

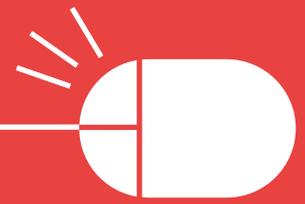
Diese **Leseprobe** haben Sie von



[Das Buch erhalten Sie hier](#)

3

Die ersten Programme



3.1 Der Python-Editor

🚩 Projekt: Der Begrüßungsautomat

3.2 Ein Programm im Dateimanager starten

3.3 Strings – Experimente in der IDLE-Shell

3.4 Das EVA-Prinzip

3.5 Zahlen eingeben

3.6 So oder so? Programmverzweigung

3.7 Zeilen und Blöcke – Layout eines Python-Programms

3.8 Noch einmal bitte! Wiederholung

🚩 Projekt: Zahlenraten

3.9 Wahr oder nicht wahr: Logische Aussagen

3.10 Logische Operatoren

3.11 Schwierige Entscheidungen

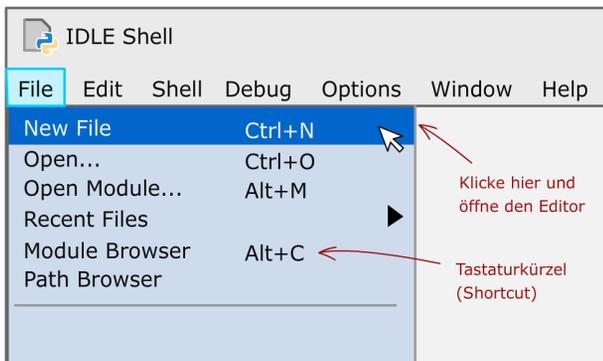
3.12 Datum und Zeit

Bisher hast du nur einzelne Befehle in der Python-Shell ausprobiert. In diesem Kapitel verwenden wir den Editor der Programmierumgebung *IDLE*, um Programme mit mehreren Anweisungen zu schreiben.

3.1 Der Python-Editor

Den Editor starten

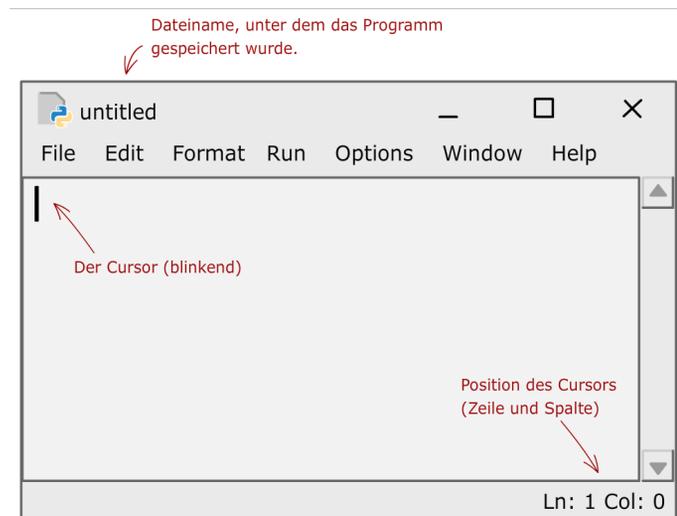
Ein Editor ist – ganz allgemein – eine Software, mit der man Texte oder Bilder bearbeiten kann. IDLE enthält einen speziellen Editor für Python-Programme. Starte IDLE und klicke dann im Menü **File** (Datei) auf den Befehl **New File** (Neue Datei). Dann öffnet sich ein neues Editor-Fenster. Oben im Rahmen trägt es den Titel `untitled` (ohne Titel).



Das Editor-Fenster

Im Editor-Fenster von IDLE kannst du einen Programmtext eingeben.

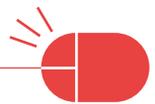
Es ist ein Texteditor mit einigen Besonderheiten, die die Programmierung erleichtern. Wie in der Python-Shell werden Textpassagen je nach Bedeutung automatisch in unterschiedlichen Farben angezeigt. Das nennt man auch *Syntax-Highlighting* (sprich: Süntax-Hailaiting). Zum Beispiel sind die Namen von Standardfunktionen lila.



Shortcuts

Menüs mit Befehlen zum Anklicken sind ja schön und gut, aber die Handhabung kostet Zeit. Für Editor-Befehle, die du häufig verwendest, solltest du dir Tastenkürzel merken.

Taste	Wirkung
<code>[Strg]+[N]</code>	Neues Editor-Fenster
<code>[Strg]+[S]</code>	Speichern
<code>[F5]</code>	Programm starten
<code>[Alt]+[G]</code>	Gehe zu Zeilennummer ...
<code>[Strg]+[C]</code>	Kopieren
<code>[Strg]+[V]</code>	Einfügen



Projekt Der Begrüßungsautomat

Die Idee

Wenn Menschen sich treffen, begrüßen sie sich. Programme einen Begrüßungsautomat. Der Benutzer wird nach seinem Namen gefragt. Dann wird er freundlich begrüßt.



Programmierung

1 Programmtext eingeben

- ➔ Starte die Programmierumgebung IDLE.
- ➔ Wähle den Menübefehl **File|New File** oder drücke das Tastenkürzel `[Strg] + [N]`.
- ➔ Schreibe in das Editorfenster das Python-Programm.

In dem Begrüßungsprogramm spielen Texte eine wichtige Rolle. Ein Text wird in der Programmierung *Zeichenkette* oder *String* (engl. für Kette) genannt. Ein String steht immer in Anführungszeichen, z.B. 'Wie heißt du? '.

```

untitled
File Edit Format Run Options

print('Wie heißt du?')
name = input('Name: ')
gruß = 'Hallo ' + name + '!'
print(gruß)
  
```

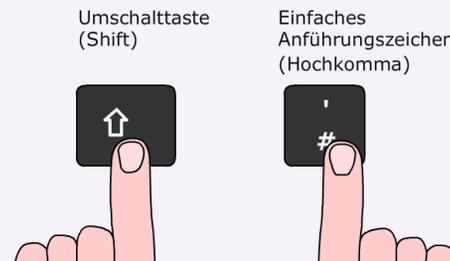
Standardfunktionen sind lila.

Strings sind grün.

PRAXISTIPP

Anführungszeichen

Das einfache Anführungszeichen ' findest du auf der Taste mit dem Hashtag-Zeichen #. Verwechsle es nicht mit einem Apostroph `.



PRAXISTIPP

Achte auf die Textfarbe!

Das Syntax-Highlighting hilft dir, Schreibfehler zu erkennen. Zum Beispiel sind Strings im IDLE-Editor grün.

Wenn du bei einem String wie 'Hallo' das zweite Anführungszeichen vergessen hast, fällt dir das so leicht auf.

```
gruß = 'Hallo + name + '!'
```

Hier wurde das zweite Anführungszeichen vergessen.

Hoppla, das sollte eigentlich nicht grün sein.

2 Speichern

Speichere dein Programm in deinem Projektordner unter dem Namen `gruss.py` ab. Das geht so:

- Wähle im Menü **File** den Befehl **save as** (Speichern unter).
- Gehe in deinen Projektordner.
- Gib den Dateinamen `gruss.py` ein. Achte darauf, dass du die Datei-Endung `.py` verwendest.

Übrigens: Im Menü findest du auch die Shortcuts der Befehle (Tastenkürzel).

2: Nach dem ersten Speichern steht hier der Dateiname.

gruss.py		
File	Edit	Format Run Options Window
New File		Ctrl+N ← Shortcut Strg+N
Open...		Ctrl+O
Open Module...		Alt+M
Recent Files		
Module Browser		Alt+C
Path Browser		
<hr/>		
Save		Ctrl+S
Save As...		Ctrl+Shift+S
Save Copy As		Alt+Shift+S

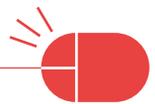
3: Wenn das Programm schon einen Dateinamen hat, speicherst du mit diesem Befehl.

1: Zum Speichern beim ersten Mal hier klicken!

Warum soll ich denn die Datei `gruss.py` nennen? Warum schreibe ich nicht `gruß.py`?

In Dateinamen verwendest du besser kein `ß` und keine Umlaute wie `ä`, `ö`, `ü`. Denn manche Betriebssysteme kommen damit nicht zurecht.

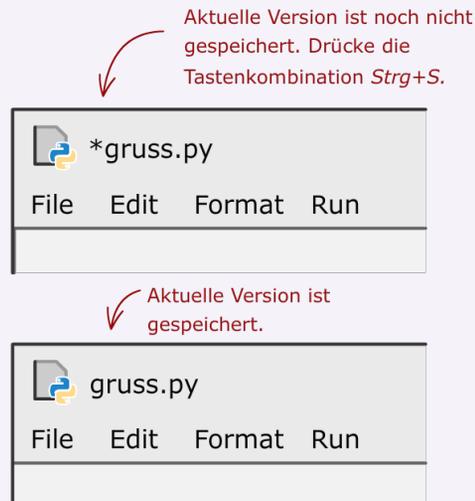
Und dein Programm soll doch auf allen Computern laufen, oder?



PRAXISTIPP

Das Geheimnis des Sternchens

Während du dein Programm entwickelst, solltest du dein Projekt immer wieder zwischenspeichern. Das geht am schnellsten, indem du die Tastenkombination `[Strg]+[S]` drückst. Achte auf den Dateinamen am oberen Rand des Editor-Fensters! Wenn vor dem Dateinamen ein Sternchen steht, ist die aktuelle Version deines Programms noch nicht gespeichert.



3 Das Programm mit IDLE starten

So startest du dein Programm: Klicke im Menü **Run** (Ausführen) auf den Befehl **Run module** (Programm-Modul ausführen) oder drücke die Taste `[F5]`.

Es öffnet sich die IDLE-Shell mit der Meldung `RESTART` und das Programm läuft. Du kannst mit dem Computer kommunizieren.

```
>>> = RESTART:
      Wie heißt du?
      Name: Tina
      Hallo Tina!
>>>
```

Wenn die Ausführung des Programms beendet ist, erscheint wieder der Prompt der IDLE-Shell.

So funktioniert es

```
print('Wie heißt du?')
```

Der Computer gibt diesen Text aus.

Wie heißt du?

```
name = input('Name: ')
```

Die Funktion `input()` sorgt dafür, dass dieser Text als Prompt auf dem Bildschirm erscheint. Dann wartet der Computer auf eine Eingabe.

Wie heißt du?
Name:

Hier steht der Cursor. Dahinter kann Tina ihren Namen eingeben.

```
name = input('Name: ')
```

Der Computer speichert die Zeichen, die Tina eingibt, in der Variablen `name`.

Tina

name

Wie heißt du?
Name: Tina

Eingabe über die Tastatur

```
gruß = 'Hallo ' + name + '!'
```

Der Computer setzt einen Text aus drei Stücken zusammen und speichert ihn in der Variablen `gruß`.

Halo

Tina

!!!

gruß

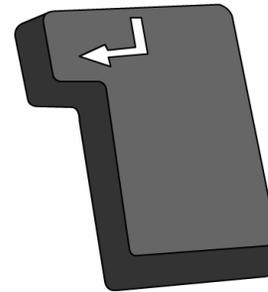
```
print(gruß)
```

Der Computer gibt den Inhalt der Variablen `gruß` aus.

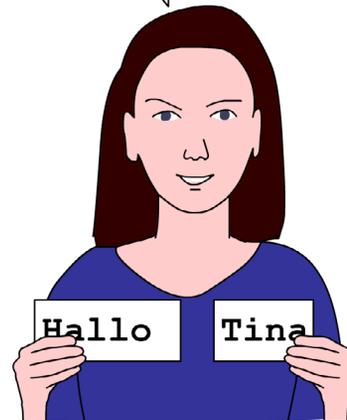
Programmlauf beendet!

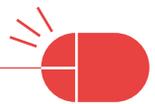
Wie heißt du?
Name: Tina
Hallo Tina!
>>>

Bei `input()` wartet der Computer, bis die ENTER-Taste gedrückt worden ist.



Mit `+` kann man Strings verbinden.





3.2 Ein Programm im Dateimanager starten

Das Programm-Icon

Wenn du deinen Programmtext gespeichert hast, erscheint in deinem Projektordner ein Programm-Icon.



Schließen des Konsolenfensters verhindern

Es gibt nur ein Problem: Sobald der Computer die letzte `print()`-Anweisung ausgeführt hat, ist das Programm beendet und das Konsolenfenster schließt sich wieder. Um das zu verhindern, fügst du noch eine `input()`-Anweisung hinzu. Die Funktion `input()` wartet immer, bis die Taste  gedrückt worden ist.

```
print('Wie heißt du? ')
name = input('Name: ')
gruß = 'Hallo ' + name + '!'
print(gruß)
input()
```

Hier wartet der Computer, bis die ENTER-Taste gedrückt worden ist.
Erst dann ist das Programm beendet.

Start durch Anklicken

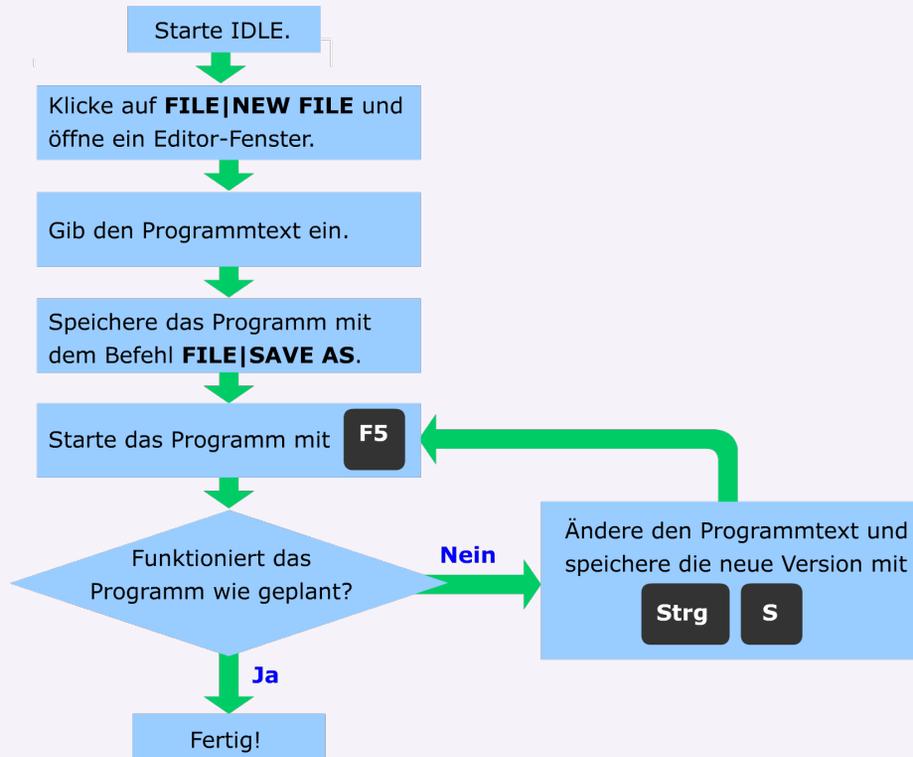
Wenn du das Programm-Icon anklickst (bei Windows musst du doppelklicken), startet das Programm. Es öffnet sich dann *nicht* die IDLE-Shell, sondern ein Konsolenfenster des Betriebssystems. Es hat normalerweise einen schwarzen Hintergrund. Wie in der Python-Shell kannst du jetzt etwas eingeben.

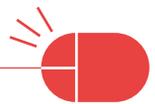


Konsolenfenster

PRAXISTIPP

Auf einen Blick: Ein Python-Programm entwickeln





3.3 Strings – Experimente in der IDLE-Shell

Wie kann man einen String aufschreiben?

Ein Gruß, wie z.B. »Hallo, wie gehts!« ist ein Text. Technisch besteht ein Text aus Zeichen, d.h. Buchstaben, Leerzeichen, Satzzeichen usw. Eine solche Zeichenkette nennt man auch String. Bei Python schreibst du einen String in einfache oder doppelte Anführungszeichen. Beides ist erlaubt. Du darfst aber nicht unterschiedliche Anführungszeichen vorne und hinten verwenden.

Lange Strings

Ein normaler String muss in eine Zeile geschrieben werden. Es gibt aber auch sogenannte *lange Strings*, die über mehrere Zeilen gehen können. Sie werden vorne und hinten jeweils mit *drei einfachen* Anführungszeichen aufgeschrieben.

```
>>> print(''' Dies ist ein Text
        mit zwei Zeilen.''' )
        Dies ist ein Text
        mit zwei Zeilen.
>>>
```

An dieser Stelle ist ein
Zeilenumbruch im String.

Strings verbinden (Konkatenation)

Mit dem Plusoperator + können Zeichenketten verbunden werden. Das nennt man auch *Verkettung* oder *Konkatenation*.

```
>>> "Hallo"
'Hallo'
>>> print("Hallo")
Hallo
>>> print('Hallo')
Hallo
>>> print("Hallo')
SyntaxError
>>> print('Er sagte "Hallo".')
Er sagte "Hallo".
```

Der Computer stellt Strings immer nur mit einfachen Anführungszeichen dar.

Wenn ein String mit `print()` ausgegeben wird, fehlen die Anführungszeichen.

Einfache Anführungszeichen funktionieren auch und sind sogar sehr beliebt.

Unterschiedliche Anführungszeichen vorne und hinten sind verboten.

Doppelte Anführungszeichen in einem String sind erlaubt, wenn er außen einfache Anführungszeichen hat.

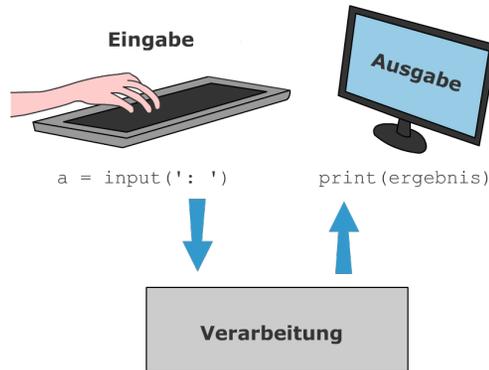
```
>>> 'Hallo ' + 'Leute!'
'Hallo Leute!'
>>> 2 + '-mal'
```

Leerzeichen

Fehler! Eine Zahl und einen String kann man nicht aneinanderhängen.

3.4 Das EVA-Prinzip

Viele Programme (wie auch der Begrüßungsautomat) folgen dem EVA-Prinzip. Die drei Buchstaben stehen für *Eingabe*, *Verarbeitung* und *Ausgabe*. Zuerst gibt ein Mensch Daten ein. Dann verarbeitet der Computer die Daten und berechnet ein Ergebnis. Schließlich wird das Ergebnis ausgegeben. Ein solches Programm nennt man auch *interaktiv*.



CHALLENGE 1

Glückwunschkarte

Wandle das Programm des Begrüßungsautomaten ab und schreibe ein Programm, das Name und Alter einer Person einliest und dann einen Geburtstagsgruß ausgibt.

```
Name: Melissa
Alter: 17
Hallo Melissa,
alles Gute zu deinem 17. Geburtstag!
```

Eingaben der Benutzerin

3.5 Zahlen eingeben

Ganze Zahlen und Kommazahlen

Zahlen sind etwas anderes als Strings. Mit Zahlen kann man rechnen, mit Strings nicht. Bei Python gibt es unterschiedliche Typen von Zahlen. Besonders wichtig sind ganze Zahlen wie 1 (Typ: `int`) und Dezimalbrüche wie 1.23 (Typ: `float`).

Dezimalbrüche nennt man im Deutschen auch Kommazahlen oder Gleitkommazahlen, weil sie in der deutschen Schreibweise ein Komma enthalten. Bei Python verwendest du jedoch einen Punkt statt des Kommas.

int

Eine ganze Zahl enthält *keinen* Punkt, ...

... darf nicht mit einer 0 beginnen, ...

... kann beliebig lang sein.

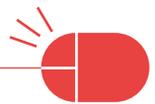
float

Eine Gleitkommazahl muss einen Punkt enthalten, ...

... hat eine *begrenzte* Anzahl von Ziffern nach dem Punkt, ...

... kann auch in wissenschaftlicher Schreibweise angegeben werden.

Das Diagramm zeigt Beispiele für `int` und `float` Typen. Für `int` sind die Zahlen 123 und 01 dargestellt. Ein roter Balken über dem 01 zeigt an, dass es nicht mit einer 0 beginnen darf. Ein langer Streifen mit den Ziffern 12345678901660023418 zeigt, dass die Länge beliebig sein kann. Für `float` sind die Zahlen 1.2, 0.3333333333333333 und 5.1e+16 dargestellt. Ein roter Balken über dem 0.3333333333333333 zeigt an, dass die Anzahl von Ziffern nach dem Punkt begrenzt ist. Ein roter Pfeil weist auf 5.1e+16 hin, was die wissenschaftliche Schreibweise für 5100000000000000.0 (17 Stellen vor dem Punkt) darstellt.



Datentyp feststellen

Mit der Funktion `type()` kannst du feststellen, zu welchem Typ eine Zahl oder ein anderes Objekt gehört.

Hier gibt es auch einen Unterschied zur Mathematik. In der Mathematik sind die Zahlen 1 und 1.0 gleich. In der Python-Programmierung gehören diese beiden Zahlen zu unterschiedlichen Typen.

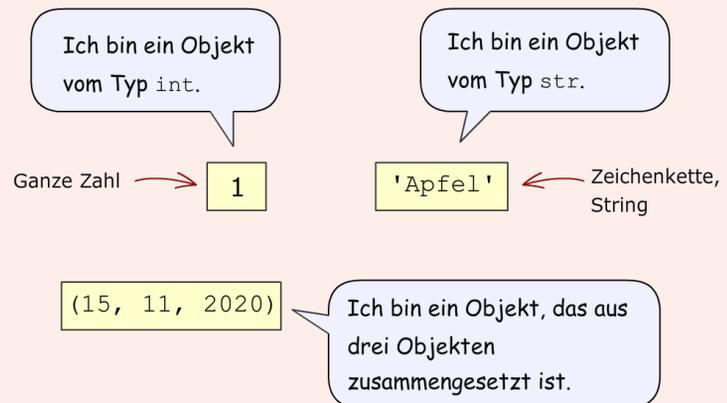
IDLE-Shell

```
>>> type(1)
<class 'int'>
>>> type(1.0)
<class 'float'>
>>> type('1')
<class 'str'>
```

So bezeichnet Python den Typ `int` (ganze Zahl).
 Typ `float` (Gleitkommazahl).
 Typ `str` (String).

? WAS IST EIN OBJEKT?

Die Daten (Werte), die in einem Python-Programm verarbeitet werden, nennt man *Objekte*. Es gibt unterschiedliche *Typen* von Objekten, z.B. ganze Zahlen (Typ `int`) oder Texte in Anführungszeichen, sogenannte Strings oder Zeichenketten (Typ `str`).



Beispiel: Flächenberechnung

Der erste Versuch

Wir werden nun ein Programm nach dem EVA-Prinzip entwickeln, das die Fläche eines Rechtecks berechnet. Kennst du die Formel? Sie lautet: Fläche gleich Länge mal Breite.

Gib im IDLE-Editorfenster das Programm ein und speichere es mit dem Befehl **File|Save as**.

```
a = input('Länge: ')
b = input('Breite: ')
fläche = a * b
print('Fläche:', fläche)
```

Eingabe
Verarbeitung
(kritische Stelle)
Ausgabe

Fehler!

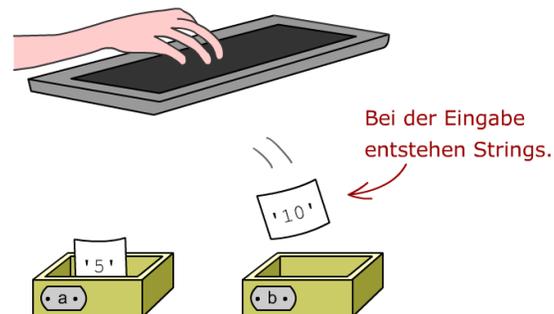
Starte das Programm mit dem Befehl **Run|Run Module**. Gib nacheinander zwei Zahlen ein. Nachdem du die zweite Zahl eingegeben und  gedrückt hast, sollte eigentlich das Rechenergebnis kommen. Stattdessen gibt es eine Fehlermeldung.

```
Prompt      Eingaben des Benutzers
↓
Länge: 5
Breite: 10
Traceback ...
... line 3 ...
    fläche = a*b
TypeError ...
```

Traceback bedeutet Rückverfolgung. Python sucht die Fehlerquelle. Fehler ist in Zeile 3 aufgetreten. Diese Anweisung hat den Fehler verursacht. Typ-Fehler

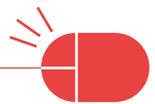
Das Problem

Die Funktion `input()` liefert immer einen String, wenn der Benutzer über die Tastatur etwas eingibt. Das gilt auch dann, wenn Ziffern eingegeben worden sind. Für Python gibt es einen Unterschied zwischen dem String 10 und der Zahl 10. Man kann nur Zahlen multiplizieren und keine Strings.



```
fläche = a*b
```

Strings kann man nicht multiplizieren.

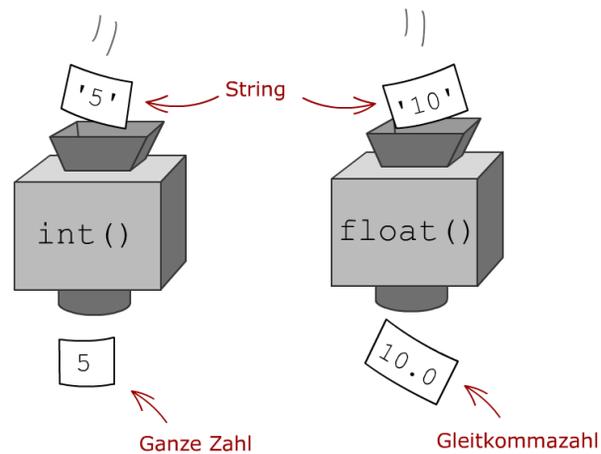


Die Lösung: Zahlen aus Strings gewinnen

Mit der Funktion `int()` kannst du aus einem String, der eine ganze Zahl als Text darstellt, eine ganze Zahl gewinnen, mit der man dann auch rechnen kann. Mit `float()` kannst du aus einem String eine Gleitkommazahl gewinnen.

Du kannst das einmal in der Python-Shell ausprobieren:

```
>>> int('5')
5
```



Das verbesserte Programm – der zweite Versuch

```
a = input('Länge: ')
b = input('Breite: ')
länge = float(a)
breite = float(b)
fläche = länge * breite
print('Fläche:', fläche)
```

Aus den eingegebenen Strings werden Gleitkommazahlen gewonnen.

Die Multiplikation funktioniert jetzt.

Ausgabe

```
Länge: 5
Breite: 10
Fläche: 50.0
```

Das Ergebnis ist auch eine Gleitkommazahl.

Nach diesem Muster kannst du alle möglichen Berechnungsprogramme schreiben.



Diese **Leseprobe** haben Sie von



Das Buch erhalten Sie hier