

Michael Weigend



So einfach!

Programmieren lernen mit **Python**



Spielend lernen anhand von anschaulichen Bildern

Mit einfachen Schritt-für-Schritt-Anleitungen

Für Kinder und Erwachsene – ab 10 Jahre

Inhalt

Einleitung	8
1 Willkommen in der Welt der Programmierung	10
1.1 Computerprogramme: Schlaue Texte für dumme Computer	12
1.2 Wahrscheinlich kannst du schon programmieren	16
1.3 Lösungen zu den Aufgaben im Text	22
2 Einstieg in Python	24
2.1 Was ist Python?	26
2.2 Python als Taschenrechner – die IDLE-Shell	28
2.3 Funktionen aufrufen	33
2.4 Module	34
2.5 Variablen	38
2.6 Der Python-Interpreter	41
3 Die ersten Programme	42
3.1 Der Python-Editor	44
 Projekt: Der Begrüßungsautomat	45
3.2 Ein Programm im Dateimanager starten	49
3.3 Strings – Experimente in der IDLE-Shell	51
3.4 Das EVA-Prinzip	52
3.5 Zahlen eingeben	52
3.6 So oder so? Programmverzweigung	56

3.7 Zeilen und Blöcke – Layout eines Python-Programms.	59
3.8 Noch einmal bitte! Wiederholung	60
 Projekt: Zahlenraten.	64
3.9 Wahr oder nicht wahr: Logische Aussagen.	65
3.10 Logische Operatoren	66
3.11 Schwierige Entscheidungen	68
3.12 Datum und Zeit.	72

4 Neue Möglichkeiten mit Funktionen 74

4.1 Wie definiert man eine Funktion?	77
4.2 Funktionen können Werte zurückgeben	83
 Projekt: Volumenassistent – ein Programm mit Auswahlmenü.	86
 Projekt: Der Briefassistent – globale Variablen und Seiteneffekte	89
4.3 Bilder mit der Turtle-Grafik	92
4.4 Was ist Rekursion?	100
4.5 Rekursive Funktionen	101

5 Daten in Kollektionen zusammenfassen 104

5.1 Welche Arten von Kollektionen gibt es?	106
5.2 Suchen und aufzählen	110
5.3 Sequenzen verarbeiten	112
5.4 Listen können sich verändern.	114
 Projekt: Planeten aufzählen	117
 Projekt: Digitales Telefonbuch	119
5.5 Dateien öffnen, speichern und bearbeiten.	122
 Projekt: Eine Notiz schreiben und als Textdatei speichern.	125
5.6 Laufzeitfehler abfangen.	126



5.7	Kollektionen speichern und laden	128
	Projekt: Telefonnummern verwalten	129
	Projekt: Die Reimmaschine	131
	Projekt: Wie warm wird es? – Informationen aus dem Internet	134

6 Bunt und zum Anklicken: Grafische Benutzungsoberflächen **138**

	Projekt: Digitaler Würfel	141
6.1	Widgets konfigurieren	143
	Projekt: Farbmischer	147
	Projekt: Dichten mit Goethe	150
	Projekt: StarQuest – Bilder anzeigen	154
	Projekt: Digitaluhr	157

7 Der nächste Schritt: Die große Welt der Informatik **162**

7.1	Die Geschwindigkeit von Programmen	164
7.2	Objektorientierte Programmierung – eigene Klassen definieren	168
7.3	Agile Software-Entwicklung – arbeiten wie die Profis	171
	Projekt: Digitales Notizbuch – agil entwickelt	172
7.4	Webseiten programmieren mit JavaScript	180
7.5	Noch mehr Programmiersprachen	184
7.6	Programmierwettbewerbe	187

Lösungen **188**

Stichwortverzeichnis **205**

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 unbekannteren 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 kennlernst. 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:

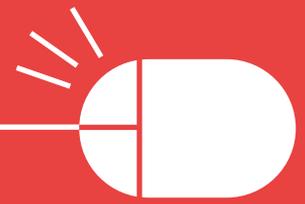
<https://www.mitp.de/0655>

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



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

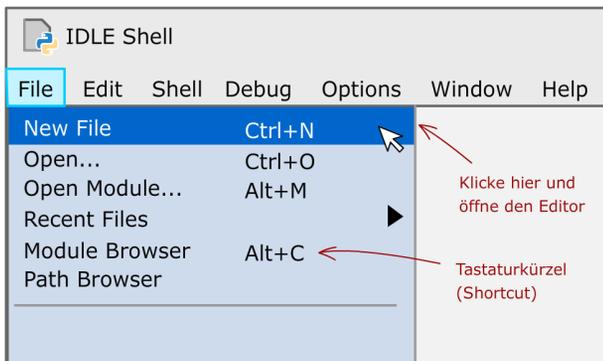
3.12 Datum und Zeit

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

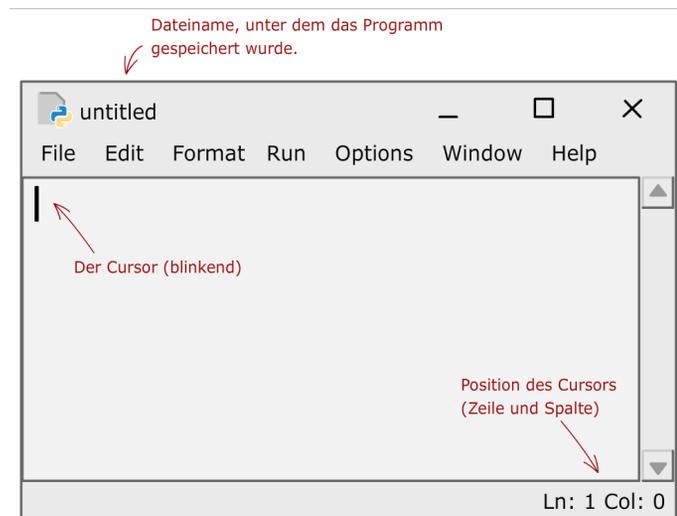
Ein Editor ist – ganz allgemein – eine Software, mit der man Texte oder Bilder bearbeiten kann. IDLE enthält einen speziellen Editor für Python-Programme. Starte IDLE und klicke dann im Menü **File** (Datei) auf den Befehl **New File** (Neue Datei). Dann öffnet sich ein neues Editor-Fenster. Oben im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



Das Editor-Fenster

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

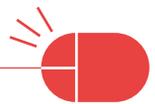
Es ist ein Texteditor mit einigen Besonderheiten, die die Programmierung erleichtern. Wie in der Python-Shell werden Textpassagen je nach Bedeutung automatisch in unterschiedlichen Farben angezeigt. Das nennt man auch *Syntax-Highlighting* (sprich: Süntax-Hailaiting). Zum Beispiel sind die Namen von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

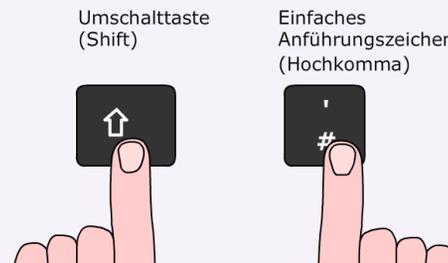
Standardfunktionen sind lila.

Strings sind grün.

PRAXISTIPP

Anführungszeichen

Das einfache Anführungszeichen ' findest du auf der Taste mit dem Hashtag-Zeichen #. Verwechsele 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.

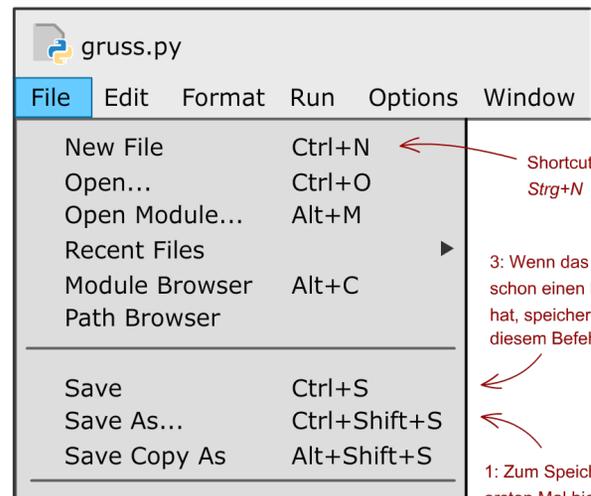
2 Speichern

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).

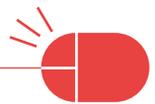
2: Nach dem ersten Speichern steht hier der Dateiname.



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.

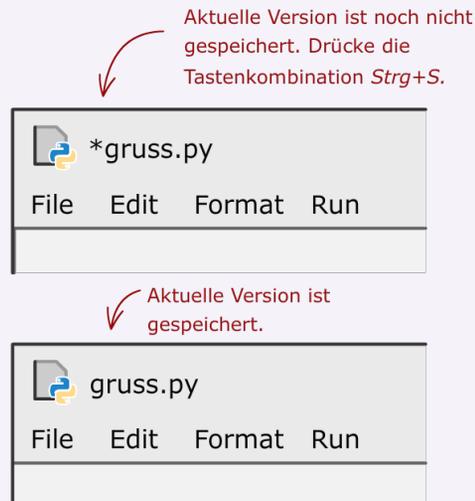
Und dein Programm soll doch auf allen Computern laufen, oder?



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.

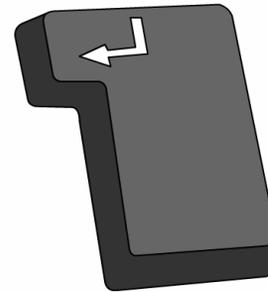
So funktioniert es

```
print('Wie heißt du?')
```

Der Computer gibt diesen Text aus.

Wie heißt du?

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



```
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.

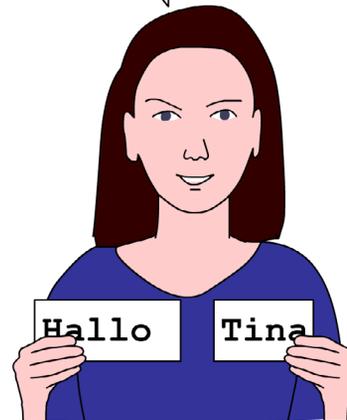
```
name = input('Name: ')
```

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

Wie heißt du?
Name: Tina

Eingabe über die Tastatur

Mit `+` kann man Strings verbinden.



```
gruß = 'Hallo ' + name + '!'
```

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

'Hallo ' 'Tina' '!'

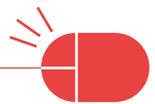
'Hallo Tina!'

```
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.



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

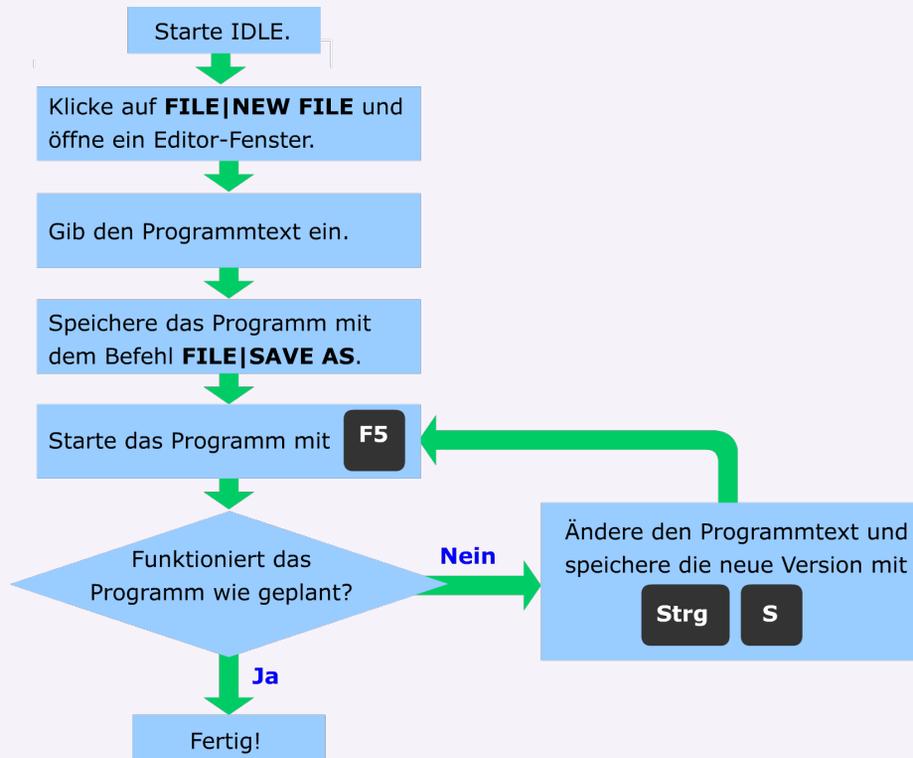
Wenn du das Programm-Icon anklickst (bei Windows musst du doppelklicken), startet das Programm. Es öffnet sich dann *nicht* die IDLE-Shell, sondern ein Konsolenfenster des Betriebssystems. Es hat normalerweise einen schwarzen Hintergrund. Wie in der Python-Shell kannst du jetzt etwas eingeben.

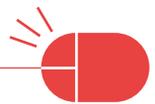


Konsolenfenster

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".')
```

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.

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

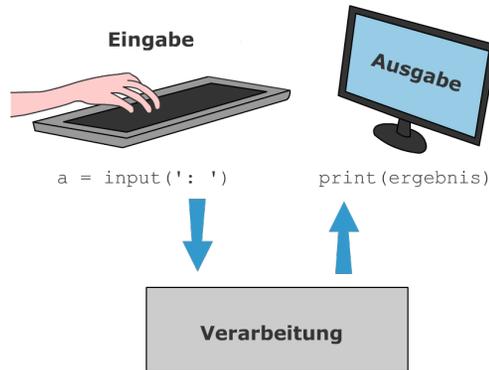
```
>>> 'Hallo ' + 'Leute!'
'Hallo Leute!'
>>> 2 + '-mal'
```

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!
```

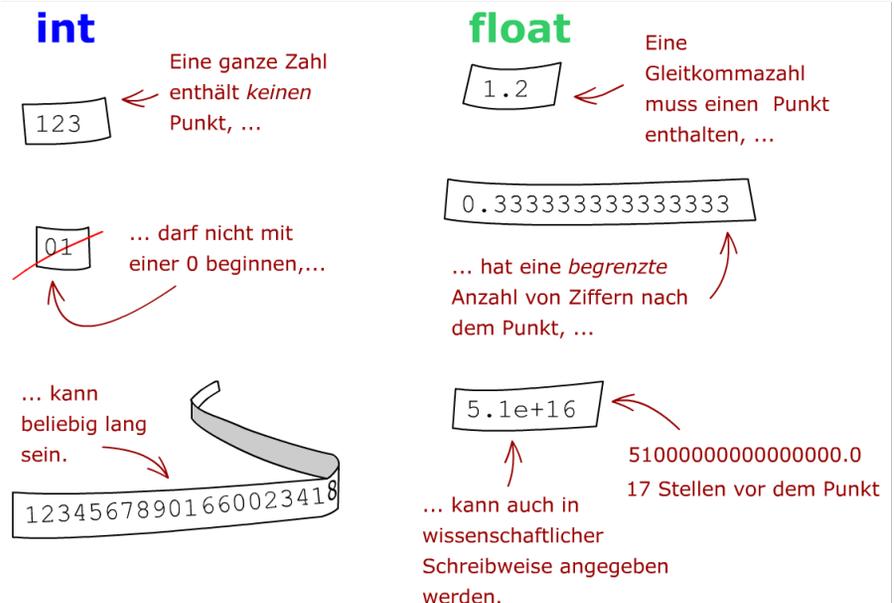
Eingaben der Benutzerin

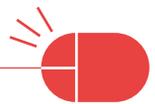
3.5 Zahlen eingeben

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.





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.

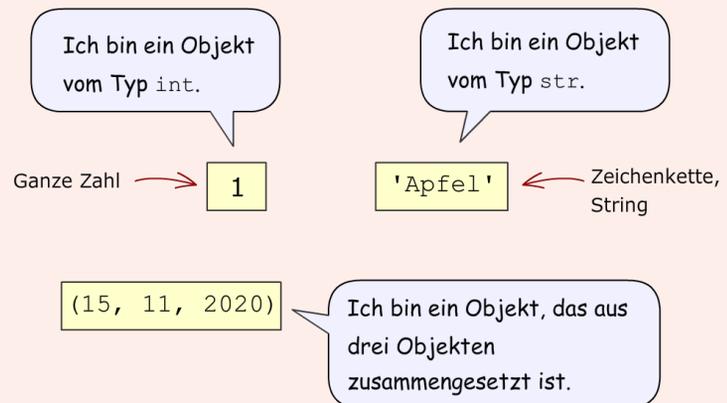
IDLE-Shell

```
>>> 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!

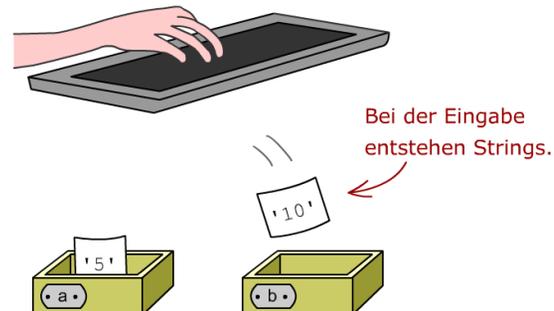
Starte das Programm mit dem Befehl **Run|Run Module**. Gib nacheinander zwei Zahlen ein. Nachdem du die zweite Zahl eingegeben und  gedrückt hast, sollte eigentlich das Rechenergebnis kommen. Stattdessen gibt es eine Fehlermeldung.

```
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

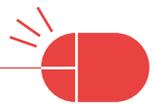
Das Problem

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.

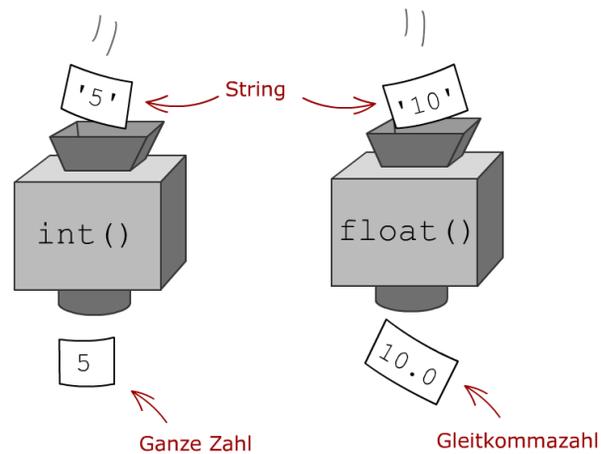


Die Lösung: Zahlen aus Strings gewinnen

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.



Stichwortverzeichnis

A

Abstand 146
after() 158
Agile Software-Entwicklung 171
Algorithmus 16, 17
analog 13
Ankerpunkt 152
Anwendungsfenster 140
App 12
Architektur 173
Ausrichtung 152

B

background 145
Baum 103
bd 145
Bedingung 19
Betriebssystem 14
bg 145
Bildersammlung 155
Binärdatei 128
Boole 67
borderwidth 145
Botschaft 170
Bremsweg 91
Briefassistent 89
Brunnen 84
Bubblesort 166
Bytestring 108

C

C++ 186
Calltip 78
Code 12
Code Playground 184
Codierung 123
count() 131

D

Datei öffnen 125
Datentyp 53
def 77
Dichten 150
digital 13
Digitaler Würfel 141
Digitales Telefonbuch 119
Digitalisierung 13
Digitaluhr 157
Dunkle Farben 149

E

Editor-Fenster 44
Eindeutig 17
Einrückung 59
Endlos-Wiederholung 63
endwith() 131
Entscheidungsbaum 68
Erdbeschleunigung 84
Ereignis 159
EVA-Prinzip 52, 149
Event 159

F

Fallunterscheidung 71
Farbe 147
Farbmischer 147
Farbtupel 96
fg 145
fillcolor() 94
Flächenberechnung 54
Flasche 169
font 145
for-Anweisung 19, 61
foreground 145
format() 157

Formatstring 157
Funktion 33, 76
Funktion aufrufen 79

G

Ganze Zahl 52
Ganzzahlige Division 31
Geschenkpapier 99
Geschwindigkeit 164
globale Variable 91
Glückwünsche 77
Glückwunschkarte 52
Goethe 150
Grafische Benutzeroberfläche 140
Grundstück 32
GUI 140
Guido van Rossum 27

H

Hardware 14
HDD 122
height 145
helle Farben 149
History 37
HTML 183

I

IDLE-Shell 28
if-Anweisung 19, 56
if-elif-else 71
if-else-Anweisung 57
image 145
importieren 34
Instanz 170
Iteration 110, 173

J

Java 186
JavaScript 182
justify 145, 152

K

Klasse 168
Kollektion 106, 121
Kommazahl 52
konfigurieren 144
Konkatenation 51

L

Laden 122
Langer String 51, 108
Laufzeitfehler abfangen 126
Layout 146
Liste 107
localtime() 72, 157
Login 56
Logische Aussage 65
lokale Variable 91
lower() 131

M

Mapping 165
Mathequiz 64
Modul 34
Modulo 31

N

Notiz 125
Notizbuch 172

O

Objekt 53
Objektorientierte Programmierung 168
Operator 30, 33



P

pack() 146
padx 145
pady 145
Papert 92
pickle 128
Planet 117
Potenzieren 32
Programmiersprache 26
Programmierwettbewerb 187
Projektmetapher 172
Python-Interpreter 41

Q

Quadratspirale 102

R

random 36
range() 61
Rechnen 38
Refactoring 178
Reimmaschine 131
Rekursion 100
Rekursionsabbruch 102
Rekursive Funktion 76, 101
relief 145
replace() 131, 132
return 83
Rosette 93
Ruby 186

S

Schleife 60
Seiteneffekt 89
Sequenz 18, 112
Shortcut 44
Sicherheitslücke 15
Smartphone 41
Software 14
Sortieren 165
Speichern 122

split() 131, 132
SSD 122
StarQuest 154
Story 172
Straight Selection 167
String 51, 107
String-Methoden 131
Suchen 110
Syntax 26

T

Taschenrechner 28
Tastenkombination 37
Telefonnummern verwalten 129
text 145
Textdatei 124
Text speichern 123
Text zerlegen 136
time 35
try-except 127
Tupel 107
Turtle-Grafik 98
type() 53

U

Uhrzeit 63
Unicode 122
Update 15
upper() 131
UTF-8 124

V

Variable 38
Verkettung 51
Vollbildmodus
159
Volumenassistent 86
Volumenberechnung 150

W

Wahrheitswert 67
WDR 134
Wertetabelle 62
Wetervorhersage 134
while-Anweisung 19, 62
Widget 143
width 145
Wiederholung 19, 60
wrap 151
Würfelmaschine 61

Z

Zahlenraten 64
Zeilenstruktur 59
Zeilenumbruch 151
Zeitmessung 164
Zeitobjekt 72
Zufallsbaum 103
Zufallssterne 99
Zuweisung 38