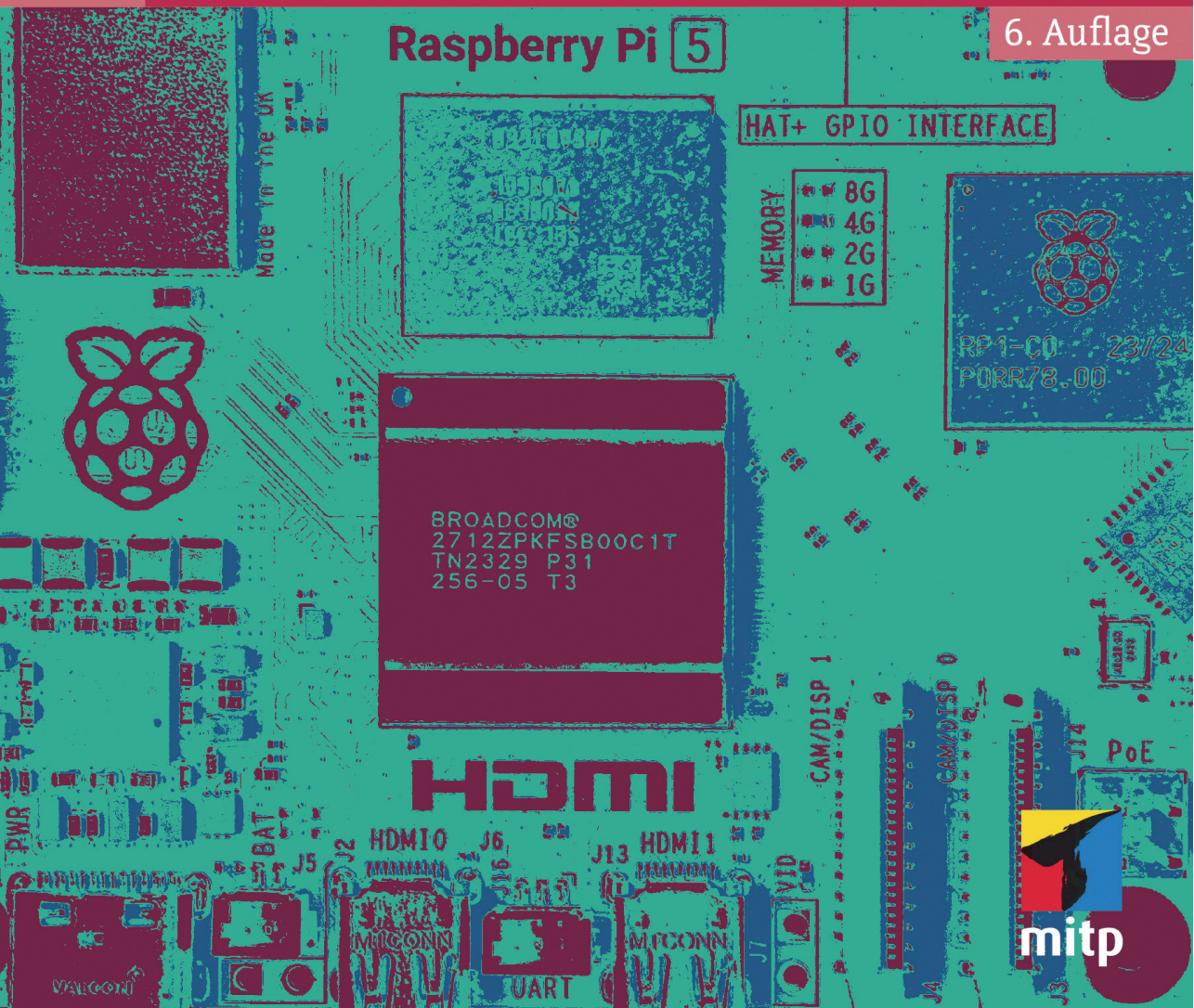


Michael Weigend

# Raspberry Pi programmieren mit Python

6. Auflage



# Inhaltsverzeichnis

	<b>Einleitung</b> . . . . .	13
	Lösungen und Programmcode zum Download . . . . .	16
<b>1</b>	<b>Begegnung mit Python</b> . . . . .	17
1.1	Was ist Python? . . . . .	17
1.2	IDLE . . . . .	18
1.2.1	IDLE-Shell . . . . .	20
1.2.2	Hotkeys . . . . .	21
1.3	Die IDLE-Shell als Taschenrechner . . . . .	22
1.3.1	Operatoren und Terme . . . . .	22
1.3.2	Zahlen . . . . .	24
1.3.3	Mathematische Funktionen . . . . .	28
1.4	Hilfe . . . . .	32
1.5	Namen und Zuweisungen . . . . .	33
1.5.1	Zuweisungen für mehrere Variablen . . . . .	35
1.5.2	Rechnen mit Variablen in der Shell . . . . .	36
1.5.3	Syntaxregeln für Bezeichner . . . . .	36
1.5.4	Neue Namen für Funktionen und andere Objekte . . . . .	37
1.5.5	Erweiterte Zuweisungen . . . . .	38
1.6	Mit Python-Befehlen Geräte steuern . . . . .	39
1.6.1	Projekt: Eine LED ein- und ausschalten . . . . .	39
1.6.2	Das Modul gpiozero . . . . .	41
1.6.3	Steuern mit Relais . . . . .	42
1.6.4	Projekt: Eine Taschenlampe an- und ausschalten . . . . .	44
1.6.5	Projekt: Steuerung eines LED-Strips mit MOSFETs . . . . .	46
1.7	Aufgaben . . . . .	49
<b>2</b>	<b>Python-Skripte</b> . . . . .	53
2.1	Ein Skript mit IDLE erstellen . . . . .	53
2.1.1	Ein neues Projekt starten . . . . .	53
2.1.2	Programmtext eingeben . . . . .	54
2.1.3	Das Skript ausführen . . . . .	55
2.1.4	Shortcuts . . . . .	55
2.2	Programme ausführen . . . . .	55
2.2.1	Programm in der Konsole starten . . . . .	55
2.2.2	Anklicken des Programmicons im File-Manager . . . . .	56
2.3	Interaktive Programme – das EVA-Prinzip . . . . .	59

2.3.1	Format mit Bedeutung – Aufbau eines Python- Programmtextes . . . . .	60
2.3.2	Eingabe – die input()-Funktion . . . . .	61
2.3.3	Verarbeitung – Umwandeln von Datentypen und Rechnen. . . . .	61
2.3.4	Ausgabe – die print()-Funktion . . . . .	62
2.4	Programmverzweigungen . . . . .	64
2.4.1	Einfache Bedingungen . . . . .	64
2.4.2	Wie erkennt man eine gute Melone? Zusammengesetzte Bedingungen. . . . .	66
2.4.3	Einseitige Verzweigungen und Programmblöcke . . . . .	67
2.4.4	Haben Sie Ihr Idealgewicht? . . . . .	68
2.4.5	Eine Besonderheit von Python: Wahrheitswerte für Objekte . . . . .	72
2.5	Bedingte Wiederholung – die while-Anweisung. . . . .	72
2.5.1	Projekt: Zahlenraten. . . . .	73
2.5.2	Have a break! Abbruch einer Schleife. . . . .	74
2.6	Projekte mit dem GPIO. . . . .	75
2.6.1	Blinklicht . . . . .	75
2.6.2	Schalter . . . . .	76
2.6.3	Zähler . . . . .	78
2.6.4	Die Methode wait_for_press() . . . . .	79
2.7	Projekt: Eine Alarmanlage. . . . .	79
2.7.1	Aufbau und Arbeitsweise der Alarmanlage . . . . .	79
2.7.2	Programmierung . . . . .	81
2.8	Aufgaben . . . . .	82
<b>3</b>	<b>Kollektionen: Mengen, Listen, Tupel und Dictionaries . . . . .</b>	<b>85</b>
3.1	Die Typhierarchie. . . . .	85
3.1.1	Reihenfolge der Elemente und Zugriff. . . . .	86
3.1.2	Änderbarkeit . . . . .	86
3.2	Gemeinsame Operationen für Kollektionen . . . . .	87
3.3	Kollektionen in Bedingungen. . . . .	88
3.3.1	Projekt: Kundenberatung. . . . .	89
3.3.2	Projekt: Sichere Kommunikation . . . . .	89
3.4	Iteration – die for-Anweisung. . . . .	90
3.4.1	Verwendung von break . . . . .	91
3.5	Sequenzen . . . . .	92
3.5.1	Konkatenation und Vervielfältigung . . . . .	92
3.5.2	Direkter Zugriff auf Elemente – Indizierung. . . . .	93
3.5.3	Slicing . . . . .	93
3.5.4	Projekt: Lesbare Zufallspasswörter . . . . .	94
3.6	Tupel. . . . .	96

3.7	Zeichenketten (Strings).....	97
3.7.1	Strings durch Bytestrings codieren.....	98
3.7.2	Formatieren .....	99
3.8	Listen .....	101
3.8.1	Listen sind Objekte und empfangen Botschaften .....	101
3.8.2	Klasse, Typ und Instanz.....	103
3.8.3	Kopie oder Alias? .....	103
3.8.4	Listenoperationen.....	104
3.8.5	Projekt: Zufallsnamen .....	106
3.8.6	Projekt: Telefonliste .....	107
3.8.7	Listen durch Comprehensions erzeugen .....	108
3.9	Zahlen in einer Folge – range()-Funktion .....	109
3.10	Projekt: Klopffzeichen .....	111
3.11	Mengen.....	114
3.12	Projekt: Zufallssounds .....	115
3.12.1	Wie kommen Töne aus dem Raspberry Pi?.....	115
3.12.2	Sounds mit PyGame .....	116
3.12.3	Programmierung .....	117
3.13	Dictionaries .....	119
3.13.1	Operationen für Dictionaries.....	120
3.13.2	Projekt: Morsen .....	121
3.14	Aufgaben .....	122
<b>4</b>	<b>Funktionen</b> .....	<b>125</b>
4.1	Aufruf von Funktionen .....	125
4.1.1	Unterschiedliche Anzahl von Argumenten .....	126
4.1.2	Positionsargumente und Schlüsselwort-Argumente.....	126
4.2	Definition von Funktionen .....	128
4.3	Funktionen in der IDLE-Shell testen .....	129
4.4	Docstrings .....	130
4.5	Veränderliche und unveränderliche Objekte als Parameter.....	131
4.6	Voreingestellte Parameterwerte .....	133
4.7	Beliebige Anzahl von Parametern .....	134
4.8	Die return-Anweisung unter der Lupe.....	135
4.9	Mehr Sicherheit! Vorbedingungen testen .....	136
4.10	Namensräume: Global und lokal .....	138
4.11	Rekursive Funktionen – die Hohe Schule der Algorithmik.....	141
4.11.1	Projekt: Rekursive Summe .....	141
4.11.2	Projekt: Quicksort.....	142
4.12	Funktionen per Knopfdruck aufrufen: Callback-Funktionen .....	143
4.12.1	Projekt: Klangmaschine.....	144
4.12.2	Projekt: Zeit schätzen.....	145
4.13	Aufgaben .....	147

<b>5</b>	<b>Fenster für den RPi – Grafische Benutzungsoberflächen</b> . . . . .	<b>151</b>
5.1	Wie macht man eine Benutzungsoberfläche? . . . . .	151
5.2	Projekt: Die digitale Lostrommel . . . . .	152
	5.2.1 Die Gestaltung der Widgets . . . . .	154
	5.2.2 Das Layout-Management . . . . .	156
5.3	Projekt: Farbmischer . . . . .	158
5.4	Wer die Wahl hat, hat die Qual: Checkbutton und Radiobutton . . . . .	160
5.5	Projekt: Automatische Urlaubsgrüße . . . . .	160
5.6	Projekt: Digitaler Glückskeks . . . . .	162
5.7	Viele Widgets schnell platziert: Das Grid-Layout . . . . .	164
5.8	Projekt: 100 Farben . . . . .	165
	5.8.1 Die professionelle Version . . . . .	167
5.9	Aufgaben . . . . .	170
<b>6</b>	<b>Daten finden, laden und speichern</b> . . . . .	<b>173</b>
6.1	Dateien . . . . .	173
	6.1.1 Daten speichern . . . . .	173
	6.1.2 Daten laden . . . . .	174
6.2	Ein Blick hinter die Kulissen: Die SD-Karte . . . . .	174
	6.2.1 Experiment 1: Wie viel Zeit wird zum Schreiben von 10 MB auf die SD-Karte benötigt? . . . . .	175
	6.2.2 Experiment 2: Wie viel Zeit wird zum Lesen von 100 MB von der SD-Karte benötigt? . . . . .	176
6.3	Datenstrukturen haltbar machen mit pickle . . . . .	176
6.4	Versuch und Irrtum – Mehr Zuverlässigkeit durch try-Anweisungen . . . . .	177
6.5	Projekt: Karteikasten . . . . .	178
	6.5.1 Der Editor . . . . .	179
	6.5.2 Der Presenter . . . . .	182
6.6	Benutzungsoberfläche zum Laden und Speichern . . . . .	185
	6.6.1 Dialogboxen . . . . .	185
	6.6.2 Erweiterung des Editors für Karteikarten . . . . .	187
	6.6.3 Erweiterung des Presenters . . . . .	190
6.7	Daten aus dem Internet . . . . .	192
6.8	Projekt: Goethe oder Schiller? . . . . .	194
	6.8.1 Methoden der String-Objekte . . . . .	194
	6.8.2 Programmierung . . . . .	196
6.9	Daten finden mit regulären Ausdrücken . . . . .	199
	6.9.1 Reguläre Ausdrücke . . . . .	199
	6.9.2 Die Funktion findall() . . . . .	201
	6.9.3 Projekt: Staumelder . . . . .	202
	6.9.4 Programmierung . . . . .	202
6.10	Aufgaben . . . . .	206

<b>7</b>	<b>Projekte mit Zeitfunktionen</b> . . . . .	<b>209</b>
7.1	Projekt: Fünf Sekunden stoppen und gewinnen. . . . .	209
7.2	Datum und Zeit im Überblick . . . . .	211
7.3	Projekt: Digitaluhr. . . . .	212
	7.3.1 Woher bekommt der RPi die Zeit? . . . . .	213
	7.3.2 Was ist ein Prozess? . . . . .	213
	7.3.3 Vollbildmodus . . . . .	215
	7.3.4 Event-Verarbeitung. . . . .	218
	7.3.5 Autostart . . . . .	219
7.4	Projekt: Ein digitaler Bilderrahmen . . . . .	219
	7.4.1 Zugriff auf das Dateisystem: Das Modul os . . . . .	220
	7.4.2 Python Imaging Library (PIL) . . . . .	221
	7.4.3 Die Programmierung . . . . .	223
7.5	Projekt: Wahrnehmungstest. . . . .	225
	7.5.1 Die Programmierung . . . . .	226
7.6	Aufgaben . . . . .	230
<b>8</b>	<b>Objektorientierte Programmierung</b> . . . . .	<b>233</b>
8.1	Überall Objekte . . . . .	233
8.2	Klassen und Vererbung bei Python . . . . .	235
	8.2.1 Einführendes Beispiel: Alphabet. . . . .	236
	8.2.2 Qualitätsmerkmal Änderbarkeit . . . . .	239
	8.2.3 Vererbung . . . . .	241
8.3	Projekt: Pong revisited . . . . .	242
	8.3.1 Die Klasse Canvas. . . . .	244
	8.3.2 Aufbau des Projekts . . . . .	249
	8.3.3 Die Tick-Metapher . . . . .	250
	8.3.4 Die Programmierung . . . . .	251
	8.3.5 Erweiterungen . . . . .	256
8.4	Aufgaben . . . . .	256
<b>9</b>	<b>Sensortechnik</b> . . . . .	<b>259</b>
9.1	Was ist ein digitaler Temperatursensor? . . . . .	259
9.2	Den DS1820 anschließen . . . . .	260
9.3	Temperaturdaten lesen . . . . .	261
	9.3.1 Temperaturdaten eines Sensors automatisch auswerten . . . . .	263
9.4	Projekt: Ein digitales Thermometer mit mehreren Sensoren . . . . .	264
	9.4.1 Ein Modul für die Messwerterfassung . . . . .	265
	9.4.2 Die grafische Oberfläche . . . . .	267
	9.4.3 Temperaturdaten per E-Mail senden . . . . .	268
9.5	Projekt: Ein Temperaturplotter. . . . .	270
	9.5.1 Temperatur-Zeitdiagramme . . . . .	270
	9.5.2 Programmierung . . . . .	271

9.6	Spannung messen . . . . .	275
9.6.1	Das SPI-Protokoll . . . . .	277
9.6.2	Programmierung . . . . .	278
9.7	Alkoholsensor. . . . .	280
9.7.1	Projekt: Achtung! Alkoholisches Getränk! . . . . .	281
9.7.2	Projekt: Den Alkoholgehalt der Luft und von Flüssigkeiten messen . . . . .	284
9.7.3	Wie kann man den Alkoholgehalt von Flüssigkeiten messen? . . . . .	285
9.7.4	Messen und interpolieren . . . . .	288
9.8	Projekte mit einem digitalen Lichtsensor . . . . .	291
9.8.1	Ein Paket in einer virtuellen Umgebung installieren . . . . .	292
9.8.2	Die Programmierung . . . . .	293
9.8.3	Das Modul smbus . . . . .	294
9.8.4	Mit dem Raspberry Pi Farben messen – Absorptionsspektrometer . . . . .	298
9.9	Kohlendioxid-Sensor . . . . .	304
9.9.1	Projekt: Datenlogger . . . . .	305
9.9.2	Das Sensormodul kalibrieren . . . . .	306
9.9.3	Projekt: Ein Experiment zur Diffusion von Gasen. . . . .	307
9.10	Mit Ultraschall Entfernungen messen . . . . .	310
9.10.1	Die Schaltung . . . . .	311
9.10.2	Projekt: Abstandsmessungen . . . . .	312
9.10.3	Projekt: Fische – eine interaktive Animation mit Ultraschall-Sensor. . . . .	313
9.11	Aufgaben . . . . .	318
9.12	Lösung des Rätsels. . . . .	320
<b>10</b>	<b>Projekte mit der Kamera . . . . .</b>	<b>321</b>
10.1	Das Kameramodul anschließen . . . . .	321
10.2	Die Kamerasoftware. . . . .	323
10.3	Projekt: Überwachungskamera – Livebild auf dem Bildschirm. . . . .	324
10.4	Projekt: Bewegung erfassen . . . . .	326
10.5	Projekt: Gerichtete Bewegungen erfassen . . . . .	330
10.5.1	Files verarbeiten mit subprocess und io . . . . .	331
10.5.2	Die Programmierung . . . . .	331
10.6	Projekt: Birnen oder Tomaten? . . . . .	337
10.6.1	Magische Methoden – das Überladen von Operatoren . . . . .	338
10.6.2	Programmierung . . . . .	340
10.6.3	Weiterentwicklungen . . . . .	343
10.7	Projekt: Fotos per E-Mail verschicken . . . . .	344
10.8	Randbemerkung: Was darf man? Was soll man? . . . . .	345
10.9	Aufgabe: Wie lang? Wie breit? . . . . .	346

<b>11</b>	<b>Webserver</b> . . . . .	349
11.1	Der RPi im lokalen Netz . . . . .	349
	11.1.1 WLAN . . . . .	349
	11.1.2 Virtual Network Computing (VNC) . . . . .	350
11.2	Webserver . . . . .	352
	11.2.1 Der Apache-Webserver . . . . .	352
	11.2.2 Die eigene Startseite . . . . .	353
11.3	Was ist los im Gartenteich? . . . . .	355
	11.3.1 Projekt: Einfache Webcam mit statischer Webseite . . . . .	355
11.4	Lösungen zu den Zwischenfragen . . . . .	358
<b>12</b>	<b>Erweiterungen: OLED und HAT</b> . . . . .	359
12.1	OLED-Display . . . . .	359
12.2	Anschluss . . . . .	359
12.3	Installation der SSD1306-Bibliothek . . . . .	360
12.4	Auf dem Display Texte und Formen ausgeben . . . . .	360
	12.4.1 Projekt: Uhrzeit . . . . .	361
12.5	Mit ImageDraw zeichnen und Texte schreiben . . . . .	362
	12.5.1 Grafiken zeichnen . . . . .	363
	12.5.2 Schriftarten definieren . . . . .	365
12.6	Projekt: Thermometer . . . . .	366
12.7	Sense HAT . . . . .	368
12.8	Die Klasse SenseHat . . . . .	369
12.9	Grafische Ausgabe über die LED-Matrix . . . . .	371
12.10	Die räumliche Orientierung des Sense HAT . . . . .	372
	12.10.1 Projekt: Das Murmellabyrinth . . . . .	373
12.11	Der Joystick . . . . .	376
	12.11.1 InputEvent . . . . .	377
	12.11.2 Die Klasse JoyStick . . . . .	378
	12.11.3 Definition von Eventhandlern . . . . .	378
	12.11.4 Auf Events warten und Events abfragen . . . . .	380
12.12	Aufgaben . . . . .	382
<b>A</b>	<b>Den Raspberry Pi einrichten</b> . . . . .	383
<b>B</b>	<b>Der GPIO</b> . . . . .	389
<b>C</b>	<b>Autostart</b> . . . . .	393
<b>D</b>	<b>Bau eines Fußschalters</b> . . . . .	395
<b>E</b>	<b>Virtuelle Umgebungen</b> . . . . .	397
	<b>Stichwortverzeichnis</b> . . . . .	401





# Einleitung

Der Raspberry Pi – kurz RPi – ist ein preiswerter, kreditkartengroßer Computer, der fast keinen Strom verbraucht, eine SD-Karte als Peripheriespeicher verwendet und an einen hochauflösenden Monitor angeschlossen werden kann. Mittlerweile gibt es ihn in vielen Varianten. Das neuste Modell ist der Raspberry Pi 5, der nicht mehr ganz so billig, aber sehr leistungsfähig ist. Es gibt aber eine preiswerte Alternative, den Raspberry Pi Zero W. Alle Projekte in diesem Buch funktionieren auf beiden Modellen und auf älteren RPis ab Modell 3.

Der RPi beflügelt die Fantasie von Bastlern, professionellen Technikern und Wissenschaftlern. In Kombination mit der Programmiersprache Python bietet er eine wunderbare Umgebung zur Realisierung technischer Ideen.

Dieses Buch erklärt alles von Grund auf. Es werden keine Vorkenntnisse zu Linux, zur Programmierung und zur Hardware des Raspberry Pi vorausgesetzt. Im Anhang finden Sie Hinweise zur Hardware und eine Schritt-für-Schritt-Anleitung zur Installation des Betriebssystems.

## Was macht man mit dem Raspberry Pi?

Dieses Buch ist eine Einführung in die Programmiersprache Python auf dem Raspberry Pi. Doch die Beschäftigung mit dem Raspberry Pi ist oft nicht nur reine Programmierung. Ziel eines typischen RPi-Projekts ist der Prototyp einer kompletten Maschine – Hardware und Software. Der RPi legt Technik, die sonst versteckt ist, offen. Auf dem Markt gibt es eine zunehmende Zahl elektronischer Bauteile, die man mit dem RPi verbinden kann. Zudem gibt es immer mehr Firmen, die das benötigte Material im Internet anbieten. Bestellung und Lieferung der oft sehr speziellen Bauteile sind heute kein Problem.

Dieses Buch will eine Idee vom Charme der Programmiersprache Python vermitteln. Darüber hinaus soll es inspirieren, das gelernte Programmierwissen in konkrete Projekte einfließen zu lassen. Damit die Beschreibung von Hardwaretechnik und spezieller Schnittstellen nicht ins Uferlose wächst, gehe ich von fünf allgemeinen Hardwarekonfigurationen aus.

## Interaktives Exponat

Auf dem Raspberry Pi läuft ein interaktives Programm mit grafischer Benutzeroberfläche. Das kleine Gerät ist hinter einen großen Touchscreen geklebt und nicht zu sehen. Der RPi startet das Programm automatisch beim Einschalten. Eine solche Anordnung kann ein interaktives Exponat einer Ausstellung oder ein Auskunftssystem im Foyer eines öffentlichen Gebäudes sein. Da der RPi Grafik in HD-Qualität unterstützt, ist er für diesen Zweck hervorragend geeignet. Projekte dieser Art sind Maschinen, die Bilder und Texte automatisch erzeugen (Kapitel 5), digitale Karteikästen, Staumelder, die Informationen aus dem Internet auswerten und einen Überblick über die aktuelle Verkehrslage geben (Kapitel 6), digitale Bilderrahmen und Kalender (Kapitel 7) oder Animationen, die ihr Verhalten ändern, wenn sich jemand dem Bildschirm nähert (Kapitel 9).

## System mit speziellen Eingabegeräten

Bei diesem Typ ist der RPi mit Sensoren oder einer Kamera verbunden. Auf dem Computer läuft ein Programm, das auf Signale dieser Sensoren reagiert. Das kann z.B. ein Spiel sein, bei dem Objekte auf dem Bildschirm über Fußschalter gesteuert werden. Die Sensoren kann man sich mit wenigen Elektronikbauteilen (Kabeln, Widerständen, Thermoelementen, AD-Wandlern) und Alltagsmaterialien (Pappe, Alufolie, Schaumgummi) zusammenbauen. Schon in den ersten beiden Kapiteln finden Sie einfache Beispiele für Programme, die Signale externer Schalter verarbeiten und LEDs oder Relais ansteuern: z.B. Zähler, Alarmanlagen. Komplexere Projekte mit einer grafischen Oberfläche sind z.B. ein Pong-Spiel, bei dem ein Schläger auf dem Bildschirm mit einer selbst gebauten Konsole bewegt wird (Kapitel 8) oder eine Simulation, bei der Sie ein Auto mit einem Potentiometer steuern. Ein ganz besonderes Eingabegerät ist das Kameramodul des Raspberry Pi. In Kapitel 10 werden Projekte vorgestellt, bei denen das Livebild der Kamera auf dem Bildschirm dargestellt und ausgewertet wird. Bewegungen werden erkannt und sogar die Bewegungsrichtung eines Objekts erfasst. Für den RPi gibt es eine Reihe von speziellen Sensoren (Kohlendioxid, Alkohol, Licht), mit denen man Geräte wie Spektralphotometer oder naturwissenschaftliche Experimente entwickeln kann, die automatisch oder halbautomatisch durchgeführt und ausgewertet werden (Kapitel 9).

## System mit speziellen Ausgabegeräten

Viele Computerprogramme verwenden den Monitor als Ausgabegerät für Texte oder Bilder. Typische RPi-Projekte experimentieren aber auch mit anderen Ausgabegeräten. Zum Beispiel kann man die aktuelle Uhrzeit oder Temperatur auch auf einem OLED-Display anzeigen (Kapitel 12) oder über eine Bluetooth-Box als gesprochenen Text ausgeben (Kapitel 9). Mithilfe von MOSFET-Chips können Sie farbige LED-Strips ansteuern (Kapitel 1).

## Mobiles Gerät

Der RPi ist klein und braucht wenig Strom. Er ist deshalb sehr gut für mobile Geräte und autonome Roboter geeignet. Bei den Projekten in diesem Buch steht die Programmierung im Vordergrund. Die Hardware ist möglichst einfach und enthält Bauteile (Steckplatinen, LEDs, Widerstände, Thermoelement, Ultraschallsensor, Kamera, OLED-Display), die man auch noch für andere Vorhaben verwenden kann. Typische Anwendungen sind mobile Messgeräte, die Messwerte (z.B. die Temperatur) speichern, oder ein Suchgerät, das in der Lage ist, die heißeste Stelle im Raum zu finden (Kapitel 9). In Kapitel 11 finden Sie ein Beispiel für eine mobile Webcam. Eine spezielle Erweiterung des RPi ist die HAT-Technik. In Kapitel 12 wird das Sense-HAT vorgestellt, eine Zusatzplatine, die man auf den RPi aufschraubt und die einige Sensoren und eine LED-Matrix enthält.

## Server für spezielle Aufgaben

Der RPi kostet wenig und benötigt eine elektrische Leistung von nur 3,5 Watt (Modell B). Damit ist er der ideale Server, der permanent arbeitet und ständig bereit ist, Anfragen über das Internet oder Intranet zu beantworten (HTTP-Server). Ein solcher Server braucht weder Tastatur noch Monitor. Er kann über eine VNC-Verbindung von einem anderen Rechner aus gesteuert werden. In Kapitel 11 finden Sie Beispiele für serverbasierte Projekte, darunter eine Webcam.

## Warum überhaupt Python auf dem RPi?

Ursprünglich sollte der RPi mit einem fest eingebauten Interpreter für Python-Programme ausgestattet werden (Pi steht für *Python interpreter*). Aber letztlich ist das Design doch flexibler geworden. Betriebssystem und Programmiersprachen können nach Wunsch installiert werden. Python ist besonders leicht zu lernen und erlaubt aber dennoch die Entwicklung komplexer Programme. Ein Vorteil für experimentelle Projekte mit dem Raspberry Pi ist, dass ein reicher Schatz freier Softwarepakete zur Verfügung steht, die man aus dem öffentlichen Python Package Index (PyPI) herunterladen und installieren kann.

## Zum Aufbau dieses Buchs

In den Kapiteln werden Schritt für Schritt die wesentlichen Elemente der Python-Programmierung eingeführt. Ab Kapitel 2 werden kleine in sich abgeschlossene Projekte beschrieben, die praktische Anwendungsmöglichkeiten der zuvor eingeführten Techniken illustrieren. Dabei spielen in den ersten Kapiteln Peripheriegeräte noch keine Rolle. Der RPi wird in einer Standard-Hardwarekonfiguration mit Tastatur, Maus und Monitor benutzt wie ein normaler Computer. Die Elemente der Programmiersprache Python werden Schritt für Schritt eingeführt, von den

elementaren Grundlagen bis zu fortgeschrittenen Techniken der objektorientierten Programmierung. Die spannenderen Projekte kommen weiter hinten. Haben Sie also zu Beginn etwas Geduld.

In Kapitel 5 werden grafische Benutzungsoberflächen eingeführt. Die meisten Projekte sind nun Anwendungsprogramme, die nicht mehr auf eine Tastatur angewiesen sind und leicht zu interaktiven Exponaten für Ausstellungen oder Ähnlichem weiterentwickelt werden können. Etwa ab der Mitte des Buchs enthält jedes Kapitel Anregungen und Beispiele für Projekte mit speziellen Hardware-Komponenten, wie Kamera, Temperatur-Sensoren, AD-Wandlern, Schaltern und LEDs. Am Ende jedes Kapitels finden Sie Aufgaben mit Lösungen zum Download, mit denen Sie Ihr Wissen festigen, erweitern und vertiefen können.

Speziellere Informationen zum Betriebssystem (Installation, Autostart) und zur Hardware des RPi (GPIO) finden Sie in den Anhängen.

### **Lösungen und Programmcode zum Download**

Unter der Webadresse <http://www.mitp.de/0829> können Sie die Listings aus dem Buch sowie die Lösungen zu den Aufgaben kostenlos downloaden.

# Begegnung mit Python

In diesem Kapitel verwenden Sie Python im interaktiven Modus. Sie geben in der Kommandozeile der IDLE-Shell einzelne Befehle ein, die der Python-Interpreter sofort ausführt. Sie lernen Operatoren, Datentypen, die Verwendung von Funktionen und den Aufbau von Termen kennen. Dabei bekommen Sie einen ersten Eindruck vom Charme und der Mächtigkeit der Programmiersprache Python. Ich gehe davon aus, dass Sie bereits ein fertig konfiguriertes Computersystem besitzen, bestehend aus SD-Karte, Tastatur, Netzteil, Monitor und natürlich – als Herzstück – den Raspberry Pi, der meist als Raspi oder RPi abgekürzt wird. Auf der SD-Speicherkarte ist als Betriebssystem Raspberry Pi OS installiert. Das neueste und leistungsstärkste Modell des Minicomputers ist der Raspberry Pi 5. Alle Projekte in diesem Buch funktionieren aber auch auf den älteren Modellen 3 und 4 und auch auf dem besonders preiswerten Raspberry Pi Zero WH. Beachten Sie, dass der kleine Raspberry Pi Zero sehr langsam ist. Haben Sie mit ihm Geduld und halten Sie nur möglichst wenige Fenster geöffnet.

Falls Sie noch nicht so weit sind, können Sie in Anhang A nachlesen, welche Komponenten Sie benötigen und wie Sie bei der Einrichtung Ihres RPi-Systems vorgehen.

## 1.1 Was ist Python?

Python ist eine Programmiersprache, die so gestaltet wurde, dass sie leicht zu erlernen ist und besonders gut lesbare Programmtexte ermöglicht. Ihre Entwicklung wurde 1989 von Guido van Rossum am Centrum voor Wiskunde en Informatica (CWI) in Amsterdam begonnen und wird nun durch die nichtkommerzielle Organisation »Python Software Foundation« (PSF, <http://www.python.org/psf>) koordiniert. Das Logo der Programmiersprache ist eine stilisierte Schlange. Dennoch leitet sich der Name nicht von diesem Schuppenkriechtier ab, sondern soll an die britische Comedy-Gruppe Monty Python erinnern.

Ein Python-Programm – man bezeichnet es als Skript – wird von einem Interpreter ausgeführt und läuft auf den gängigen Systemplattformen (Unix, Windows, Mac OS). Ein Programm, das auf Ihrem Raspberry Pi funktioniert, läuft in der Regel auch auf einem Windows-Rechner. Python ist kostenlos und kompatibel mit der GNU General Public License (GPL).


Python ist objektorientiert, unterstützt aber auch andere Programmierparadigmen (z.B. funktionale und imperative Programmierung). Python ist eine universelle Programmiersprache mit vielen Einsatzmöglichkeiten. Es wird in der Wissenschaft und Technik verwendet (z.B. im Deutschen Zentrum für Luft- und Raumfahrt), aber auch für visuell-kreative Projekte (z.B. bei Disney zur Entwicklung von Computerspielen). Python hat gegenüber älteren Programmiersprachen drei Vorteile:

- Python ist leicht zu erlernen und hat eine niedrige »Eingangsschwelle«. Ohne theoretische Vorkenntnisse kann man sofort die ersten Programme schreiben. Im interaktiven Modus kann man einzelne Befehle eingeben und ihre Wirkung beobachten. Es gibt nur wenige Schlüsselwörter, die man lernen muss. Gewohnte Schreibweisen aus der Mathematik können verwendet werden, z.B. mehrfache Vergleiche wie  $0 < a < 10$ .
- Python-Programme sind kurz und gut verständlich. Computerprogramme werden von Maschinen ausgeführt, aber sie werden für Menschen geschrieben. Software wird meist im Team entwickelt. Programmtext muss für jedermann gut lesbar sein, damit er verändert, erweitert und verbessert werden kann. Der berühmte amerikanische Informatiker Donald Knuth hat deshalb schon vor drei Jahrzehnten vorgeschlagen, Programme als Literatur zu betrachten, so wie Romane und Theaterstücke. Nicht nur Korrektheit und Effizienz, auch die Lesbarkeit ist ein Qualitätsmerkmal.
- Programme können mit Python nachweislich in kürzerer Zeit entwickelt werden als mit anderen Programmiersprachen. Das macht Python nicht nur für die Software-Industrie interessant; auch Universitäten verwenden immer häufiger Python, weil so weniger Zeit für den Lehrstoff benötigt wird.

## 1.2 IDLE

Zur Standardinstallation von Python gehört eine integrierte Entwicklungsumgebung namens IDLE. Der Name erinnert an den englischen Schauspieler, Autor und Komponisten Eric Idle, ein Mitglied der Comedy-Gruppe Monty Python.

IDLE ist nicht (mehr) in der Standarddistribution von Raspberry Pi OS enthalten. Sie können es aber in wenigen Schritten installieren:

- Klicken Sie oben links auf den Startbutton mit der Himbeere, dann auf EINSTELLUNGEN und dann auf ADD/REMOVE SOFTWARE (Software hinzufügen oder entfernen). Es erscheint ein Fenster wie in Abbildung 1.1.
- Geben Sie oben links als Suchbegriff `Idle` ein und drücken Sie . Es erscheint eine Liste von Software-Angeboten.
- Rollen Sie etwas herunter und wählen Sie eine IDLE-Version, indem Sie die Checkbox vor dem Eintrag anklicken.

- Klicken Sie unten rechts auf OK.
- Zur Sicherheit müssen Sie im Authentifizierungsfenster Ihr Passwort eingeben.
- Dann wird IDLE installiert.

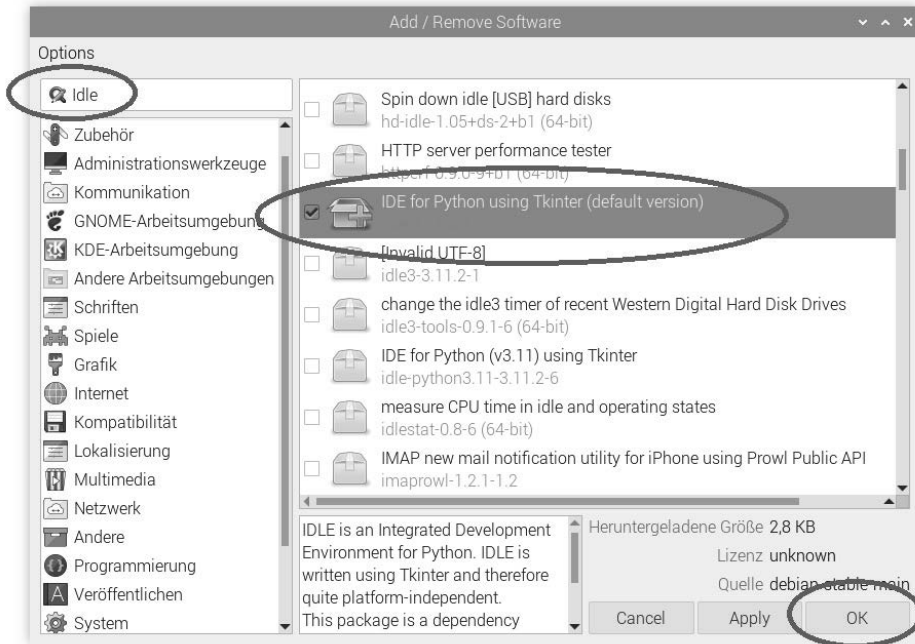


Abb. 1.1: IDLE installieren

Klicken Sie oben links auf den Startbutton mit der Himbeere und öffnen Sie das Untermenü ENTWICKLUNG. Sie sehen, dass das Menü einen neuen Eintrag enthält, z.B.: IDLE (USING PYTHON 3.11). Wenn Sie diesen Eintrag anklicken, öffnet sich die Entwicklungsumgebung. Beachten Sie, dass die Python-Version von Ihrem Raspberry-Pi-Modell abhängt. Auf dem kleinen RPi Zero kann eine ältere Version installiert sein als auf dem RPi 5.

IDLE besteht im Wesentlichen aus drei Komponenten:

- **Die IDLE-Shell.** Wenn Sie IDLE starten, öffnet sich zuerst das IDLE-Shell-Fenster. Die Shell ist eine Anwendung, mit der Sie direkt mit dem Python-Interpreter kommunizieren können: Sie können auf der Kommandozeile einzelne Python-Anweisungen eingeben und ausführen lassen. Ein Python-Programm, das eine Bildschirmausgabe liefert, gibt diese in einem Shell-Fenster aus.

- **Der Programmierer.** Das ist eine Art Textverarbeitungsprogramm zum Schreiben von Programmen. Sie starten den Programmierer vom Shell-Fenster aus (FILE|NEW FILE).
- **Der Debugger.** Er dient dazu, den Lauf eines Programms zu kontrollieren und zu überwachen, um logische Fehler zu finden.

Neben IDLE gibt es natürlich noch viele andere Entwicklungsumgebungen für Python. Auf dem RPi sind standardmäßig noch GEANY und THONNY installiert.

## 1.2.1 IDLE-Shell

Anweisungen sind die Bausteine von Computerprogrammen. Mit der IDLE-Shell können Sie einzelne Python-Anweisungen ausprobieren. Man spricht auch vom interaktiven Modus, weil Sie mit dem Python-Interpreter eine Art Dialog führen: Sie geben über die Tastatur einen Befehl ein – der Interpreter führt ihn aus und liefert eine Antwort.

Öffnen Sie das Shell-Fenster der Python-3-Version auf Ihrem Rechner (STARTBUTTON|ENTWICKLUNG|IDLE). Da Sie ständig mit IDLE arbeiten werden, lohnt es sich das Programmicon auf den Bildschirm zu bringen. Das geht so: Sie klicken mit der rechten Maustaste auf das Icon PYTHON 3 und wählen im Kontextmenü den Befehl ZUR ARBEITSFLÄCHE HINZUFÜGEN. Die IDLE-Shell meldet sich immer mit einer kurzen Information über die Version und einigen weiteren Hinweisen.

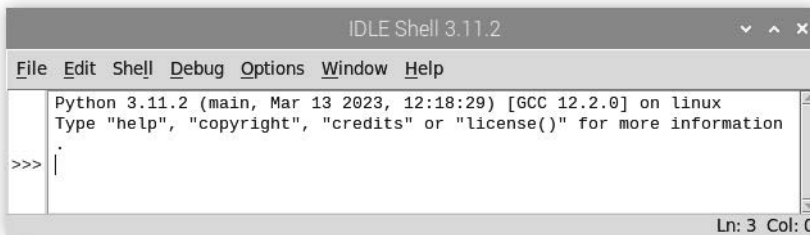



Abb. 1.2: Die IDLE-Shell

Die unterste Zeile beginnt mit einem Promptstring aus drei spitzen Klammern >>> als Eingabeaufforderung. Das ist die Kommandozeile. Hinter dem Prompt können Sie eine Anweisung eingeben. Wenn Sie die Taste  drücken, wird der Befehl ausgeführt. In den nächsten Zeilen kommt entweder eine Fehlermeldung, ein Ergebnis oder manchmal auch *keine* Systemantwort. Probieren Sie aus:

```
>>> 2+2
4
```



Hier ist die Anweisung ein mathematischer Term. Wenn Sie  $\leftarrow$  drücken, wird der Term ausgewertet (also die Rechnung ausgeführt) und in der nächsten Zeile (ohne Prompt) das Ergebnis dargestellt.

```
>>> 2 +  
SyntaxError: invalid syntax
```

Jetzt haben Sie einen ungültigen Term eingegeben (der zweite Summand fehlt). Dann folgt eine Fehlermeldung.

```
>>> a = 1257002  
>>>
```

Eine solche Anweisung nennt man eine Zuweisung. Der Variablen `a` wird der Wert `1257002` zugewiesen. Dabei ändert sich zwar der Zustand des Python-Laufzeitsystems (es merkt sich eine Zahl), aber es wird nichts ausgegeben. Sie sehen in der nächsten Zeile sofort wieder der Prompt. Die gespeicherte Zahl können Sie wieder zum Vorschein bringen, indem Sie den Variablennamen eingeben:

```
>>> a  
1257002
```

## 1.2.2 Hotkeys

Um effizient mit der IDLE-Shell arbeiten zu können, sollten Sie einige Tastenkombinationen (Hotkeys) beherrschen.

Manchmal möchten Sie ein früheres Kommando ein zweites Mal verwenden – vielleicht mit kleinen Abänderungen. Dann benutzen Sie die Tastenkombination  $\text{Alt}+\text{P}$ . Beispiel:

```
>>> 1 + 2*3 + 4  
11  
>>>
```

Wenn Sie jetzt *einmal* die Tastenkombination  $\text{Alt}+\text{P}$  betätigen, erscheint hinter dem letzten Prompt wieder das vorige Kommando (*previous*):

```
>>> 1 + 2*3 + 4
```

Wenn Sie nochmals diesen Hotkey drücken, verschwindet der Term wieder und es erscheint das *vorletzte* Kommando, beim nächsten Mal das *vorvorletzte* und so wei-

ter. Die Shell merkt sich alle Ihre Eingaben in einer Folge, die man *History* nennt. Mit `[Alt]+P` und `[Alt]+N` können Sie in der History rückwärts- und vorwärtsgehen (Abbildung 1.3).

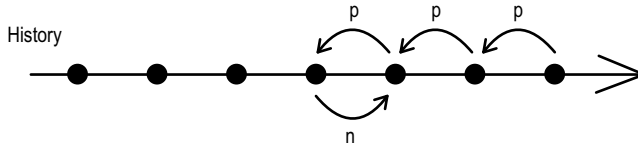


Abb. 1.3: Navigieren in der History mit `[Alt]+P` und `[Alt]+N`

Eine dritte wichtige Tastenkombination, die Sie sich merken sollten, ist `[Strg]+C`. Damit können Sie die Ausführung des gerade laufenden Programms abbrechen, z.B. wenn sie zu lange dauert.

Tastenkombination	Wirkung
<code>[Alt]+P</code>	Previous Command. Die vorige Anweisung in der History (Liste der bisherigen Anweisungen) erscheint hinter dem Prompt.
<code>[Alt]+N</code>	Next Command. Die nachfolgende Anweisung in der History erscheint hinter dem Prompt.
<code>[Strg]+C</code>	Keyboard Interrupt. Der Abbruch eines Programms (z.B. bei einer Endlosschleife) wird erzwungen.

Tabelle 1.1: Wichtige Tastenkombinationen der IDLE-Shell

## 1.3 Die IDLE-Shell als Taschenrechner

Die IDLE-Shell können Sie wie einen mächtigen und komfortablen Taschenrechner benutzen. Sie geben einen mathematischen Term ein, drücken `[↵]` und erhalten das Ergebnis in der nächsten Zeile.

### 1.3.1 Operatoren und Terme

Ein mathematischer Term (Ausdruck) kann aus Zahlen, Operatoren und Klammern aufgebaut werden. Die Schreibweise ist manchmal ein kleines bisschen anders als in der Schulmathematik. Zum Beispiel dürfen Sie beim Multiplizieren den Multiplikationsoperator `*` niemals weglassen. Das Kommando

```
>>> (1 + 1) (6 - 2)
```

führt zu einer Fehlermeldung. Korrekt ist:

```
>>> (1 + 1) * (6 - 2)
8
```

Es gibt keine langen Bruchstriche. Für Zähler oder Nenner müssen Sie Klammern verwenden:

```
>>> (2 + 3) / 2
2.5
```

Python unterscheidet zwischen der exakten Division `/` und der ganzzahligen Division `//`. Die ganzzahlige Division liefert eine ganze Zahl, und zwar den nach unten gerundeten Quotienten. Probieren Sie aus:

```
>>> 3/2
1.5
>>> 3//2
1
```

Die Modulo-Operation `%` liefert den Rest, der bei einer ganzzahligen Division übrig bleibt. Beispiel: 7 geteilt durch 3 ist 2 Rest 1.

```
>>> 7 // 3
2
>>> 7 % 3
1
```

Zum Potenzieren einer Zahl verwenden Sie den Operator `**`. Beachten Sie, dass Sie mit Python fast beliebig große Zahlen berechnen können.

```
>>> 2**3
8
>>> 2**-3
0.125
>>> 2**0.5
1.4142135623730951
>>> 137 ** 57
620972443101902588551304810097687105537832218918245689182643787308016217
31509707020422858215922309341135893663853254591817
```

Bei Termen mit mehreren Operatoren müssen Sie deren Prioritäten beachten (Tabelle 1.2). Ein Operator mit höherer Priorität wird zuerst ausgewertet. Der Potenzoperator hat die höchste Priorität, Addition und Subtraktion die niedrigste.

```
>>> 2*3**2
18
>>> (2*3)**2
36
```

Operator	Bedeutung
**	Potenz, $x^{**}y = x^y$
*, /, //	Multiplikation, Division und ganzzahlige Division
%	Modulo-Operation. Der Rest einer ganzzahligen Division.
+, -	Addition und Subtraktion

Tabelle 1.2: Arithmetische Operatoren in der Reihenfolge ihrer Priorität

### 1.3.2 Zahlen

Wer rechnet, verwendet Zahlen. Mit Python können Sie drei Typen von Zahlen verarbeiten:

- Ganze Zahlen (`int`)
- Gleitpunktzahlen (`float`)
- Komplexe Zahlen (`complex`)

Was ist überhaupt eine Zahl? In der Informatik unterscheidet man zwischen dem abstrakten mathematischen Objekt und der Ziffernfolge, die eine Zahl darstellt. Letzteres nennt man auch *Literal*. Für ein und dieselbe Zahl im mathematischen Sinne, sagen wir die 13, gibt es unterschiedliche Literale, z.B. 13 oder 13.0 oder 13.0000. Drei Schreibweisen – eine Zahl.

#### Ganze Zahlen (Typ `int`)

Literale für ganze Zahlen sind z.B. 12, 0, -3. Ganze Dezimalzahlen bestehen aus den zehn Ziffern 0, 1, 2, 3, 4, 5, 6, 7, 8, 9. Es kann ein negatives Vorzeichen vor die erste Ziffer geschrieben werden. Eine ganze Dezimalzahl darf nicht mit einer Null beginnen. Probieren Sie es aus!

```
>>> 023
SyntaxError: invalid token
```

Ganze Zahlen dürfen bei Python beliebig lang sein. Eine Grenze ist nur durch die Speicherkapazität des Rechners gegeben. Probieren Sie es aus:

Ganze Zahlen sind vom Typ `int` (*engl.* integer = unversehrt, ganz). Mit der Funktion `type()` können Sie den Typ eines Literals abfragen:

```
>>> type(13)
<class 'int'>
>>> type(13.0)
<class 'float'>
```


Sie sehen, dass die Literale `13` und `13.0` zu verschiedenen Typen gehören, obwohl sie den gleichen mathematischen Wert darstellen. Was hat es mit dem Begriff `class` auf sich? Der Typ `int` wird durch eine Klasse (engl. *class*) realisiert. Eine Klasse kann man sich als eine Art Bauplan für Objekte eines Typs vorstellen. In der Klasse `int` ist z.B. definiert, wie die Operationen für ganze Zahlen ausgeführt werden. Mehr Informationen zu Klassen finden Sie in Abschnitt 3.8.2.

## Binär, oktal und hexadezimal – andere Schreibweisen für ganze Zahlen

Üblicherweise verwenden wir das dezimale Zahlensystem. Es gibt aber auch Binärzahlen (mit nur zwei Ziffern 0 und 1), Oktalzahlen (mit acht verschiedenen Ziffern) und Hexadezimalzahlen (mit 16 Ziffern).

Wie das Dezimalsystem ist auch das *Binärsystem* (Dualsystem, Zweiersystem) ein Stellenwertsystem für Zahlen. Aber anstelle von zehn Ziffern gibt es nur zwei, die Null und die Eins. Jede Zahl lässt sich als Summe von Zweierpotenzen (1, 2, 4, 8, 16, ...) darstellen. Die Binärzahl 10011 hat den Wert

$$1*16 + 0*8 + 0*4 + 1*2 + 1*1 = 16 + 2 + 1 = 19$$

Nun muss man natürlich erkennen können, ob eine Ziffernfolge wie 10011 als Dezimalzahl (zehntausendundelf) oder als Binärzahl gemeint ist. Deshalb beginnen bei Python Literale für Binärzahlen immer mit der Ziffer 0 und dem Buchstaben `b`, also z.B. `0b10011`. Wenn Sie in der IDLE-Shell eine solche Ziffernfolge eingeben und  drücken, erscheint in der nächsten Zeile der Wert als Dezimalzahl:

```
>>> 0b10011
19
```

Mit der Funktion `bin()` können Sie zu einer Dezimalzahl die Binärdarstellung berechnen lassen:

```
>>> bin(19)
'0b10011'
```

Das *Oktalsystem* verwendet die Zahl 8 als Zahlenbasis. Jede Oktalzahl repräsentiert eine Summe aus Achterpotenzen (1, 8, 64, 512, ...). Bei Python beginnen die Literale von Oktalzahlen mit der Ziffer 0 und dem Buchstaben `o` oder `O`. Beispiel:

```
>>> 0o210
136
```

Der Dezimalwert berechnet sich so:  $2*64 + 1*8 + 0*1 = 128 + 8 = 136$

Im *Hexadezimalsystem* ist 16 die Zahlenbasis. Eine Hexadezimalzahl repräsentiert also eine Summe aus Potenzen der Zahl 16. Die 16 Ziffern werden durch die üblichen Dezimalziffern 0 bis 9 und die sechs ersten Buchstaben des Alphabets dargestellt. Dabei hat A den Wert 10, B den Wert 11 usw. Bei Python beginnen Hexadezimalzahlen immer mit den Zeichen 0x oder 0X. Das erste Zeichen ist die Ziffer null und nicht der Buchstabe O. Beispiel:

```
>>> 0x10A
266
```

Der Dezimalwert berechnet sich so:  $1 \cdot 256 + 0 \cdot 16 + 10 \cdot 1 = 256 + 10 = 266$

Die Tatsache, dass die 16 Ziffern der Hexadezimalzahlen auch Buchstaben enthalten, hat Programmierer zum *Hexspeak* inspiriert. Zahlen, die in einem Programmsystem eine besondere Bedeutung haben (*magical numbers*), werden gerne so gewählt, dass ihre Hexadezimaldarstellung ein sinnvolles Wort ist.

```
>>> 0xCAFE
51966
>>> 0xBADBEEF
195935983
>>> xABAD1DEA
2880249322
```

## Gleitkommazahlen (Typ float)

Gleitpunktzahlen oder Gleitkommazahlen (engl. *floating point numbers*) sind Dezimalbrüche. Meist schreibt man eine Gleitkommazahl als eine Folge von Ziffern mit einem einzigen Punkt auf. In der Schulmathematik verwenden wir in Deutschland ein Komma, in Python und allen anderen Programmiersprachen wird die angelsächsische Schreibweise verwendet, bei der ein Punkt an die Stelle des Kommas tritt. Es ist ein bisschen seltsam, von einer Gleitkommazahl zu sprechen und dann einen Punkt zu schreiben. Um diesen Widerspruch zu vermeiden, verwenden viele Leute den Begriff *Gleichpunktzahl*. »Gleitkommazahl« ist übrigens ein Gegenbegriff zu »Festkommazahl«. Eine Festkommazahl ist ein Dezimalbruch mit einer festen Anzahl von Nachkommastellen. Z.B. geben wir Geldbeträge in Euro immer mit zwei Nachkommastellen an. Wir schreiben 3,50 Euro anstelle von 3,5 Euro.

Gültige Python-Gleitkommazahlen sind:

- 3.14 oder 0.2 oder 0.00012
- .2 (eine Null vor dem Punkt darf man auch weglassen)
- 5. (eine Null nach dem Punkt darf man weglassen)

Das Literal 5 ist dagegen keine Gleitkommazahl (Punkt fehlt).

Für Zahlen, die sehr nahe bei 0 liegen oder sehr groß sind, wird die Exponentialschreibweise verwendet. Dabei wird die Zahl als Produkt einer rationalen Zahl  $m$  (Mantisse) mit einer Zehnerpotenz mit dem Exponenten  $e$  dargestellt:

$$z = m \cdot 10^e$$

Beispiele:

$$123000000 = 123 \cdot 10^6$$
$$0.00012 = 1.2 \cdot 10^{-4}$$

Bei Python ist eine Gleitkommazahl in Exponentialschreibweise so aufgebaut: Sie beginnt mit einem Dezimalbruch oder einer ganzen Zahl (ohne Punkt) für die Mantisse. Danach folgt der Buchstabe  $e$  oder  $E$ , ein Vorzeichen (+ oder -), das bei positiven Exponenten auch weggelassen werden kann, und schließlich eine ganze Zahl als Exponent.

Gültige Literale sind:

- 1.0e-8 entspricht der Zahl 0.00000001
- 2.1E+7 entspricht der Zahl 21000000
- .2e0 entspricht der Zahl 0.2
- 001e2 entspricht der Zahl 100 (mehrere führende Nullen sind erlaubt)

Ungültig sind:

- 0.1-E7 (Minuszeichen vor dem E)
- 1.2e0.3 (keine ganze Zahl als Exponent)

Mantisse und Exponent sind immer Dezimalzahlen und niemals Oktal- oder Hexadezimalzahlen.

Gleitkommazahlen sind vom Datentyp `float`. Mit der Funktion `type()` können Sie das nachprüfen:

```
>>> type(1.2)
<class 'float'>
```

Im Unterschied zu ganzen Zahlen (Typ `int`), die immer exakt sind, haben Gleitkommazahlen nur eine begrenzte Genauigkeit. Gibt man längere Ziffernfolgen ein, so werden die letzten Stellen einfach abgetrennt.

```
>>> 1.2345678901234567890
1.2345678901234567
```

### 1.3.3 Mathematische Funktionen

Wenn Sie die IDLE-Shell als Taschenrechner verwenden wollen, benötigen Sie auch mathematische Funktionen wie Sinus, Kosinus, die Quadratwurzelfunktion oder die Exponentialfunktion. Nun stellt Python eine Reihe von Standardfunktionen zur Verfügung, die gewissermaßen fest in die Sprache eingebaut sind (*built-in functions*). Wir haben schon die Funktion `type()` verwendet, die den Typ eines Objekts (z.B. einer Zahl) zurückgibt. Nur wenige Standardfunktionen haben eine mathematische Bedeutung (Tabelle 1.3). Der Aufruf einer Funktion ist so aufgebaut: Zuerst kommt der Name der Funktion, dahinter folgen in runden Klammern die Argumente. Das sind Objekte, die die Funktion verarbeitet, um daraus einen neuen Wert zu berechnen und zurückzugeben. Statt *Argument* sagt man manchmal auch *Parameter*. Beispiel:

```
>>> abs(-12)
12
```

Hier ist `abs` der Name der Funktion und die Zahl `-12` das Argument. Die Funktion `abs()` liefert den positiven Wert einer Zahl. Die Funktion ist einstellig, das heißt, sie akzeptiert immer nur genau ein Argument. Wenn Sie zwei Argumente angeben, erhalten Sie eine Fehlermeldung.

Es gibt aber auch Funktionen, die man mit einer unterschiedlichen Anzahl von Argumenten aufrufen kann. So liefert `min()` die kleinste Zahl von den Zahlen, die als Argumente übergeben worden sind.

```
>>> min(3, 2)
2
>>> min(2.5, 0, -2, 1)
-2
```

Die Funktion `round()` können Sie mit einem oder zwei Argumenten aufrufen. Das erste Argument ist die Zahl, die gerundet werden soll. Das zweite Argument ist optional und gibt die Anzahl der Nachkommastellen an, auf die gerundet werden soll. Fehlt das zweite Argument, gibt die Funktion eine ganze Zahl zurück.



```
>>> round(1.561)
2
>>> round(1.561, 1)
1.6
```

Funktion	Erklärung
<code>abs(x)</code>	Liefert den absoluten (positiven) Wert einer Zahl $x$ .
<code>float(x)</code>	Liefert zu einer Zahl (oder einem anderen Objekt) eine Gleitkommazahl.
<code>int(x)</code>	Liefert zu einer Gleitkommazahl (oder einem anderen Objekt) eine nach unten gerundete ganze Zahl.
<code>max(x0, ..., xn)</code>	Liefert die größte Zahl von $x0, \dots, xn$ .
<code>min(x0, ..., xn)</code>	Liefert die kleinste Zahl von $x0, \dots, xn$ .
<code>round(x, [n])</code>	Die Zahl $x$ wird auf $n$ Stellen nach dem Komma gerundet und das Ergebnis zurückgegeben.

**Tabelle 1.3:** Mathematische Standardfunktionen (müssen nicht importiert werden)

## Module importieren

Die meisten mathematischen Funktionen gehören nicht zu den Standardfunktionen. Will man sie benutzen, muss man zunächst das Modul `math` importieren. Module sind Sammlungen von Funktionen, Klassen und Konstanten zu einer Thematik. Sie sind so etwas wie Erweiterungen der Grundausstattung. Das Modul `math` enthält z.B. die Konstanten `e` und `pi` und mathematische Funktionen wie die Quadratwurzelfunktion `sqrt()`. Für Python gibt es Tausende von Modulen. Die wichtigsten gehören zum Standardpaket von Python und sind auf Ihrem RPi bereits installiert. Speziellere Module müssen zuerst aus dem Internet heruntergeladen werden. Dazu später mehr (z.B. in Abschnitt 9.8.1).

Sie können ein Modul auf verschiedene Weisen importieren. Beispiele:

```
>>> import math
>>> import math as m
>>> from math import *
>>> from math import pi, sqrt
```

Mit dem Befehl

```
>>> import math
```

importieren Sie den Modulnamen. Wenn Sie eine Funktion aufrufen wollen, müssen Sie dem Funktionsnamen noch den Modulnamen voranstellen.

```
>>> math.sqrt(4)
2.0
>>> math.pi
3.141592653589793
```

Sie können ein Modul unter einem anderen Namen importieren. Das ist vor allem dann praktisch, wenn ein Modul einen langen Namen hat, den man im Programmtext nicht immer wieder ausschreiben will.

```
>>> import math as m
>>> m.pi
3.141592653589793
```

Mit dem Befehl

```
>>> from math import *
```

importieren Sie alle Namen des Moduls `math`. Sie können dann die Funktionen und Konstanten ohne vorangestellten Modulnamen verwenden.

```
>>> sqrt(4)
2.0
>>> pi
3.141592653589793
```

Sie können auch gezielt nur die Namen importieren, die Sie auch verwenden wollen. Dann müssen Sie die Namen der benötigten Funktionen und Konstanten auflisten. Beispiel:

```
>>> from math import sqrt
>>> sqrt(4)
2.0
```

## Mathematische Funktionen und Konstanten

Wenn Sie die IDLE-Shell als wissenschaftlichen Taschenrechner verwenden wollen, geben Sie einmal die Import-Anweisung

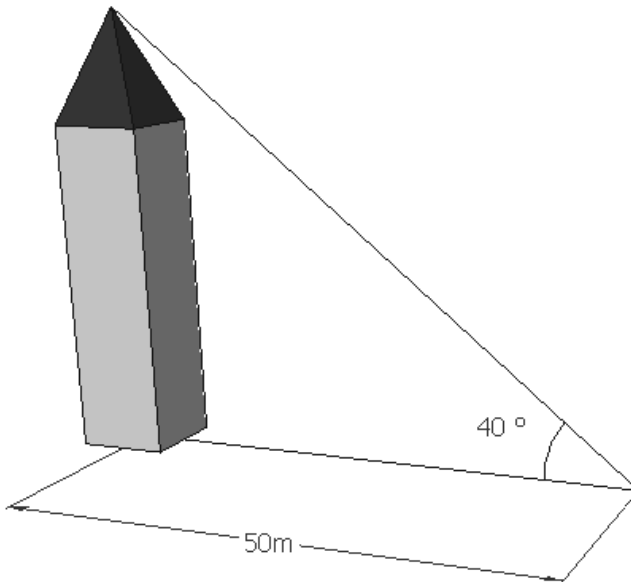
```
>>> from math import *
```

ein. Dann können Sie auf einen Fundus mathematischer Funktionen und Konstanten zurückgreifen. Hier einige Beispiele:

```
>>> degrees (2*pi)
360.0
>>> radians(180)
3.141592653589793
```

Winkel können in Grad  $^{\circ}$  oder als Bogenmaß angegeben werden. Der Winkel von  $360^{\circ}$  entspricht dem Bogenmaß  $2\pi$ .

Mithilfe der Angaben aus Abbildung 1.4 soll die Höhe des Turms berechnet werden.



**Abb. 1.4:** Bestimmung der Höhe eines Turms durch Anpeilen der Turmspitze

Nun verarbeiten die trigonometrischen Funktionen ( $\sin()$ ,  $\cos()$ , ...) des Moduls `math` keine Winkel ( $0^{\circ}$  bis  $360^{\circ}$ ), sondern Bogenmaße (Radianten). Deshalb muss der Winkel aus der Abbildung in den Radianten umgerechnet werden. Die Höhe des Turms können Sie mit folgendem Ausdruck berechnen:

```
>>> tan(radians(40))*50
41.954981558864
```

Funktion/Konstante	Erklärung
<code>acos(x)</code>	Arcuskosinus von $x$ (Bogenmaß)
<code>asin(x)</code>	Arcussinus von $x$ (Bogenmaß)
<code>atan(x)</code>	Arcustangens von $x$ (Bogenmaß)
<code>cos(x)</code>	Kosinus von $x$ ( $x$ ist das Bogenmaß eines Winkels)
<code>cosh(x)</code>	Hyperbolischer Kosinus von $x$
<code>degrees(x)</code>	Liefert zu einem Winkel, der als Bogenmaß angegeben ist, den Winkel in Grad.
<code>e</code>	Die mathematische Konstante $e == 2.7182\dots$
<code>exp(x)</code>	$e^x$
<code>fabs(x)</code>	Absolutwert der Gleitkommazahl $x$
<code>floor(x)</code>	Der nach unten gerundete Wert von $x$ als Gleitkommazahl
<code>log(x)</code>	Natürlicher Logarithmus von $x$ (Basis $e$ )
<code>log10(x)</code>	Logarithmus von $x$ zur Basis 10
<code>modf(x)</code>	Liefert ein Paar bestehend aus dem Nachkommateil und dem ganzzahligen Teil der Gleitkommazahl $x$ . Beispiel: <code>modf(4.2)</code> ergibt <code>(0.2, 4.0)</code> .
<code>radians(x)</code>	Berechnet zum Winkel von $x$ Grad das Bogenmaß.
<code>sin(x)</code>	Sinus von $x$ ( $x$ ist das Bogenmaß eines Winkels)
<code>sinh(x)</code>	Hyperbolischer Sinus von $x$
<code>pi</code>	Die mathematische Konstante $\pi == 3.1415\dots$
<code>sqrt(x)</code>	Quadratwurzel von $x$
<code>tan(x)</code>	Tangens von $x$ ( $x$ ist das Bogenmaß eines Winkels)
<code>tanh(x)</code>	Hyperbolischer Tangens von $x$

Tabelle 1.4: Die wichtigsten Funktionen und Konstanten des Moduls `math`

## 1.4 Hilfe

Mit der Funktion `help()` können Sie detaillierte Informationen zu allen Sprach-elementen von Python abfragen. Sie verwenden `help()` auf zwei unterschiedliche Weisen: mit und ohne Argument.

Wenn Sie Informationen zu einer bestimmten Funktion (oder einem anderen Objekt) benötigen, übergeben Sie den Namen als Argument. Beispiel:

```
>>> help(round)
Help on built-in function round in module builtins:
round(...)
    round(number[, ndigits]) -> number
...
```

Wenn Sie nicht genau wissen, was Sie suchen, geben Sie `help()` ohne Argument ein. Dann wechselt die IDLE-Shell in einen interaktiven Hilfe-Modus. Sie sehen einen neuen Prompt:

```
help>
```

Das Hilfesystem gibt Ihnen Hinweise, was Sie nun tun können. Wenn Sie z.B. eine Liste aller verfügbaren Module haben wollen, geben Sie `modules` ein. Um den Hilfe-Modus zu verlassen, geben Sie `quit` ein.

```
help> quit
You are now leaving help and returning to the Python interpreter ...
>>>
```

## 1.5 Namen und Zuweisungen

Namen für Objekte spielen in der Programmierung eine wichtige Rolle. Bei einer Zuweisung wird ein Name mit einem Objekt (z.B. die Zahl 12) verbunden. Über den Namen kann man auf das Objekt später wieder zugreifen. Der Zuweisungsoperator ist ein Gleichheitszeichen. Links vom Gleichheitszeichen steht ein Name, rechts ein Ausdruck.

Die einfachste Form der Zuweisung hat die Form

```
name = wert
```

Beispiel:

```
>>> x = 12
```

Über den Namen `x` kann man auf den zugeordneten Wert wieder zugreifen:

```
>>> x
12
```

Eine solche Zuweisung kann man sich anschaulich als Speichern von Daten vorstellen. Die Variable ist eine Art Behälter. Der Name `x` ist ein Etikett auf dem Behälter und der Zahlenwert 12 ist der Inhalt des Behälters.

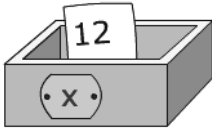


Abb. 1.5: Visualisierung einer Variablen als Behälter mit Inhalt

Wenn ein Name einmal in einer Zuweisung verwendet worden ist, kann er in Ausdrücken anstelle des Zahlenwerts verwendet werden.

```
>>> x*2  
24
```

Das Einführen von Namen ist bei Python ein sehr einfacher Mechanismus. Im Unterschied zu anderen Programmiersprachen (wie z.B. Java) müssen Namen nicht deklariert und mit einem Datentyp verknüpft werden.

Statt *Name* sagt man im Deutschen häufig auch *Bezeichner*. Beides meint dasselbe. Das Wort *Bezeichner* ist eine schlechte Übersetzung des englischen Begriffs *identifier*. Der englische Begriff weist auf die wichtigste Funktion von Namen hin, nämlich Objekte eindeutig zu identifizieren.

Der Inhalt einer Variablen kann einer anderen Variablen zugewiesen werden.

```
>>> y = x  
>>> y  
12
```

Nun trägt *y* denselben Wert wie *x*. Diese Zuweisung stellen sich viele Leute wie in Abbildung 1.6 vor. Die Variable *y* erhält eine Kopie des Inhalts von *x*.

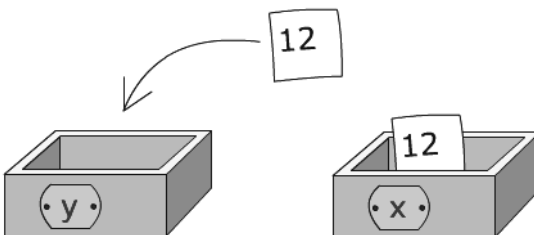


Abb. 1.6: Visualisierung einer Wertübertragung

Namen kann man sich auch als Etiketten vorstellen, die an Objekten (wie Zahlen) kleben. Man kann auch sagen, dass die Zahl 12 nun an zwei Namen gebunden ist. In vielen Fällen ist diese Etiketten-Intuition angemessener.

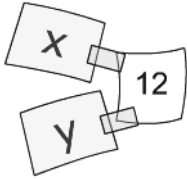


Abb. 1.7: Zuweisen als Hinzufügen eines neuen Namens

### 1.5.1 Zuweisungen für mehrere Variablen

Die Programmiersprache Python ist so gestaltet, dass man möglichst kurze und verständliche Programmtexte schreiben kann. Das merkt man an vielen Stellen – auch bei Zuweisungen. Sie können in einer einzigen Anweisung mehrere Variablen mit dem gleichen Wert belegen. Anstelle von

```
>>> x = 1
>>> y = 1
```

schreiben Sie

```
>>> x = y = 1
```

Und so ordnen Sie in einer einzigen Zuweisung mehreren Variablen *unterschiedliche* Werte zu:

```
>>> x, y = 1, 2
```

Prüfen Sie nach:

```
>>> x
1
>>> y
2
```

Es ist möglich, in einer einzigen Zuweisung ohne Hilfsvariable die Werte zweier Variablen zu vertauschen.

```
>>> x, y = 1, 2
>>> x, y = y, x
>>> x
2
>>> y
1
```

## 1.5.2 Rechnen mit Variablen in der Shell

Im Unterschied zur Mathematik sind Namen in Computerprogrammen meist nicht nur einzelne Buchstaben (wie z.B.  $x$ ), sondern ganze Wörter, die eine sinnvolle Bedeutung haben (»sprechende Namen«). Das erleichtert das Verstehen von Berechnungen.

```
>>> from math import pi
>>> erddurchmesser = 12756
>>> erdradius = erddurchmesser/2
>>> erdvolumen = 4/3 * pi * erdradius ** 3
>>> erdoberfläche = 4 * pi * erdradius ** 2
>>> erdvolumen
1086781292542.8892
>>> round(erdoberfläche)
511185933
```

## 1.5.3 Syntaxregeln für Bezeichner

Man darf nicht jede beliebige Zeichenfolge als Namen (Bezeichner) verwenden. Es sind folgende Regeln zu beachten:

1. Ein Bezeichner darf kein Python-Schlüsselwort sein (*key word*). Schlüsselwörter sind reservierte Wörter, die eine festgelegte Bedeutung haben, z.B. `not` (Negation einer logischen Aussage) oder `if` (Start einer bedingten Anweisung). Eine vollständige Liste der Schlüsselwörter erhalten Sie in der IDLE-Shell über die Hilfe-Funktion. Geben Sie zuerst `help()` ein und wechseln Sie in den Hilfemodus (der Prompt ist nun `help>`). Dann geben Sie das Kommando `keywords` ein. Mit `quit` gelangen Sie wieder in den Interpreter-Modus der Shell.
2. Ein Bezeichner besteht nur aus Buchstaben (`a...z`, `A...Z`), Ziffern (`0...9`) oder Unterstrichen (`_`).
3. Ein Bezeichner muss mit einem Buchstaben oder Unterstrich beginnen.



Gültige Bezeichner sind: `summe`, `__summe`, `summe_1`, nicht aber `1_summe` (Ziffer zu Beginn ist nicht erlaubt), `summe-1` (Minuszeichen ist nicht erlaubt).

Tückisch ist es, wenn Sie (vielleicht aus Versehen) einen Funktionsnamen als Variablennamen verwenden. Das ist zwar erlaubt, aber Sie können anschließend die Funktion nicht mehr aufrufen. Beispiel:

```
>>> int(1.2)
1
```

Die Funktion `int()` liefert zu einer Zahl eine ganze Zahl. Nun ordnen wir dem Namen `int` eine Zahl zu.

```
>>> int = 12.78
>>> int
12.78
```

Damit ist die Funktion `int()` nicht mehr verfügbar:

```
>>> int(1.2)
Traceback (most recent call last):
...
TypeError: 'float' object is not callable
```

Die Fehlermeldung ist so zu lesen: Der Name `int` ist nun einer Gleitkommazahl (Objekt vom Typ `float`) zugeordnet und kann nicht (wie eine Funktion) aufgerufen werden.

#### 1.5.4 Neue Namen für Funktionen und andere Objekte

Mittels einer Zuweisung können Sie einer Funktion einen neuen (zusätzlichen) Namen geben:

```
>>> runden = round
```

Nun haben Sie für die Funktion `round()` einen zweiten (deutschen) Namen eingeführt. Probieren Sie aus:

```
>>> runden(1.23)
1
```

Solche Zuweisungen kann man sich als Benennungen vorstellen. Es ist so, als ob Sie einem Freund einen Spitznamen geben. Damit verschwindet der richtige Name nicht. Aber der Spitzname ist eine zweite Möglichkeit, die Person anzusprechen.

### 1.5.5 Erweiterte Zuweisungen

Eine erweiterte Zuweisung ist eine Kombination aus einer Operation und einer Zuweisung. Sie bewirkt die Änderung des Wertes einer Variablen. Der Operator einer erweiterten Zuweisung endet mit einem Gleichheitszeichen: +=, -=, \*=, /=, //=, \*\*=.

Die Anweisung

```
>>> x += 2
```

liest man so: »Der Wert von x wird um 2 erhöht.« Diese Anweisung hat die gleiche Wirkung wie

```
>>> x = x + 2
```

Bei einer erweiterten Zuweisung wird der aktuelle Wert der Variablen als erster Operand gewählt. Der zweite Operand ist der Wert des Ausdrucks, der hinter dem Gleichheitszeichen steht. Auf beide Werte wird die Operation angewendet und das Ergebnis der Variablen vor dem Gleichheitszeichen zugewiesen.

Weitere Beispiele:

```
>>> a = 2
>>> a *= 3
>>> a
6
>>> a //= 2
>>> a
3
>>> a **= 3
>>> a
27
```

Inkrementierungen und Dekrementierungen (wie z.B. x++, x-- bei Java oder C) gibt es bei Python übrigens nicht.

# Stichwortverzeichnis

100 Farben (Projekt) 165

1-Wire-Bus 259

\_thread 215

.venv 293

## A

Absorptionsdiagramm 298

Absorptionsspektrometer 298

Abstandsmessungen (Projekt) 312

activate 292

adafruit\_tsl256 292

adafruit-circuitpython-tsl2561 292

Aggregat 239

Aggregation 239

Alarmanlage (Projekt) 79

Alias 103

Alkoholgehalt messen (Projekt) 284

Alkoholisches Getränk (Projekt) 281

Alkoholsensor 280

Alphabet (Projekt) 236

anchor 157

Animierte Uhr 382

anonymes Objekt 256

Anweisungsblock 128

Apache 352

Apache-Server

    anhalten 353

    starten 353

Application Launch Bar 386

Argument 28, 125

asctime() 211

Astro Pi 368

Ausdruck 22

    regulärer 199

Ausführbar (executable) 57

Autosimulator 319

Autostart 219, 393

## B

Bedingung 64

Benutzungsoberfläche

    grafische 151

Betriebssystem

    installieren 383

Bewegung erfassen (Projekt) 326

Bezeichner 36

    Syntaxregeln 36

Binärmodus 177

bind() 218

Birnen oder Tomaten (Projekt) 337

Blinken (Projekt) 84

Blinklicht 75

Body-Mass-Index 68

Boolescher Ausdruck 83

Botschaft 101

Breadboard 39

break 74, 91

Broadcom-Nummerierung (BCM) 42

Built-in function 28, 125

Bytestring 98

## C

Callback-Funktion 143

Canvas (Klasse)

    Methoden 248

Canvas (Objekt) 244

Casting 88

CGI-Skript 352

Channel 116

chdir () 220

Checkbox 160

close() 173

Colorimetrie 343

Comprehension 108

crop() 222

CSI-Verbinder 321

## D

Datei 173

Datenlogger (Projekt) 305

def 128

Dialogbox 185

Dictionary 86, 119

    Operationen 120

Diffusion  
     Gase 307  
 Digitaler Bilderrahmen (Projekt) 219  
 Digitaler Lichtsensor (Projekt) 291  
 Digitales Thermometer (Projekt) 264  
 Digitaluhr (Projekt) 212  
 dir() 234  
 Docstring 130  
 Drumscore 111  
 DS1820 259  
 dump() 176  
**E**  
 Editor 179  
 Einschalten 384  
 Elektrostatik 321  
 elif 70  
 else 69  
 Epoche 211  
 EPROM 174  
 Erweiterte Zuweisung 38  
 Escape-Sequenz 97  
 EVA-Prinzip 59  
 Event 218  
 Eventhandler 218  
 Event-Sequenz 218  
 except 178  
 Experiment zur Diffusion von Gasen (Projekt) 307  
 Exponentialschreibweise 27  
**F**  
 Fang das Pixel 382  
 Farbe  
     messen 298  
 Farbmischer (Projekt) 158  
 Farbwechsellampe 299  
 File 173  
     Modus 173  
 File-Manager 56  
 fill 157  
 findall() 201  
 Fische (Projekt) 313  
 Flash-Technik 174  
 float 26  
 for 90  
 Format 60  
 format() 99  
 Formatieren 99

Fotos per E-Mail verschicken (Projekt) 344  
 Freie Literatur 199  
 frozenset 114  
 Fünf Sekunden stoppen und gewinnen (Projekt) 209  
 Funktion 125  
     Aufruf per Knopfdruck 143  
     Definition 128  
     Kopf 128  
     Körper 128  
     mathematische 28  
     Parameter 125  
     Parameterübergabe 131  
     voreingestellter Parameterwert 133  
 Funktionskopf 128  
 Funktionskörper 128  
**G**  
 Gas  
     Diffusion 307  
 Gate 47  
 Gerät  
     steuern 39  
 Gerichtete Bewegungen (Projekt) 330  
 getcwd() 220  
 getPixel() 222  
 Gleitkommazahl 26  
 Glückskeks (Projekt) 162  
 gmtime() 211  
 Goethe oder Schiller (Projekt) 194  
 GPIO 40, 75, 389  
 gpiozero 41  
 Grafische Benutzungsoberfläche 151  
 Grafische Oberfläche  
     Raspberry Pi OS 385  
 Grid-Layout 156, 164  
     Optionen 165  
 GUI 151  
**H**  
 Hardware  
     Ausstattung 383  
 Heißeste Stelle im Raum 318  
 Hexadezimalsystem 26  
 Hilfe 32  
 Holzklammer 76  
 Hotkey  
     Tastenkombination 21  
 HTML 354

**I**

I2C-Bus 295  
 IDLE 18  
 if 67  
 ifconfig 349  
 ImageDraw 362  
 ImageFont 365  
 Indent Region 68  
 Index 86, 93  
 Indizierung 93  
 input() 61  
 Instanz 103  
 int 24  
 Interaktivität 151  
 Interpolation  
   lineare 289  
 Item 87, 119, 244  
   Koordinaten 245  
 Iteration 90  
 Iterierbar 88

**J**

Joystick 376  
 Jumperkabel 39

**K**

Kamera 321  
 Kameramodul 321  
 Karteikasten (Projekt) 178  
 Klasse 41, 103, 233, 235  
 Klassendefinition 233, 236  
 Klopfschalter (Projekt) 111  
 Kohlendioxid-Sensor 304  
 Kollektion  
   Operationen 87  
 Kommentar 61  
 Konkatenation 92  
 Kontrollvariable 160  
 Kopie 103  
 Kundenberatung (Projekt) 89

**L**

Label 154  
 Laden 174  
 Layout-Management 156  
 Layout-Manager 151  
 LED (Projekt) 39  
 LED-Matrix 371  
 LED-Objekt 41  
 LED-Strip 47, 231

Lesbare Zufallspasswörter (Projekt) 94

Lichtsensordaten  
   digitaler 291  
 listdir () 220  
 Liste 86, 101  
   Operationen 104  
 Literal 24, 86  
 Literatur  
   freie 199  
 load() 177  
 localtime() 212  
 Lostrommel (Projekt) 152  
 LXTerminal. 387

**M**

MAC-Adresse 349  
 Magic Line 57, 152  
 Magische Methode 234, 338  
 Mathematische Funktion 28  
 Matplotlib 301  
 Menge 86, 114  
 Messagebox 186  
 Messwerterfassung 265  
 Methode 42, 102  
   magische 234, 338  
 MIME 269  
 Mobile Stromversorgung 355  
 Modul 29  
 Modulo 23  
 Morsen (Projekt) 121  
 MOSFET 46  
 MPC3008 275  
 Murmellabyrinth (Projekt) 373

**N**

Name 33

**O**

Oberfläche  
   grafische 385  
 Objekt 233, 235  
   änderbar/nicht-änderbar 235  
   Wahrheitswert 72  
 Objektorientierte Programmierung 233  
 Objektorientierung 41  
 Oktalsystem 25  
 OLED-Display 359  
 open() 173  
 Operator 22  
   Plus 235  
   Vergleichsoperator 65

- os 220
- os.path 220
- P**
- Packer 156
- Parameter 28
  - beliebige Anzahl 134
- Parameterliste 128
- Parameterübergabe 131, 132
- Parameterwert
  - voreingestellter 133
- Passwort 94
- pickle 176
- PIL
  - ImageDraw 362
  - ImageFont 365
- PIL.Image
  - crop() 222
  - load() 222
  - resize() 223
  - save() 223
  - size 223
- Pin-Nummer 41
- pip 292
- Placer 156
- Plotter (Projekt) 270
- Plusoperator 235
- Polymorphie 338
- Pong (Projekt) 242
- Port 352
- Positionsargument 126
- Potenz 23
- Potenzimeter 275
- print() 62
  - Zeilenwechsel unterdrücken 63
- Programm
  - starten 55
- Programmierung
  - objektorientierte 233
- Programmverzweigung 64
- Projekt
  - 100 Farben 165
  - Abstandsmessungen 312
  - Alarmanlage 79
  - Alkoholgehalt messen 284
  - Alkoholisches Getränk 281
  - Alphabet 236
  - Bewegung erfassen 326
  - Birnen oder Tomaten 337
  - Blinken 84
  - Datenlogger 305
  - Digitaler Bilderrahmen 219
  - digitaler Lichtsensor 291
  - Digitales Thermometer 264
  - Digitaluhr 212
  - Experiment zur Diffusion von Gasen 307
  - Farbmischer 158
  - Fische 313
  - Fotos per E-Mail verschicken 344
  - Fünf Sekunden stoppen und gewinnen 209
  - Gerichtete Bewegungen 330
  - Glückskeks 162
  - Goethe oder Schiller 194
  - Karteikasten 178
  - Klangmaschine 144
  - Klopfzeichen 111
  - Kundenberatung 89
  - LED 39
  - Lesbare Zufallspasswörter 94
  - Lostrommel 152
  - Morsen 121
  - Murmellabyrinth 373
  - Plotter 270
  - Pong 242
  - Quicksort 142
  - Rekursive Summe 141
  - Schiller 194
  - Sichere Kommunikation 89
  - Staumelder 202
  - Taschenlampe an- und ausschalten 44
  - Telefonliste 107
  - Thermometer 366
  - Überwachungskamera 324
  - Urlaubsgrüße 160
  - Wahrnehmungstest 225
  - Webcam 355
  - Zahlenraten 73
  - Zufallsnamen 106
  - Zufallssounds 115
- Prozedur 135
- Prozess 213
- PSF 17
- pygame 116
- pygame.mixer 117
- Python 17
  - Shell 20
- Python Imaging Library (PIL) 221
- Python Software Foundation 17

**Q**

Quicksort (Projekt) 142

**R**

Radiobutton 160

range() 109

Raspberry Pi OS 385

Reaktionstest 171

Reed-Schalter 80

Regulärer Ausdruck 199

Rekursive Summe (Projekt) 141

rename () 220

RGB-Code 158

Rossum, Guido van 17

rpicam-still 322, 323

rpicam-vid 323

RTC 213

run module 55

**S**

Schalter 76, 243

Schicksalsknopf 170

Schiller oder Goethe (Projekt) 194

Schleife 72

Schlüssel 120

Schlüsselwort-Argument 63, 126

Schriftart 365

SD-Karte 174

Seiteneffekt 140

self 238

Sense HAT 368

Sensor 259

Sequenz 86, 92

Index 93

Indizierung 93

Slicing 93

Vervielfältigung 92

set 114

Shebang 58

Shell 19

Shell-Fenster 55

Shortcuts

Tastenkombination 55

Sichere Kommunikation (Projekt) 89

side 157

Skript 53

sleep() 212

Slicing 93

smbus 294

Sound 116

Spannung messen 275

Speichern 173

SPI-Protokoll 277

SSD1306 360

start\_new\_thread() 215

Starten 384

Staumelder (Projekt) 202

Steckplatine 39

String 97

Methoden 194

StringIO 331

Stromversorgung

mobile 355

subprocess 331

Syntax-Highlighting 54

**T**Taschenlampe an- und ausschalten (Projekt)  
44

Taschenrechner 22

Tastenkombination 21, 55

Telefonliste (Projekt) 107

Temperaturdaten

E-Mail 268

Temperatursensor 259

Temperatur-Zeitdiagramm 270

Term 22

Thermoelement 259

Thermometer (Projekt) 366

Thread 213

Tick 250

time 211

time() 212

tkinter 151

tkinter.filedialog 186

try 177

Tupel 87, 96

Typ 103

type() 233

Typhierarchie 85

**U**

Überladen 338

Überwachungskamera (Projekt) 324

Ultraschall 310

UML-Klassendiagramm 239

URL 192

Urlaubsgrüße (Projekt) 160

urllib.request 192

urlopen() 192

US-020 310

## V

venv 292  
Verdünnungsreihe  
  lineare 287  
Vererbung 235, 241  
Vergleichsoperator 65  
Verzweigung 67  
  einseitig 67  
  zweiseitig 69  
virtuelle Umgebung 292  
Vollbildmodus 168, 215  
Voreingestellter Parameterwert 133

## W

Wahrnehmungstest (Projekt) 225  
Wear leveling 175  
Webcam (Projekt) 355  
Webserver 349  
while 72  
Widerstand 39  
Widget 151, 154  
  Aussehen 155

Wiederholung 72  
WiFi-Adapter 349  
WLAN 349  
write() 173

## Z

Zahl 24  
Zahlenraten (Projekt) 73  
Zahlenschloss 230  
Zähler 78  
Zeichenkette 97  
Zeit 209  
Zeiteinstellung 230  
Zeit-Objekt 211  
Zufallsfunktion 105  
Zufallsnamen (Projekt) 106  
Zufallssounds (Projekt) 115  
Zusicherung 137  
Zuweisung  
  erweiterte 38  
Zuweisungsoperator 33