Michael Weigend

So einfach!

Programmieren lernen mit Python

2. Auflage

- Spielend lernen anhand von anschaulichen Bildern
- Mit einfachen Schritt-für-Schritt-Anleitungen
- Für Kinder und Erwachsene – ab 10 Jahre

mitp

Inhalt

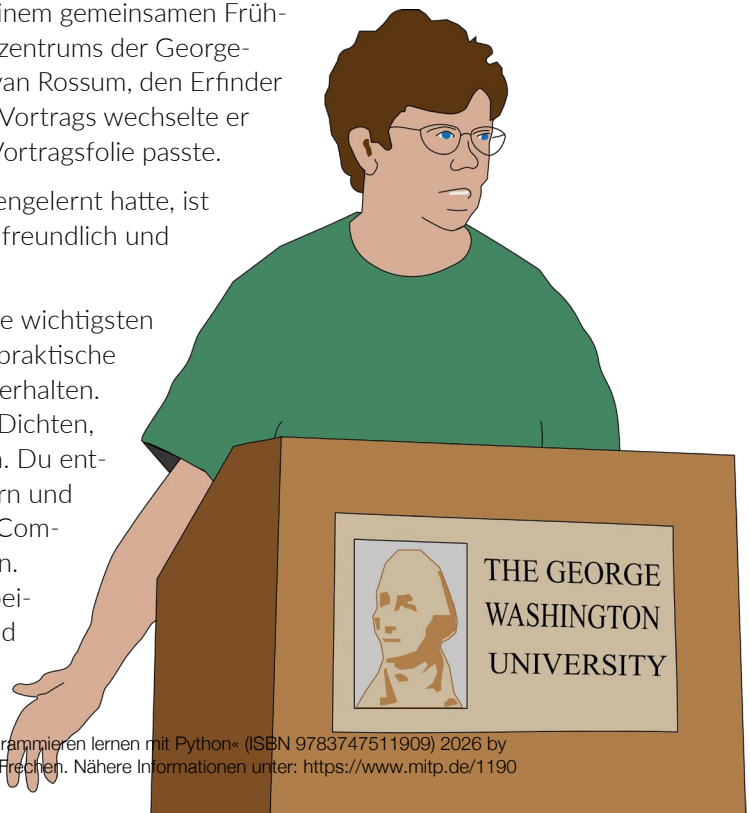
Einleitung

Python ist eine Programmiersprache, die sehr leicht zu erlernen ist und die sich deshalb perfekt für deinen Einstieg in die Programmierung eignet.

Ich selbst habe Python im Jahr 2000 kennengelernt. Damals war ich an der Technischen Universität Dortmund und betreute ein Team, das E-Learning-Material zur Einführung in die Informatik entwickelte. Ein Kapitel widmete sich dem Thema »Objektorientierte Programmierung« mit der bis dahin ziemlich unbekannten Programmiersprache Python. Als ich die ersten Python-Skripte sah, war ich sofort begeistert. Kurze Texte, deren Bedeutung man rasch erfassen konnte. Ideal zum Lernen. Zwei Jahre später schrieb ich für den mitp-Verlag mein erstes Python-Buch. Im Frühjahr 2003 besuchte ich zum ersten Mal die PyCon, die internationale Konferenz der Python-Community, in Washington DC (USA). Hier trafen sich Wissenschaftlerinnen und Wissenschaftler aus der ganzen Welt und berichteten in Vorträgen, wie sie Python in ihren Fachgebieten einsetzten. Die Konferenztage begannen immer mit einem gemeinsamen Frühstück an großen Tischen im Ballsaal des Konferenzzentrums der George-Washington-Universität. Hier traf ich auch Guido van Rossum, den Erfinder der Programmiersprache Python. Während seines Vortrags wechselte er mehrmals sein T-Shirt, damit die Farbe immer zur Vortragsfolie passte.

So wie die Menschen, die ich auf der PyCon kennengelernt hatte, ist auch die Programmiersprache selbst: Sie ist offen, freundlich und lädt zum Mitmachen ein.

Freue dich auf spannende Projekte, in denen du die wichtigsten Konzepte von Python kennenlernst. Du schreibst praktische Rechenprogramme, die sich mit dem Benutzer unterhalten. Du gestaltest Spiele und digitale Assistenten zum Dichten, Briefeschreiben und zum Bestimmen von Pflanzen. Du entwickelst Benutzungsoberflächen mit Eingabefeldern und Schaltflächen. In kreativen Projekten lässt du den Computer Bilder zeichnen und Geschenkpapier drucken. Du lernst, wie man Daten aus dem Internet verarbeiten kann und wie man eine App zum Speichern und Verwalten von Notizen entwickelt.



Die Programmbeispiele sind kurz und können von dir weiter ausgebaut werden. Alle wichtigen Codezeilen werden durch Texte und Bilder erklärt.

Du kannst die Programmbeispiele von der Website des mitp-Verlags herunterladen. Der URL ist:

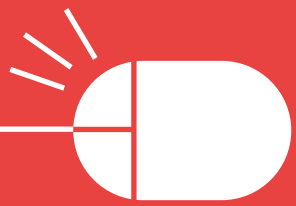
https://www.mitp.de/1190

Wähle die Registerkarte **Downloads** und klicke auf den Link **Programmbeispiele**. Nach dem Entpacken hast du für jedes Kapitel ein Verzeichnis mit den Programmen, die im Buch erklärt werden.

Viel Spaß mit Python!

Michael Weigend

3 Die ersten Programme

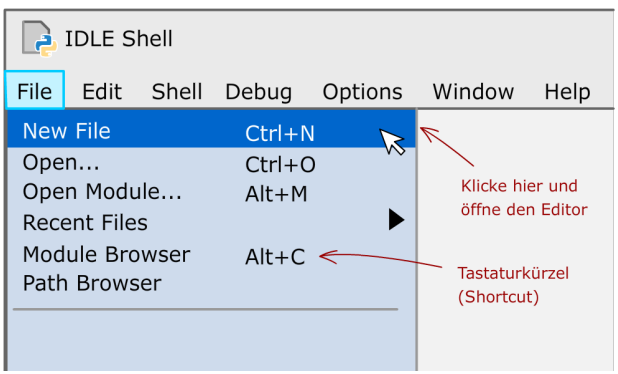


Bisher hast du nur einzelne Befehle in der Python-Shell ausprobiert. In diesem Kapitel verwenden wir den Editor der Programmierumgebung *IDLE*, um Programme mit mehreren Anweisungen zu schreiben.

3.1 Der Python-Editor

Den Editor starten

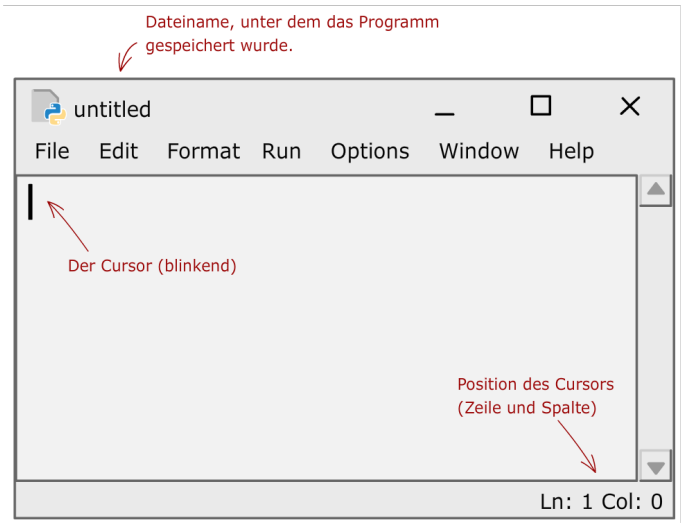
Ein Editor ist – ganz allgemein – eine Software, mit der man Texte oder Bilder bearbeiten kann. IDLE enthält einen speziellen Editor für Python-Programme. Starte IDLE und klicke dann im Menü **File** (Datei) auf den Befehl **New File** (Neue Datei). Dann öffnet sich ein neues Editor-Fenster. Oben im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

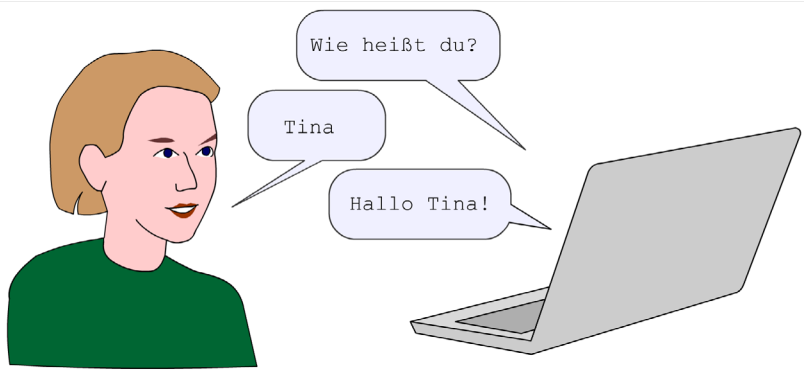
Menüs mit Befehlen zum Anklicken sind ja schön und gut, aber die Handhabung kostet Zeit. Für Editor-Befehle, die du häufig verwendest, solltest du dir Tastenkürzel merken.

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

Projekt Der Begrüßungsautomat

Die Idee

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

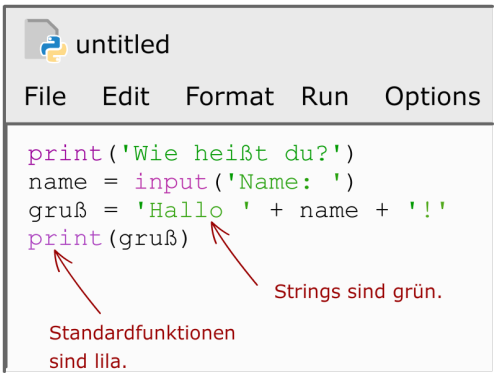


Programmierung

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.

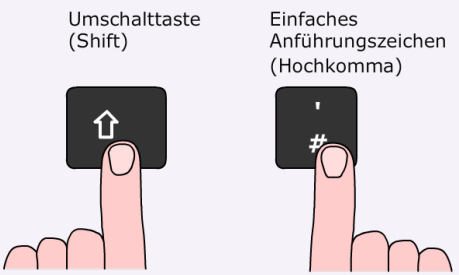
In dem Begrüßungsprogramm spielen Texte eine wichtige Rolle. Ein Text wird in der Programmierung *Zeichenkette* oder *String* (engl. für Kette) genannt. Ein String steht immer in Anführungszeichen, z.B. `'Wie heißt du? '`.



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.

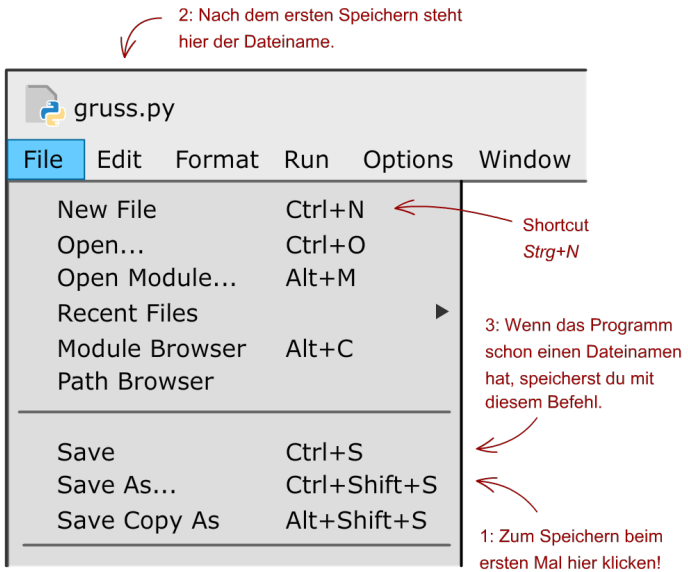
Hoppla, das sollte eigentlich nicht grün sein.

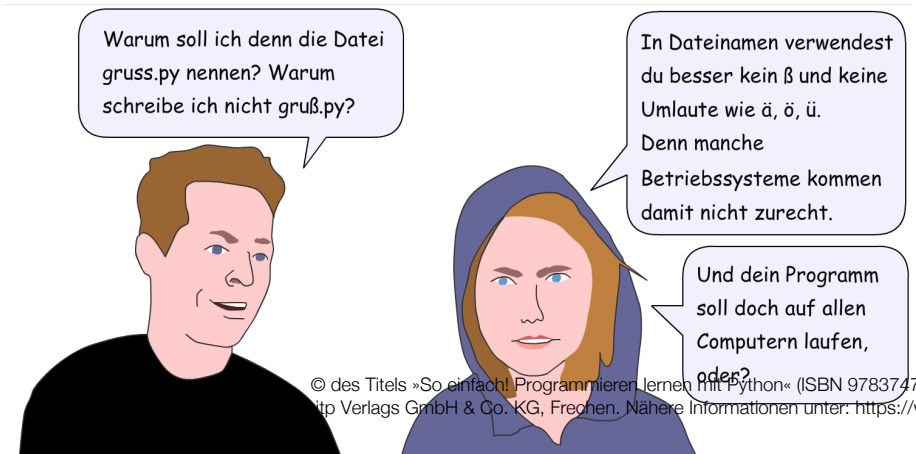
2 Speichern

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).

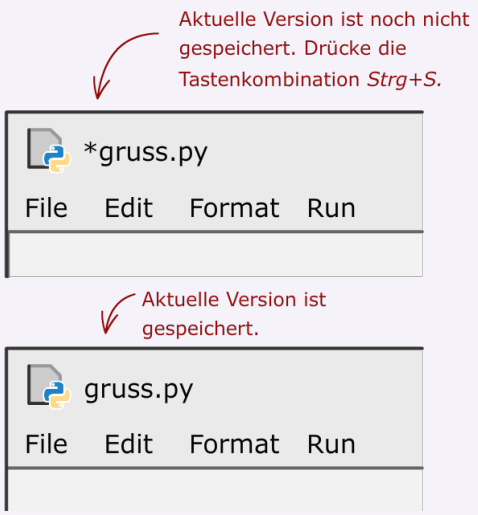




PRAXISTIPP

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: ')
```

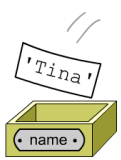
Die Funktion `input()` sorgt dafür , dass dieser Text als Prompt auf dem Bildschirm erscheint.
Dann wartet der Computer auf eine Eingabe.

```
Wie heißt du?
Name:
```

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

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

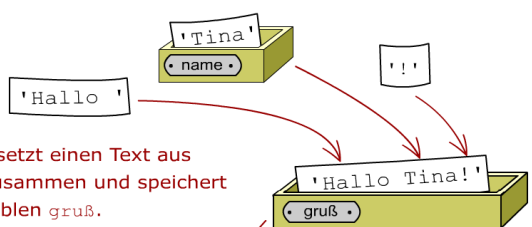
Der Computer speichert die Zeichen, die Tina eingibt, in der Variablen `name`.




```
Wie heißt du?
Name: Tina
```

Eingabe über die Tastatur

```
gruß = 'Hallo ' + name + '!'
```



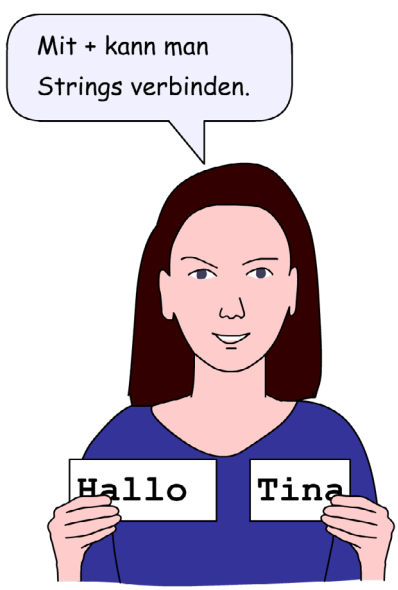
Der Computer setzt einen Text aus drei Stücken zusammen und speichert ihn in der Variablen `gruß`.

```
print(gruß)
```

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

Programmlauf beendet!

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



3.2 Ein Programm im Dateimanager starten

Das Programm-Icon

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

Anklicken und
Programm starten

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

Schließen des Konsolenfensters verhindern

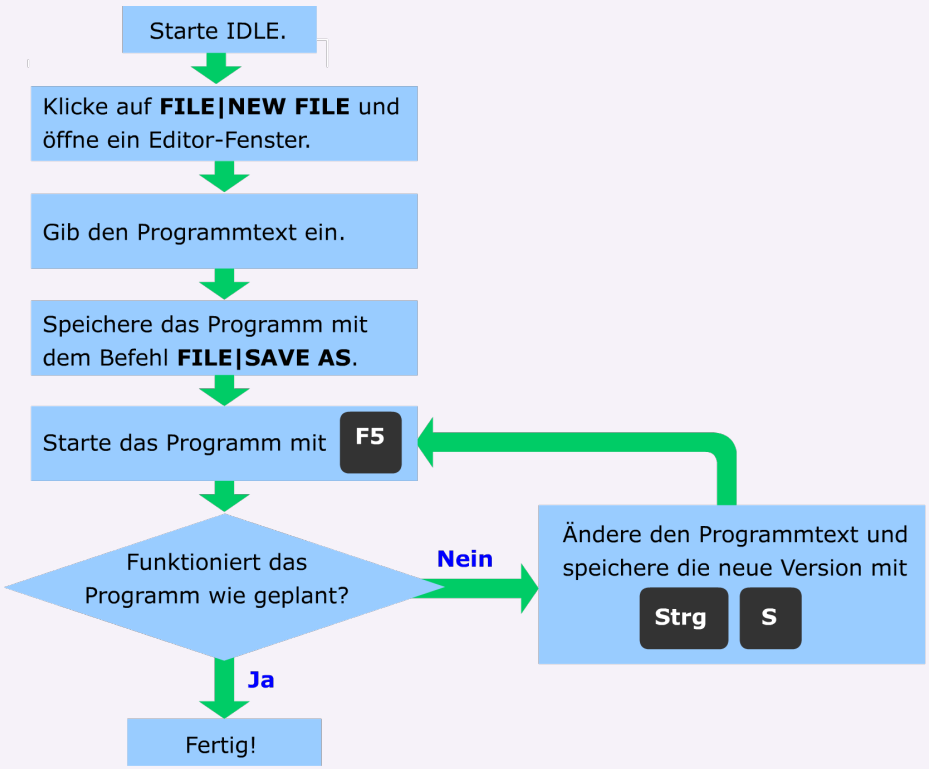
Es gibt nur ein Problem: Sobald der Computer die letzte `print()`-Anweisung ausgeführt hat, ist das Programm beendet und das Konsolenfenster schließt sich wieder. Um das zu verhindern, fügst du noch eine `input()`-Anweisung hinzu. Die Funktion `input()` wartet immer, bis die Taste ↵ gedrückt worden ist.

```
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.

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.

```
>>> "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.

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.

Stings verbinden (Konkatenation)

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

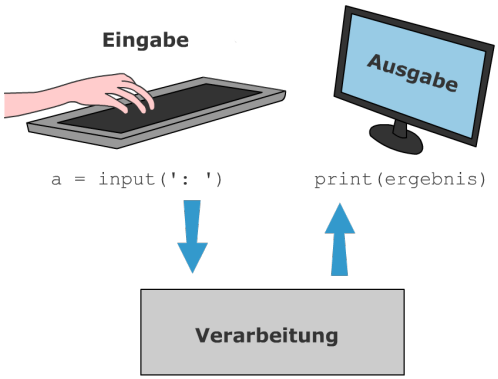
Leerzeichen

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

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!
```

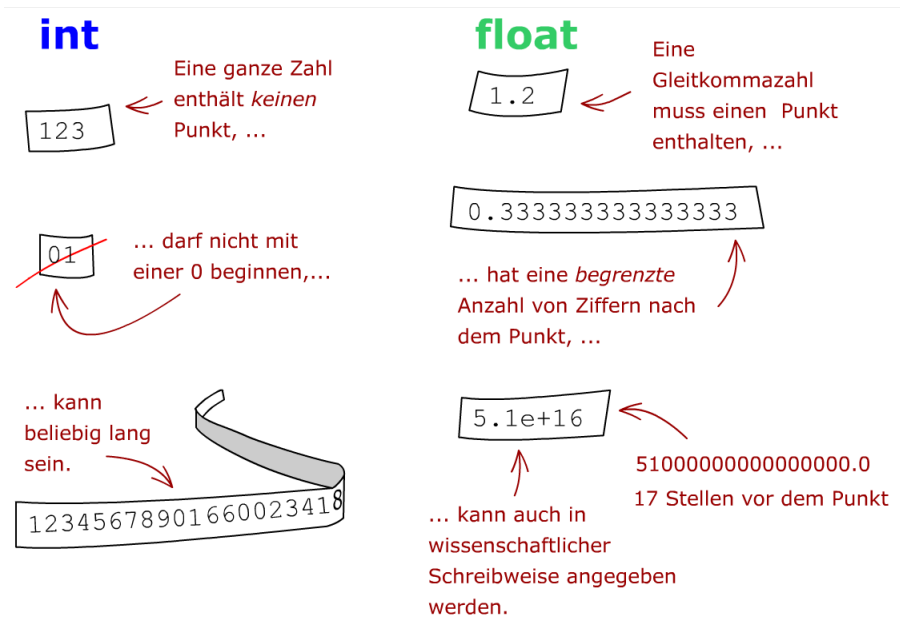
Eingaben der Benutzerin

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`).

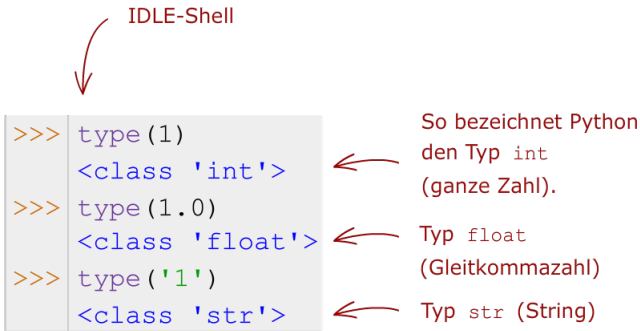
Dezimalbrüche nennt man im Deutschen auch Kommazahlen oder Gleitkommazahlen, weil sie in der deutschen Schreibweise ein Komma enthalten. Bei Python verwendest du jedoch einen Punkt statt des Kommas.



Datentyp feststellen

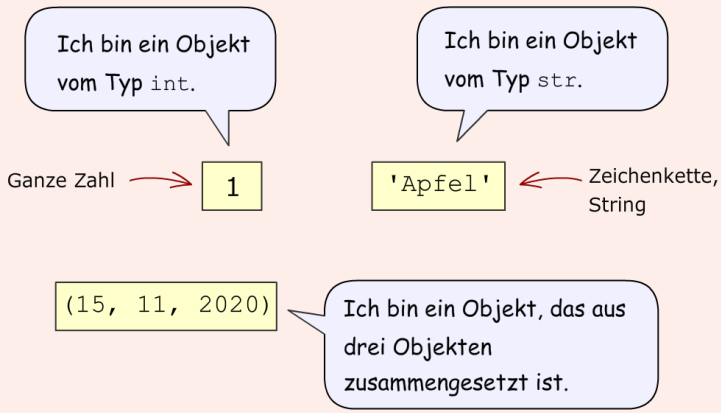
Mit der Funktion `type()` kannst du feststellen, zu welchem Typ eine Zahl oder ein anderes Objekt gehört.

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.



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)
```

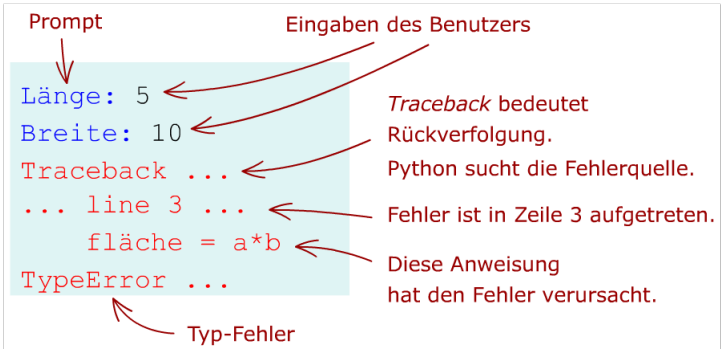
Eingabe

Verarbeitung (kritische Stelle)

Ausgabe

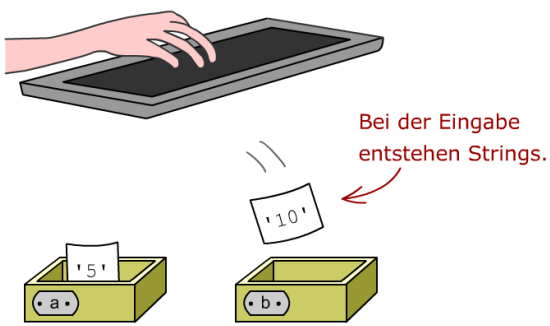
Fehler!

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.



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
```

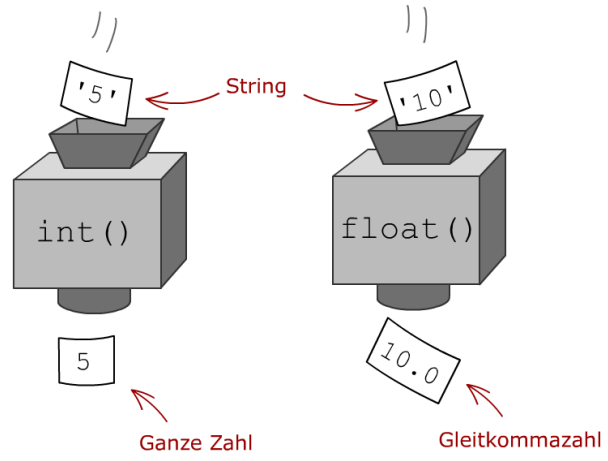
Strings kann man nicht multiplizieren.

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.

Stichwortverzeichnis

W

Z