5 Not-a-Number (`NaN`)
 - ▼ 3.5 Mathematische Operationen
 - 3.5.1 Arithmetische Operationen
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
 - 3.6 Umwandlung des Datentyps
 - ▼ 3.7 Rundung
 - 3.7.1 Runden mit `round()`

```
In [84]: my_liste_4 = ['Hepp', 'Mueller', 'Meier']
          if 'Mueller' in my_liste_4:
              print(my_liste_4.index('Mueller'))
```

1

3.12.3 Tuples

Inhaltsverzeichnis

- » 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Verweisungssyntax
- » 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
- » 3 Grundlegende Datenstrukturen
 - » 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können verändert werden
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutablen Objekte
 - 3.3 Ausgabe mit `print`
- » 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`$\inf`)
 - 3.4.5 Not-a-Number (`NaN`)
- » 3.5 Mathematische Operationen
 - 3.5.1 Arithmetische Operationen
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
- 3.6 Umwandlung des Datentyps
- » 3.7 Rundung
 - 3.7.1 Runden mit `round()`

Tuples sind strukturierte Datentypen aus mehreren Elementen. Sie sind **Immutables**, können also nicht verändert werden. Man kann aber natürlich ein neues Tuples aus geänderten Werten erzeugen.

3.12.3 Tuples

Inhaltsverzeichnis

- » 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Verweisungssyntax
- » 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
- » 3 Grundlegende Datenstrukturen
 - » 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können verschachtelt werden
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutablen Objekte
 - 3.3 Ausgabe mit `print`
- » 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`inf`)
 - 3.4.5 Not-a-Number (`nan`)
- » 3.5 Mathematische Operationen
 - 3.5.1 Arithmetische Operationen
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
- 3.6 Umwandlung des Datentyps
- » 3.7 Rundung
 - 3.7.1 Runden mit `round()`

Tuples sind strukturierte Datentypen aus mehreren Elementen. Sie sind **Immutables**, können also nicht verändert werden. Man kann aber natürlich ein neues Tuples aus geänderten Werten erzeugen.

3.12.3.1 Beispiele

```
In [83]: # Tuple
my_tuple = (1, 3, 9)
my_tuple_mixed = (1, True, 'UniBwM')
latitude = 48.0803
longitude = 11.6382
geo_position = (latitude, longitude)
```

3.12.3.2 Entpacken eines Tuples in mehrere Zielvariablen

Inhaltsverzeichnis

- » 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Verweisungssyntax
- » 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
- » 3 Grundlegende Datenstrukturen
 - » 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können verschachtelt werden
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutablen Objekte
 - 3.3 Ausgabe mit `print`
- » 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`$\inf`)
 - 3.4.5 Not-a-Number (`NaN`)
- » 3.5 Mathematische Operationen
 - 3.5.1 Arithmetische Operationen
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
- 3.6 Umwandlung des Datentyps
- » 3.7 Rundung
 - 3.7.1 Runden mit `round()`

Man kann ein Tupel elegant in seine Bestandteile zerlegen und die einzelnen Variablen zuweisen. Voraussetzung ist nur, dass auf der linken Seite ebensoviele Variablen genannt werden wie das Tupel bestandteile hat.

```
In [14]: geo_position = (48.0803, 11.6382)
lat, lon = geo_position
print(lat)
```

```
48.0803
```

3.12.3.2 Entpacken eines Tuples in mehrere Zielvariablen

Inhaltsverzeichnis

- » 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Verweisungssyntax
- » 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
- » 3 Grundlegende Datenstrukturen
 - » 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können verschachtelt werden
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutablen Objekte
 - 3.3 Ausgabe mit `print`
- » 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`inf`)
 - 3.4.5 Not-a-Number (`nan`)
- » 3.5 Mathematische Operationen
 - 3.5.1 Arithmetische Operationen
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
- 3.6 Umwandlung des Datentyps
- » 3.7 Rundung
 - 3.7.1 Runden mit `round`

Man kann ein Tupel elegant in seine Bestandteile zerlegen und die einzelnen Variablen zuweisen. Voraussetzung ist nur, dass auf der linken Seite ebensoviele Variablen genannt werden wie das Tupel bestandteile hat.

```
In [14]: geo_position = (48.0803, 11.6382)
lat, lon = geo_position
print(lat)
```

48.0803

```
In [15]: # Das funktioniert auch mit anderen komplexen Datentypen
text = "ABC"
x, y, z = text
print(x)
```

A

Auch die Elemente eines Tuples können einzeln adressiert werden:

```
In [85]: print(geo_position[0])
```

```
48.0803
```

```
In [16]: print(geo_position[1])
```

```
11.6382
```

Inhaltsverzeichnis



- ▼ 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Verweisungssyntax
- ▼ 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
- ▼ 3 Grundlegende Datenstrukturen
 - ▼ 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können verändert werden
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutablen Objekte
 - 3.3 Ausgabe mit `print`
 - ▼ 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`$\inf`)
 - 3.4.5 Not-a-Number (`NaN`)
 - ▼ 3.5 Mathematische Operationen
 - 3.5.1 Arithmetische Operationen
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
 - 3.6 Umwandlung des Datentyps
 - ▼ 3.7 Rundung
 - 3.7.1 Runden mit `round`

Die Unterelemente eines Tuples können aber nicht geändert werden

Inhaltsverzeichnis



- ▼ 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Verweisungssyntax
- ▼ 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
- ▼ 3 Grundlegende Datenstrukturen
 - ▼ 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können verändert werden
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutablen Objekte
 - 3.3 Ausgabe mit `print`
- ▼ 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`$\inf`)
 - 3.4.5 Not-a-Number (`NaN`)
- ▼ 3.5 Mathematische Operationen
 - 3.5.1 Arithmetische Operationen
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
- 3.6 Umwandlung des Datentyps
- ▼ 3.7 Rundung
 - 3.7.1 Runden mit `round()`

```
In [86]: geo_position[0] = 44.123
```

```
-----  
TypeError
```

```
Traceback (most recent call last)
```

```
<ipython-input-86-623b6a40f573> in <module>
```

```
    1 # Die Unterelemente eines Tuples können aber nicht geändert werden:  
----> 2 geo_position[0] = 44.123
```

```
TypeError: 'tuple' object does not support item assignment
```

3.12.4 Dictionaries

Inhaltsverzeichnis

- » 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Ver-
- » 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
- » 3 Grundlegende Datenstrukturen
 - 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können mehrfach zuweisen
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutabile Objekte
 - 3.3 Ausgabe mit `print`
 - 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`inf`)
 - 3.4.5 Not-a-Number (`NaN`)
 - 3.5 Mathematische Operationen
 - 3.5.1 Arithmetische Operationen
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
 - 3.6 Umwandlung des Datentyps
 - 3.7 Rundung
 - 3.7.1 Runden mit `round`

Dictionaries sind Datenstrukturen, in denen Paare aus Eigenschaften (Properties) und Werten (Values) gespeichert werden können. Diese Werte können über ihren Namen angesprochen werden:

In [22]:

```
my_dict_empty = {}
my_dict_simple = {'name' : 'Martin Hepp'}
my_dict = {'name' : 'Martin Hepp',
           'fakultaet' : 'WOW',
           'geburtsjahr' : 1971}
print(my_dict['name'])
print(my_dict['fakultaet'])
```

Martin Hepp

WOW

Inhaltsverzeichnis



- ▼ 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Verweisungssyntax
- ▼ 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
- ▼ 3 Grundlegende Datenstrukturen
 - ▼ 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können mehrfachzuweisen
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutablen Objekte
 - 3.3 Ausgabe mit `print`
- ▼ 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`$inf`)
 - 3.4.5 Not-a-Number (`$nan`)
- ▼ 3.5 Mathematische Operationen
 - 3.5.1 Arithmetische Operationen
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
- 3.6 Umwandlung des Datentyps
- ▼ 3.7 Rundung
 - 3.7.1 Runden mit `round`

In [23]:

```
# Elemente können geändert und hinzugefügt werden
print(my_dict)
my_dict['fakultaet'] = 'INF'
print(my_dict)
my_dict['lieblingsvorlesung'] = 'Programmierung in Python'
print(my_dict)

{'name': 'Martin Hepp', 'fakultaet': 'WOW', 'geburtsjahr': 1971}
{'name': 'Martin Hepp', 'fakultaet': 'INF', 'geburtsjahr': 1971}
{'name': 'Martin Hepp', 'fakultaet': 'INF', 'geburtsjahr': 1971, 'lieblingsvorlesung': 'Programmierung in Python'}
```

Wenn es den Schlüssel ('key') nicht gibt, wird eine Fehlermeldung produziert:

Inhaltsverzeichnis

- ⌚
- ▼ 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Verweisungssyntax
- ▼ 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
- ▼ 3 Grundlegende Datenstrukturen
 - 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können verändert werden
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutablen Objekte
 - 3.3 Ausgabe mit `print`
 - 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`$\inf`)
 - 3.4.5 Not-a-Number (`NaN`)
 - 3.5 Mathematische Operationen
 - 3.5.1 Arithmetische Operationen
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
 - 3.6 Umwandlung des Datentyps
 - 3.7 Rundung
 - 3.7.1 Runden mit `round`

```
In [24]: print(my_dict['einkommen'])
```

```
-----  
KeyError
```

```
Traceback (most recent call last)
```

```
<ipython-input-24-9a15f52b31bf> in <module>
```

```
----> 1 print(my_dict['einkommen'])
```

```
KeyError: 'einkommen'
```

Das kann man mit der Methode `get(<eigenschaft>)` vermeiden:

```
In [25]: print(my_dict.get('einkommen'))
```

```
None
```

Inhaltsverzeichnis

- ⌚⚙️
- ▼ 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Verweisungssyntax
- ▼ 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
- ▼ 3 Grundlegende Datenstrukturen
 - ▼ 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können verändert werden
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutablen Objekte
 - 3.3 Ausgabe mit `print`
 - ▼ 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`$\inf`)
 - 3.4.5 Not-a-Number (`$\NaN`)
 - ▼ 3.5 Mathematische Operationen
 - 3.5.1 Arithmetische Operationen
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
 - 3.6 Umwandlung des Datentyps
 - ▼ 3.7 Rundung
 - 3.7.1 Runden mit `round`

Man kann auch einen Default-Wert vorgeben (normalerweise `None`).

Dieser Wert wird zurückgeliefert, wenn es die Eigenschaft bisher nicht gibt.

```
In [26]: print(my_dict.get('einkommen', 'Unbekannt'))  
print(my_dict.get('einkommen', 0))
```

Unbekannt

0

Beispiel:

Inhaltsverzeichnis

- ⌚⚙️
- ▼ 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Verweisungssyntax
- ▼ 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
- ▼ 3 Grundlegende Datenstrukturen
 - ▼ 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können verändert werden
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutablen Objekte
 - 3.3 Ausgabe mit `print`
- ▼ 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`$\inf`)
 - 3.4.5 Not-a-Number (`NaN`)
- ▼ 3.5 Mathematische Operationen
 - 3.5.1 Arithmetische Operationen
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
- 3.6 Umwandlung des Datentyps
- ▼ 3.7 Rundung
 - 3.7.1 Runden mit `round`

In [92]:

```
adresse = {}  
print(adresse)  
  
{}
```

Beispiel:

Inhaltsverzeichnis

- ⌚⚙️
- ▼ 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Verweisungssyntax
- ▼ 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
- ▼ 3 Grundlegende Datenstrukturen
 - ▼ 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können mehrfach zuweisen
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutablen Objekte
 - 3.3 Ausgabe mit `print`
- ▼ 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`inf`)
 - 3.4.5 Not-a-Number (`NaN`)
- ▼ 3.5 Mathematische Operationen
 - 3.5.1 Arithmetische Operationen
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
- 3.6 Umwandlung des Datentyps
- ▼ 3.7 Rundung
 - 3.7.1 Runden mit `round`

```
In [92]: adresse = {}  
        print(adresse)  
  
        {}
```

```
In [93]: adresse = {}  
        adresse['plz'] = '85577'  
        print(adresse['plz'])  
        adresse['sonderfeld'] = 'Bemerkungen bitte hier'  
        print(adresse)
```

```
85577  
{'plz': '85577', 'sonderfeld': 'Bemerkungen bitte hier'}
```

Liste von Dictionaries:

Inhaltsverzeichnis



- ▼ 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Verweisungssyntax
- ▼ 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
- ▼ 3 Grundlegende Datenstrukturen
 - ▼ 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können verändert werden
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutablen Objekte
 - 3.3 Ausgabe mit `print`
 - ▼ 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`inf`)
 - 3.4.5 Not-a-Number (`NaN`)
 - ▼ 3.5 Mathematische Operationen
 - 3.5.1 Arithmetische Operationen
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
 - 3.6 Umwandlung des Datentyps
 - ▼ 3.7 Rundung
 - 3.7.1 Runden mit `round()`

In [94]:

```
gast_1 = {'name' : 'Frank Farian'}
gast_2 = {'name' : 'Lady Gaga'}
gast_3 = {'name' : 'John Lennon'}

gaesteliste = []
gaesteliste.append(gast_1)
gaesteliste.append(gast_2)
gaesteliste.append(gast_3)
gast_2['bemerkung'] = 'Supercool!'
print(gaesteliste)
```

```
[{'name': 'Frank Farian'}, {'name': 'Lady Gaga', 'bemerkung': 'Supercool!'}
 {'name': 'John Lennon'}]
```

```
In [95]: for gast in gaesteliste:  
    print(gast['name'], gast.get('bemerkung', ''))
```

Frank Farian
Lady Gaga Supercool!
John Lennon

Inhaltsverzeichnis

- ⚙️
- ▼ 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Verweisungssyntax
- ▼ 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
- ▼ 3 Grundlegende Datenstrukturen
 - ▼ 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können verändert werden
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutablen Objekte
 - 3.3 Ausgabe mit `print`
 - ▼ 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`$\inf`)
 - 3.4.5 Not-a-Number (`NaN`)
 - ▼ 3.5 Mathematische Operationen
 - 3.5.1 Arithmetische Operationen
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
 - 3.6 Umwandlung des Datentyps
 - ▼ 3.7 Rundung
 - 3.7.1 Runden mit `round`

```
In [95]: for gast in gaesteliste:  
    print(gast['name'], gast.get('bemerkung', ''))
```

Frank Farian
Lady Gaga Supercool!
John Lennon

```
In [96]: gast_2['bemerkung'] = 'Supercool!'  
print(gaesteliste)
```

[{'name': 'Frank Farian'}, {'name': 'Lady Gaga', 'bemerkung': 'Supercool!'}
{'name': 'John Lennon'}]

Inhaltsverzeichnis



- ▼ 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Verweisungen
- ▼ 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
- ▼ 3 Grundlegende Datenstrukturen
 - ▼ 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können verändert werden
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutablen Objekte
 - 3.3 Ausgabe mit `print`
- ▼ 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`$\inf`)
 - 3.4.5 Not-a-Number (`NaN`)
- ▼ 3.5 Mathematische Operationen
 - 3.5.1 Arithmetische Operationen
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
- 3.6 Umwandlung des Datentyps
- ▼ 3.7 Rundung
 - 3.7.1 Runden mit `round`

3.12.5 Sets (Mengen)

Inhaltsverzeichnis



- ▼ 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Verweisungssyntax
- ▼ 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
- ▼ 3 Grundlegende Datenstrukturen
 - ▼ 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können verändert werden
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutablen Objekte
 - 3.3 Ausgabe mit `print`
 - ▼ 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`$\inf`)
 - 3.4.5 Not-a-Number (`NaN`)
 - ▼ 3.5 Mathematische Operationen
 - 3.5.1 Arithmetische Operationen
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
 - 3.6 Umwandlung des Datentyps
 - ▼ 3.7 Rundung
 - 3.7.1 Runden mit `round`

```
In [97]: a = set(['rot', 'gruen', 'blau', 'gelb'])  
print(a)
```

```
{'gelb', 'rot', 'blau', 'gruen'}
```

3.12.5 Sets (Mengen)

Inhaltsverzeichnis



- ▼ 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Verweisungssyntax
- ▼ 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
- ▼ 3 Grundlegende Datenstrukturen
 - 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können mehrfach zuweisen
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutablen Objekte
 - 3.3 Ausgabe mit `print`
 - 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`inf`)
 - 3.4.5 Not-a-Number (`NaN`)
 - 3.5 Mathematische Operationen
 - 3.5.1 Arithmetische Operationen
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
 - 3.6 Umwandlung des Datentyps
 - 3.7 Rundung
 - 3.7.1 Runden mit `round`

```
In [97]: a = set(['rot', 'gruen', 'blau', 'gelb'])  
print(a)
```

```
{'gelb', 'rot', 'blau', 'gruen'}
```

```
In [98]: a = 'Dies ist eine Zeichenkette.'  
# Nun schauen wir, welche Buchstaben hierin vorkommen.  
zeichenvorrat = set(a)  
print(zeichenvorrat)
```

```
{' ', '.', 'i', 'h', 'e', 'z', 'k', 's', 'n', 'c', 'D', 't'}
```

3.12.6 Named Tuples (nicht klausurrelevant)

Inhaltsverzeichnis

☞ *	
▼ 1	Syntaktische Konventionen
1.1	Keine Zeilenummern
1.2	Groß-/Kleinschreibung
1.3	Reservierte Namen
1.4	Zuweisungs- und Verweisungssyntax
▼ 2	Stil und Formatierung
2.1	Namen
2.2	Leerzeichen
▼ 3	Grundlegende Datenstrukturen
▼ 3.1	Namen und Objekte
3.1.1	Alles in Python ist ein Objekt
3.1.2	Objekte können verändert werden
3.1.3	Mehrfachzuweisung
3.2	Mutable und Immutablen Objekte
3.3	Ausgabe mit <code>print</code>
▼ 3.4	Numerische Werte
3.4.1	Ganze Zahlen
3.4.2	Gleitkommazahl
3.4.3	Dezimalzahlen
3.4.4	Unendlich (<code>\$inf</code>)
3.4.5	Not-a-Number (<code>NaN</code>)
▼ 3.5	Mathematische Operationen
3.5.1	Arithmetische Operationen
3.5.2	Potenz
3.5.3	Wurzel
3.5.4	Ganzzahlige Division
3.5.5	Divisionsrest (modulo)
3.6	Umwandlung des Datentyps
▼ 3.7	Rundung
3.7.1	Runden mit <code>round()</code>

- Nicht Gegenstand dieser Vorlesung
- Python Reference: Named Tuples

3.13 Benutzereingabe mit `input()`

Inhaltsverzeichnis

- ⚙️
- ▼ 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Verweisungsschlüsselwörter
- ▼ 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
- ▼ 3 Grundlegende Datenstrukturen
 - ▼ 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können verändert werden
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutablen Objekte
 - 3.3 Ausgabe mit `print`
- ▼ 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`$\inf`)
 - 3.4.5 Not-a-Number (`NaN`)
- ▼ 3.5 Mathematische Operationen
 - 3.5.1 Arithmetische Operationen
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
- 3.6 Umwandlung des Datentyps
- ▼ 3.7 Rundung
 - 3.7.1 Runden mit `round`

In [58]:

```
# Benutzereingabe mit input([text])
# Ergebnis ist Zeichenkette (s.u.)
eingabe = input('Ihr Name? ')
```

Ihr Name? Hepp

3.14 Typumwandlung (Type Cast)

3.14.1 Zeichenkette in Ganzzahl (int)

```
In [100]: zahl_als_text = "7"  
          zahl_als_int = int(zahl_als_text)
```

Inhaltsverzeichnis

- ⚙️
- ▼ 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Ver-
- ▼ 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
- ▼ 3 Grundlegende Datenstrukturen
 - ▼ 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können mehrfach zuweisen
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutab
 - 3.3 Ausgabe mit `print`
- ▼ 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`$\inf`)
 - 3.4.5 Not-a-Number (`NaN`)
- ▼ 3.5 Mathematische Operationen
 - 3.5.1 Arithmetische Operat
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
- 3.6 Umwandlung des Datentyps
- ▼ 3.7 Rundung
 - 3.7.1 Runden mit `round`

Inhaltsverzeichnis

- ⌚ 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Verweisungssyntax
- ⌚ 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
- ⌚ 3 Grundlegende Datenstrukturen
 - ⌚ 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können verändert werden
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutablen Objekte
 - 3.3 Ausgabe mit `print`
- ⌚ 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`$\inf`)
 - 3.4.5 Not-a-Number (`NaN`)
- ⌚ 3.5 Mathematische Operationen
 - 3.5.1 Arithmetische Operationen
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
- ⌚ 3.6 Umwandlung des Datentyps
- ⌚ 3.7 Rundung
 - 3.7.1 Runden mit `round()`

3.14 Typumwandlung (Type Cast)

3.14.1 Zeichenkette in Ganzzahl (int)

```
In [100]: zahl_als_text = "7"  
          zahl_als_int = int(zahl_als_text)
```

3.14.2 Zeichenkette als Gleitkommazahl (float)

```
In [101]: float_als_text = "3.1415"  
          float_als_zahl = float(float_als_text)
```

3.14.3 Zahl als Zeichenkette (String)

Inhaltsverzeichnis

- ⌚ ⓘ
 - ▼ 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Verweisungssyntax
 - ▼ 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
 - ▼ 3 Grundlegende Datenstrukturen
 - ▼ 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können verändert werden
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutablen Objekte
 - 3.3 Ausgabe mit `print`
 - ▼ 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`$\inf`)
 - 3.4.5 Not-a-Number (`NaN`)
 - ▼ 3.5 Mathematische Operationen
 - 3.5.1 Arithmetische Operationen
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
 - 3.6 Umwandlung des Datentyps
 - ▼ 3.7 Rundung
 - 3.7.1 Runden mit `round`

```
In [102]: zahl_als_text = str(7)
           float_als_text = (str(3.1415))
```

3.14.4 Umwandlung einer Zahl in eine Zeichenkette

Inhaltsverzeichnis

- ⌚⚙️
- ▼ 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Verweisungssyntax
- ▼ 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
- ▼ 3 Grundlegende Datenstrukturen
 - ▼ 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können mehrfach zuweisen
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutablen Objekte
 - 3.3 Ausgabe mit `print`
- ▼ 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`$\inf`)
 - 3.4.5 Not-a-Number (`NaN`)
- ▼ 3.5 Mathematische Operationen
 - 3.5.1 Arithmetische Operationen
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
- 3.6 Umwandlung des Datentyps
- ▼ 3.7 Rundung
 - 3.7.1 Runden mit `round`

In [121]:

```
a = 42
a_string = str(a)

# Was ist hier der Unterschied?
print(a * 2)
print(a_string * 2)
```

84

4242

4 Übungsaufgaben

Inhaltsverzeichnis

- ⌚ ⓘ ▾ 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Ver-
- ⌚ ⓘ ▾ 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
- ⌚ ⓘ ▾ 3 Grundlegende Datenstrukturen
 - 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können verändert werden
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutab - 3.3 Ausgabe mit `print`
 - 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`$\inf`)
 - 3.4.5 Not-a-Number (`NaN`)
 - 3.5 Mathematische Operationen
 - 3.5.1 Arithmetische Operatoren
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
 - 3.6 Umwandlung des Datentyps
 - 3.7 Rundung
 - 3.7.1 Runden mit `round()`

Siehe separate Notebooks auf der [Seite zur Veranstaltung](#).

6 Quellenangaben und weiterführende Literatur

Inhaltsverzeichnis

- ⌚ ⓘ ▾ 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Verweisungssyntax
- ⌚ ⓘ ▾ 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
- ⌚ ⓘ ▾ 3 Grundlegende Datenstrukturen
 - ⌚ ⓘ ▾ 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können verändert werden
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutablen Objekte
 - 3.3 Ausgabe mit `print`
- ⌚ ⓘ ▾ 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`$\inf`)
 - 3.4.5 Not-a-Number (`NaN`)
- ⌚ ⓘ ▾ 3.5 Mathematische Operationen
 - 3.5.1 Arithmetische Operationen
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
- ⌚ ⓘ ▾ 3.6 Umwandlung des Datentyps
- ⌚ ⓘ ▾ 3.7 Rundung
 - 3.7.1 Runden mit `round`

[Pyt2019] Python Software Foundation. Python 3.8.0 Documentation. <https://docs.python.org/3/>.

Vielen Dank!

Inhaltsverzeichnis

- ⌚⚙️
- ▼ 1 Syntaktische Konventionen
 - 1.1 Keine Zeilenummern
 - 1.2 Groß-/Kleinschreibung
 - 1.3 Reservierte Namen
 - 1.4 Zuweisungs- und Verweisungsschlüsselwörter
- ▼ 2 Stil und Formatierung
 - 2.1 Namen
 - 2.2 Leerzeichen
- ▼ 3 Grundlegende Datenstrukturen
 - ▼ 3.1 Namen und Objekte
 - 3.1.1 Alles in Python ist ein Objekt
 - 3.1.2 Objekte können verändert werden
 - 3.1.3 Mehrfachzuweisung
 - 3.2 Mutable und Immutablen Objekte
 - 3.3 Ausgabe mit `print`
- ▼ 3.4 Numerische Werte
 - 3.4.1 Ganze Zahlen
 - 3.4.2 Gleitkommazahl
 - 3.4.3 Dezimalzahlen
 - 3.4.4 Unendlich (`$\inf`)
 - 3.4.5 Not-a-Number (`NaN`)
- ▼ 3.5 Mathematische Operationen
 - 3.5.1 Arithmetische Operationen
 - 3.5.2 Potenz
 - 3.5.3 Wurzel
 - 3.5.4 Ganzzahlige Division
 - 3.5.5 Divisionsrest (modulo)
- 3.6 Umwandlung des Datentyps
- ▼ 3.7 Rundung
 - 3.7.1 Runden mit `round()`

<http://www.ebusiness-unibw.org/wiki/Teaching/PIP>

Martin Hepp, martin.hepp@unibw.de