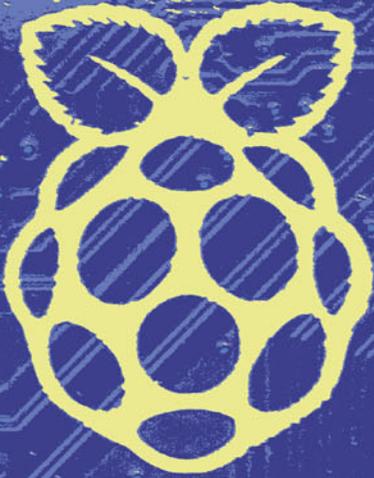




mitp



GLOBAL_EN

J2

FCC ID: 2ABCB-R

IC: 20953-RPI4B

Michael
Weigend

5. Auflage

Raspberry Pi

programmieren
mit **Python**

Inhaltsverzeichnis

	Einleitung	15
1	Begegnung mit Python	19
1.1	Was ist Python?	19
1.2	Python-Versionen	20
1.3	IDLE	21
1.3.1	Die Python-Shell	22
1.3.2	Hotkeys	23
1.4	Die Python-Shell als Taschenrechner	24
1.4.1	Operatoren und Terme	24
1.4.2	Zahlen	26
1.4.3	Mathematische Funktionen	30
1.5	Hilfe	35
1.6	Namen und Zuweisungen	36
1.6.1	Zuweisungen für mehrere Variablen	37
1.6.2	Rechnen mit Variablen in der Shell	38
1.6.3	Syntaxregeln für Bezeichner	39
1.6.4	Neue Namen für Funktionen und andere Objekte	40
1.6.5	Erweiterte Zuweisungen	40
1.7	Mit Python-Befehlen Geräte steuern	41
1.7.1	Projekt: Eine LED ein- und ausschalten	41
1.7.2	Das Modul RPI.GPIO	43
1.7.3	Steuern mit Relais	44
1.7.4	Projekt: Eine Taschenlampe an- und ausschalten	45
1.8	Aufgaben	47
1.9	Lösungen	50
2	Python-Skripte	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	54
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.	57
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	63
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	71
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.	74
2.6.1	Blinklicht	75
2.6.2	Schalter	75
2.6.3	Zähler	77
2.6.4	Grundzustände festlegen und Flankensteuerung	78
2.7	Projekt: Eine Alarmanlage.	79
2.7.1	Aufbau und Arbeitsweise der Alarmanlage	79
2.7.2	Programmierung	80
2.8	Aufgaben	83
2.9	Lösungen	85
3	Kollektionen: Mengen, Listen, Tupel und Dictionaries	89
3.1	Die Typhierarchie	89
3.2	Gemeinsame Operationen für Kollektionen	91
3.3	Kollektionen in Bedingungen.	92
3.3.1	Projekt: Kundenberatung.	93
3.3.2	Projekt: Sichere Kommunikation	93
3.4	Iteration – die for-Anweisung.	94
3.4.1	Verwendung von break	95

3.5	Sequenzen	96
3.5.1	Konkatenation und Vervielfältigung	96
3.5.2	Direkter Zugriff auf Elemente – Indizierung	97
3.5.3	Slicing	97
3.5.4	Projekt: Lesbare Zufallspasswörter	98
3.6	Tupel	100
3.7	Zeichenketten (Strings)	101
3.7.1	Strings durch Bytestrings codieren	102
3.7.2	Formatieren	103
3.8	Listen	105
3.8.1	Listen sind Objekte und empfangen Botschaften	105
3.8.2	Klasse, Typ und Instanz	106
3.8.3	Kopie oder Alias?	107
3.8.4	Listenoperationen	108
3.8.5	Projekt: Zufallsnamen	109
3.8.6	Projekt: Telefonliste	110
3.8.7	Listen durch Comprehensions erzeugen	111
3.9	Zahlen in einer Folge – range()-Funktion	112
3.10	Projekt: Klopfzeichen	114
3.11	Mengen	117
3.11.1	Projekt: Häufigkeit von Buchstaben in einem Text	118
3.12	Projekt: Zufallssounds	119
3.12.1	Wie kommen Töne aus dem Raspberry Pi?	119
3.12.2	Sounds mit PyGame	121
3.12.3	Programmierung	122
3.13	Dictionaries	123
3.13.1	Operationen für Dictionaries	125
3.13.2	Projekt: Morsen	126
3.14	Projekt: Der kürzeste Weg zum Ziel	127
3.15	Aufgaben	130
3.16	Lösungen	132
4	Funktionen	135
4.1	Aufruf von Funktionen	135
4.1.1	Unterschiedliche Anzahl von Argumenten	136
4.1.2	Positionsargumente und Schlüsselwort-Argumente	136
4.1.3	Für Experten: Funktionen als Argumente	137
4.2	Definition von Funktionen	138
4.3	Funktionen in der IDLE-Shell testen	140

4.4	Docstrings	140
4.5	Veränderliche und unveränderliche Objekte als Parameter	141
4.6	Voreingestellte Parameterwerte	143
4.7	Beliebige Anzahl von Parametern	144
4.8	Die return-Anweisung unter der Lupe	145
4.9	Mehr Sicherheit! Vorbedingungen testen	147
4.10	Namensräume: Global und lokal	149
4.11	Rekursive Funktionen – die Hohe Schule der Algorithmik	151
	4.11.1 Projekt: Rekursive Summe	151
	4.11.2 Projekt: Quicksort	152
4.12	Experimente zur Rekursion mit der Turtle-Grafik	153
	4.12.1 Turtle-Befehle im interaktiven Modus	154
	4.12.2 Projekt: Eine rekursive Spirale aus Quadraten	155
	4.12.3 Projekt: Pythagorasbaum	157
	4.12.4 Projekt: Eine Koch-Schneeflocke	159
4.13	Projekt: Der Sierpinski-Teppich	161
4.14	Funktionen per Knopfdruck aufrufen: Callback-Funktionen	163
	4.14.1 Projekt: Digitaler Türgong	164
	4.14.2 Projekt: Verkehrszählungen – Zählen mit mehreren Knöpfen	165
4.15	Aufgaben	168
4.16	Lösungen	171
5	Fenster für den RPi – Grafische Benutzungsoberflächen	175
5.1	Wie macht man eine Benutzungsoberfläche?	175
5.2	Projekt: Die digitale Lostrommel	176
	5.2.1 Die Gestaltung der Widgets	178
	5.2.2 Das Layout-Management	179
5.3	Bilder auf Widgets	181
	5.3.1 Projekt: Ein visueller Zufallsgenerator	182
	5.3.2 Bilder verarbeiten	183
	5.3.3 Projekt: Schwarz-Weiß-Malerei	185
5.4	Projekt: Der Krimiautomat	186
	5.4.1 Texteingabe	187
	5.4.2 Programmierung	188
5.5	Wer die Wahl hat, hat die Qual: Checkbutton und Radiobutton	189
	5.5.1 Projekt: Automatische Urlaubsgrüße	190
	5.5.2 Projekt: Digitaler Glückskeks	192

5.6	Viele Widgets schnell platziert: Das Grid-Layout	194
5.6.1	Projekt: Rechenquizz	195
5.7	Projekt: Farbmixer	198
5.8	Projekt: Editor mit Pulldown-Menüs	200
5.8.1	Aufbau einer Menüstruktur	201
5.8.2	Programmierung	202
5.9	Aufgaben	204
5.10	Lösungen	206
6	Daten finden, laden und speichern	211
6.1	Dateien	211
6.1.1	Daten speichern	211
6.1.2	Daten laden	212
6.2	Ein Blick hinter die Kulissen: Die SD-Karte	212
6.3	Datenstrukturen haltbar machen mit pickle	215
6.4	Versuch und Irrtum – Mehr Zuverlässigkeit durch try-Anweisungen	216
6.5	Projekt: Karteikasten	216
6.5.1	Der Editor	217
6.5.2	Der Presenter	220
6.6	Benutzungsoberfläche zum Laden und Speichern	223
6.6.1	Dialogboxen	223
6.6.2	Erweiterung des Editors für Karteikarten	225
6.6.3	Erweiterung des Presenters	228
6.7	Daten aus dem Internet	230
6.8	Projekt: Goethe oder Schiller?	231
6.8.1	Methoden der String-Objekte	232
6.8.2	Programmierung	234
6.9	Daten finden mit regulären Ausdrücken	237
6.9.1	Reguläre Ausdrücke	237
6.9.2	Die Funktion findall()	239
6.9.3	Projekt: Staumelder	239
6.9.4	Programmierung	240
6.10	Aufgaben	243
6.11	Lösungen	244
7	Projekte mit Zeitfunktionen	247
7.1	Projekt: Fünf Sekunden stoppen und gewinnen	247
7.2	Datum und Zeit im Überblick	249

7.3	Projekt: Digitaluhr	250
7.3.1	Woher bekommt der RPi die Zeit?	250
7.3.2	Was ist ein Prozess?	251
7.3.3	Vollbildmodus.	253
7.3.4	Event-Verarbeitung.	256
7.3.5	Autostart	257
7.4	Projekt: Ein digitaler Bilderrahmen	257
7.4.1	Zugriff auf das Dateisystem: Das Modul os	258
7.4.2	Python Imaging Library (PIL)	259
7.4.3	Die Programmierung.	261
7.5	Projekt: Wahrnehmungstest	263
7.5.1	Die Programmierung.	264
7.6	Projekt: Stoppuhr mit Gong	267
7.7	Aufgaben	270
7.8	Lösungen	271
8	Objektorientierte Programmierung	277
8.1	Überall Objekte	277
8.2	Klassen und Vererbung bei Python	279
8.2.1	Einführendes Beispiel: Alphabet.	280
8.2.2	Qualitätsmerkmal Änderbarkeit	283
8.2.3	Vererbung	284
8.3	Pong revisited.	286
8.3.1	Bau eines Fußschalters	287
8.3.2	Die Klasse Canvas	289
8.3.3	Die Programmierung.	293
8.4	Renn, Lola renn!.	297
8.4.1	Vorbereitung	298
8.4.2	Struktur des Programms	298
8.4.3	Background.	300
8.4.4	Switch	300
8.4.5	Display	302
8.4.6	Clock	302
8.4.7	Die Klasse Runner	303
8.4.8	Controller	304
8.4.9	Module.	306
8.5	Aufgaben	308
8.6	Lösungen	311

9	Sensortechnik	319
9.1	Was ist ein digitaler Temperatursensor?	319
9.2	Den DS1820 anschließen	320
9.3	Temperaturdaten lesen	321
9.3.1	Temperaturdaten eines Sensors automatisch auswerten ...	323
9.4	Projekt: Ein digitales Thermometer mit mehreren Sensoren	324
9.4.1	Ein Modul für die Messwerterfassung	325
9.4.2	Die grafische Oberfläche	327
9.4.3	Temperaturdaten per E-Mail senden	328
9.5	Projekt: Ein Temperaturplotter	330
9.5.1	Temperatur-Zeitdiagramme	330
9.5.2	Programmierung	331
9.6	Projekt: Mobile Datenerfassung	334
9.6.1	Experimente mit mobiler Temperaturerfassung	336
9.6.2	Programmierung	336
9.6.3	Wiedergabe der Daten	338
9.7	Spannung messen	338
9.7.1	Das SPI-Protokoll	340
9.7.2	Programmierung	342
9.8	Alkoholsensor	344
9.8.1	Projekt: Achtung! Alkoholisches Getränk!	345
9.8.2	Projekt: Den Alkoholgehalt der Luft und von Flüssigkeiten messen	348
9.8.3	Wie kann man den Alkoholgehalt von Flüssigkeiten messen?	349
9.8.4	Messen und interpolieren	351
9.9	Projekte mit einem digitalen Lichtsensor	354
9.9.1	Das Modul tsl2561	355
9.9.2	Das Modul smbus	356
9.9.3	Mit dem Raspberry Pi Farben messen – Absorptionsspektrometer	360
9.10	Kohlendioxid-Sensor	366
9.10.1	Projekt: Datenlogger	367
9.10.2	Das Sensormodul kalibrieren	368
9.10.3	Projekt: Ein Experiment zur Diffusion von Gasen	369
9.11	Aufgaben	372
9.12	Lösungen	374

10	Projekte mit der Kamera	383
10.1	Das Kameramodul anschließen	383
10.2	Die Kamerasoftware	384
	10.2.1 Einzelbilder	386
10.3	Projekt: Kameraoptionen testen	387
10.4	Projekt: Überwachungskamera – Livebild auf dem Bildschirm.	389
10.5	Projekt: Bewegung erfassen	390
10.6	Projekt: Gerichtete Bewegungen erfassen	394
	10.6.1 Files verarbeiten mit subprocess und io	395
	10.6.2 Die Programmierung	396
10.7	Projekt: Birnen oder Tomaten?	401
	10.7.1 Magische Methoden – das Überladen von Operatoren	402
	10.7.2 Programmierung	405
	10.7.3 Weiterentwicklungen	408
10.8	Das Modul picamera	408
	10.8.1 Die Klasse PiCamera	409
	10.8.2 Projekt: Einen Film aufnehmen	410
	10.8.3 Projekt: Fotos per E-Mail verschicken	411
10.9	Was ist los am Autobahnkreuz? Bilder einer Webcam auswerten.	413
	10.9.1 Webcams im Internet anzapfen	413
	10.9.2 Auf einem Foto zeichnen – das Modul ImageDraw	414
	10.9.3 Projekt: Verkehrsdichte auf der Autobahn	417
	10.9.4 Wie findet man den URL eines Webcam-Bildes?	421
10.10	Randbemerkung: Was darf man? Was soll man?	422
10.11	Aufgabe: Wie lang? Wie breit?	422
10.12	Lösung	423
II	Webserver	427
II.1	Der RPi im lokalen Netz	427
	II.1.1 WLAN	427
	II.1.2 Eine dauerhafte IP-Adresse für den RPi	428
	II.1.3 Über SSH auf dem RPi arbeiten	429
	II.1.4 Virtual Network Computing (VNC)	429
II.2	Webserver	431
	II.2.1 Der Apache-Webserver	431
	II.2.2 Ein Webserver mit Python	433
	II.2.3 Die Startseite	434
	II.2.4 Den Server testen	435

II.3	Was ist los im Gartenteich?	435
II.3.1	Projekt: Einfache Webcam mit statischer Webseite	435
II.3.2	CGI-Skripte	439
II.3.3	CGI-Skripte für den Apache-Server	442
II.3.4	Hilfe, mein CGI-Skript läuft nicht!	443
II.3.5	Interaktive Webseiten	444
II.3.6	Eingabekomponenten in einem HTML-Formular	446
II.3.7	Verarbeitung von Eingaben in einem CGI-Skript	447
II.3.8	Zugriff aus der Ferne	449
II.4	Geräte über das Internet steuern	450
II.4.1	Privilegierte Rechte für ein CGI-Skript	450
II.4.2	Programmierung	451
II.5	Datenbanken	453
II.5.1	Das Modul sqlite3	453
II.5.2	Projekt: Freies Obst	456
II.5.3	Alternativen zu CGI	463
II.6	Aufgaben	464
II.7	Lösungen	465
12	Erweiterungen: OLED und HAT	471
12.1	OLED-Display	471
12.2	Anschluss	471
12.3	Installation der SSD1306-Bibliothek	472
12.4	Auf dem Display Texte und Formen ausgeben	472
12.4.1	Projekt: Uhrzeit	473
12.5	Mit ImageDraw zeichnen und Texte schreiben	474
12.5.1	Grafiken zeichnen	475
12.5.2	Schriftarten definieren	477
12.6	Projekt: Gespeicherte TrueType-Fonts darstellen	478
12.7	Projekt: Thermometer	480
12.8	Sense HAT	481
12.9	Die Klasse SenseHat	483
12.10	Grafische Ausgabe über die LED-Matrix	485
12.11	Die räumliche Orientierung des Sense HAT	486
12.11.1	Projekt: Das Murmellabyrinth	487
12.12	Der Joystick	490
12.12.1	InputEvent	491
12.12.2	Die Klasse JoyStick	492

12.12.3	Definition von Eventhandlern	492
12.12.4	Auf Events warten und Events abfragen	494
12.13	Aufgaben	496
12.14	Lösungen	497
A	Den Raspberry Pi einrichten	501
A.1	Hardware-Ausstattung	501
A.2	Das Betriebssystem installieren	501
A.3	Raspberry Pi das erste Mal starten und konfigurieren	502
A.4	Die grafische Oberfläche von Raspberry Pi OS	503
B	Der GPIO	507
B.1	Pinbelegung und Funktionen	507
B.2	Ein Flachbandkabel mit Pfostenverbindern	510
C	Autostart	513
D	So entstand das Titelbild	515
	Stichwortverzeichnis	519



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. 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, was angesprochen wird. 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. Zweitens soll es inspirieren, das gelernte Programmierwissen in konkrete Projekte einfließen zu lassen. Damit die Beschreibung von Hardwaretechnik und speziellen Schnittstellen nicht ins Uferlose wächst, gehe ich von vier allgemeinen Hardwarekonfigurationen aus.

Interaktives Exponat

Auf dem Raspberry Pi läuft ein interaktives Programm mit grafischer Benutzungsoberfläche. 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. Eine einfachere Variante dieser Konfiguration ist nicht interaktiv und verwendet ein einfaches LCD-Display. 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).

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 und Relais ansteuern: z. B. Zähler und Alarmanlagen. Komplexere Projekte mit einer grafischen Oberfläche sind z. B. eine Stoppuhr, die mit einem echten Gong betrieben wird, ein Pong-Spiel, bei dem die Schläger auf dem Bildschirm mit Schaltern oder Potenziometern gesteuert werden (Kapitel 8, 9) und die Simulation eines Rennens, bei dem der Spieler wirklich seine Beine bewegen muss, damit die Figur auf dem Bildschirm ihr Ziel erreicht (Kapitel 8). 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 halb automatisch durchgeführt und ausgewertet werden (Kapitel 9).

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, 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 SSH-Verbindung von einem anderen Rechner aus gesteuert werden. In Kapitel 11 finden Sie alle Details und Beispiele für serverbasierte Projekte, darunter eine Webcam, ein System zur Steuerung von elektrischen Geräten über das Internet und ein Online-Datenbanksystem.

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.

Zum Aufbau dieses Buches

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 Buches 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 und Lösungen, 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.

Downloads im Internet

Unter der Webadresse www.mitp.de/0383 können Sie die Listings aus dem Buch kostenlos downloaden.

Begegnung mit Python

In diesem Kapitel verwenden Sie Python im interaktiven Modus. Sie geben in der Kommandozeile der Python-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 RPi abgekürzt wird. Auf der SD-Speicherkarte ist als Betriebssystem Raspbian installiert.

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 nicht-kommerzielle 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 Python-Versionen

Auf dem Raspberry Pi sind zwei Versionen von Python installiert:

- Python 2 (in der Version 2.7)
- Python 3 (in der Version 3.7).

Dieses Buch bezieht sich allein auf das modernere Python 3. Aber es ist wichtig zu wissen, dass es noch eine andere Version gibt. Python 3 ist nicht kompatibel zu Python 2. Das heißt, ein Python-3-Interpreter kann mit einem Programm, das in Python 2 geschrieben worden ist, in der Regel nichts anfangen.

Das neuere Python 3 ist konsistenter und konzeptionell »sauberer«. Das bedeutet, es gibt mehr Regelmäßigkeit und weniger Ausnahmen. Deshalb ist Python 3 vielleicht etwas leichter zu erlernen und die Programme, die in Python 3 geschrieben worden sind, sind etwas leichter zu durchschauen. Aber groß sind die Unterschiede nicht.

Beide Versionen existieren parallel. Auch Python 2 wird weiterhin gepflegt.

1.3 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 `[Enter]`. Es erscheint eine Liste von Software-Angeboten.
- Rollen Sie etwas herunter und wählen Sie DIE IDLE-Version, die zu Ihrer Python-Version passt (`idle-3.7.3`), indem Sie die Checkbox vor dem Eintrag ankreuzen.
- Klicken Sie unten rechts auf OK.
- Zur Sicherheit müssen Sie Ihr Passwort eingeben. (Wenn Sie das vorgegebene Passwort nicht geändert haben, ist es `raspberrypi`.)
- Dann wird IDLE installiert.

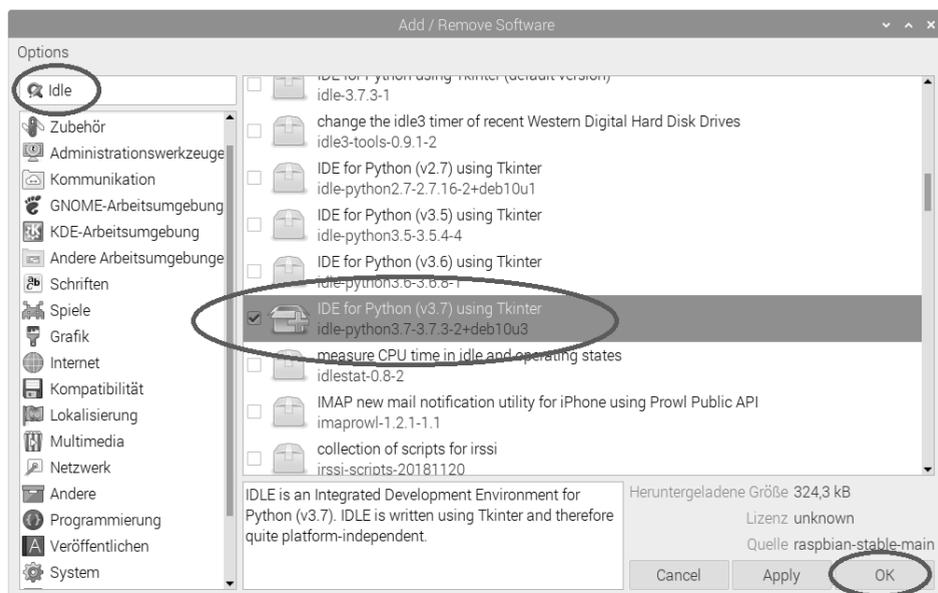


Abb. 1.1: IDLE installieren

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: PYTHON 3 (IDLE). Wenn Sie diesen Eintrag anklicken, öffnet sich die Entwicklungsumgebung.

IDLE besteht im Wesentlichen aus drei Komponenten:

- **Die Python-Shell.** Wenn Sie IDLE starten, öffnet sich zuerst das Python-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.

1.3.1 Die Python-Shell

Anweisungen sind die Bausteine von Computerprogrammen. Mit der Python-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|PYTHON 3 (IDLE)). Da Sie ständig mit IDLE arbeiten werden, lohnt es sich, das Programmicon auf den Desktop zu bringen. Das geht so: Sie klicken mit der rechten Maustaste auf das Icon PYTHON 3 und wählen im Kontextmenü den Befehl DEM DESKTOP HINZUFÜGEN. Die Python-Shell meldet sich immer mit einer kurzen Information über die Version und einigen weiteren Hinweisen.

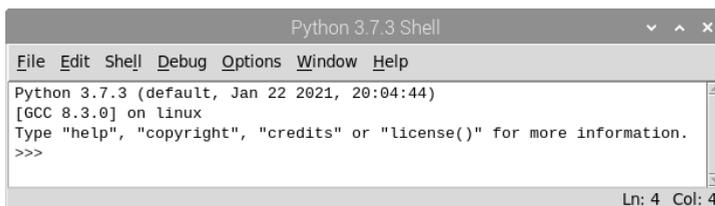


Abb. 1.2: Die Python-Shell der Entwicklungsumgebung IDLE

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 `↵` 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 den Prompt. Die gespeicherte Zahl können Sie wieder zum Vorschein bringen, indem Sie den Variablennamen eingeben:

```
>>> a
1257002
```

1.3.2 Hotkeys

Um effizient mit der Python-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 `Alt + P`. Beispiel:

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

Wenn Sie jetzt *einmal* die Tastenkombination `[Alt]+[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 weiter. 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).

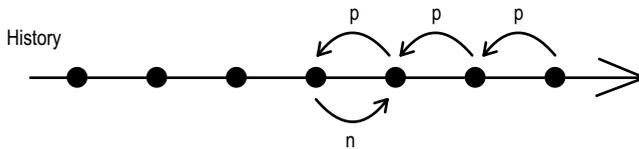


Abb. 1.3: Navigieren in der History mit `[Alt]+[P]` und `[Alt]+[N]`

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 Python-Shell

1.4 Die Python-Shell als Taschenrechner

Die Python-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.

1.4.1 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
620972443101902588551304810097687105537832218918245689182643787308016217315097
07020422858215922309341135893663853254591817
```

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

1.4.2 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 zum Beispiel 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 Python-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: $1*256 + 0*16 + 10*1 = 256 + 10 = 266$

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. Zum Beispiel 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:

$$\begin{aligned} 123000000 &= 123 \cdot 10^6 \\ 0.00012 &= 1.2 \cdot 10^{-4} \end{aligned}$$

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
```

Komplexe Zahlen (Typ `complex`)

Komplexe Zahlen werden als Summe aus einem Real- und einem Imaginärteil beschrieben:

```
c = a + b*i
```

Dabei bezeichnet der Buchstabe `i` die Wurzel aus -1 . Python verwendet (wie in der Elektrotechnik üblich) den Buchstaben `j` oder `J` anstelle von `i`, um Verwechslungen mit dem Symbol für die Stromstärke zu vermeiden. Beispiele sind:

```
0.3j
1.2e-3j
12 + 20j
```

Keine gültige komplexe Zahl ist übrigens der Buchstabe `j` alleine. Die imaginäre Zahl `i` schreiben Sie `1j`.

1.4.3 Mathematische Funktionen

Wenn Sie die Python-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:

```
>>> abs(-2, 5)
Traceback (most recent call last):
  File "<pyshell#33>", line 1, in <module>
    abs(-2, 5)
TypeError: abs() takes exactly one argument (2 given)
```

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.

Tabelle 1.3: Mathematische Standardfunktionen (müssen nicht importiert werden)

Funktion	Erklärung
<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 x_0, \dots, x_n .
<code>min(x0, ..., xn)</code>	Liefert die kleinste Zahl von x_0, \dots, x_n .
<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) (Forts.)

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 1.7).

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 pi, sqrt
>>> sqrt(4)
2.0
>>> exp(1)
Traceback (most recent call last):
  File "<pysHELL#4>", line 1, in <module>
    exp(1)
NameError: name 'exp' is not defined
```

Die Funktion `exp()` wurde nicht importiert und bleibt deswegen unbekannt.

Mathematische Funktionen und Konstanten

Wenn Sie die Python-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° entspricht dem Bogenmaß 2π .

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

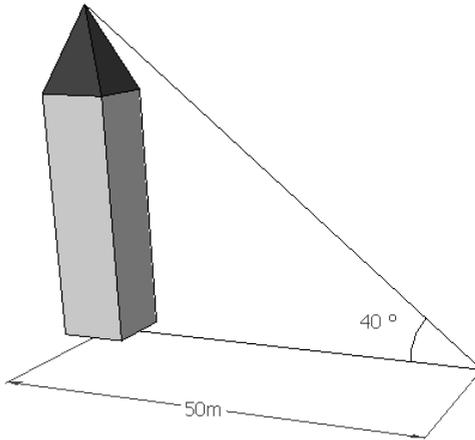


Abb. 1.4: Bestimmung der Höhe eines Turms durch Anpeilen der Turmspitze

Nun verarbeiten die trigonometrischen Funktionen ($\sin()$, $\cos()$, ...) des Moduls `math` keine Winkel (0° bis 360°), 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 $e \approx 2.7182\dots$
<code>exp(x)</code>	e^x
<code>fabs(x)</code>	Absolutwert der Gleitkommazahl x

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

Funktion/Konstante	Erklärung
<code>floor(x)</code>	Der nach unten gerundete Wert von x
<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 (0.2, 4.0).
<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 $\pi \approx 3.1415\dots$
<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` (Forts.)

1.5 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 Python-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 ...
>>>
```

1.6 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.

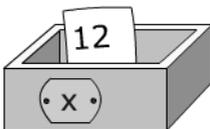


Abb. 1.5: Visualisierung einer Variablen als Behälter mit Inhalt

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*.

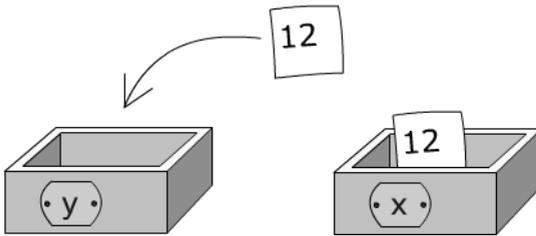


Abb. 1.6: Visualisierung einer Wertübertragung

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.

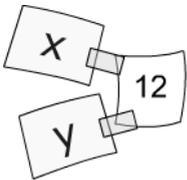


Abb. 1.7: Zuweisen als Hinzufügen eines neuen Namens

1.6.1 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
```

1.6.2 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
```

1.6.3 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 (*keyword*). 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 Python-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), 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.

1.6.4 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.

1.6.5 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.

1.7 Mit Python-Befehlen Geräte steuern

Bisher ging es um die reine Programmierung mit Python. In diesem Abschnitt soll nun erstmals die Elektronik des Raspberry Pi angesprochen werden. Mit Python-Programmen können Sie elektrische Geräte steuern. Wir fangen mit dem Einfachsten an und schalten eine LED über Python-Befehle in der Kommandozeile an und aus. Bevor es an die Programmierung geht, müssen Sie eine kleine Schaltung aufbauen und an den GPIO anschließen. Zusätzliche Informationen zu den Hardware-Grundlagen finden Sie im Anhang B.

1.7.1 Projekt: Eine LED ein- und ausschalten

Für dieses Projekt benötigen Sie

- eine Steckplatine (*Breadboard*)
- Jumperkabel (female-male)
- eine LED
- Kabelbrücken für die Steckplatine
- einen elektrischen Widerstand mit $R = 130 \text{ Ohm}$

Abbildung 1.8 zeigt, wie die Schaltung in der Realität aussieht. Die (+)-Leiterbahn (rot) der Steckplatine ist mit einem Jumperkabel mit Pin 1 des GPIO verbunden. Pin 10 ist über ein Jumperkabel mit einem Ende des Widerstands verbunden. Auf der Steckplatine sind Widerstand und die LED in Reihe geschaltet.

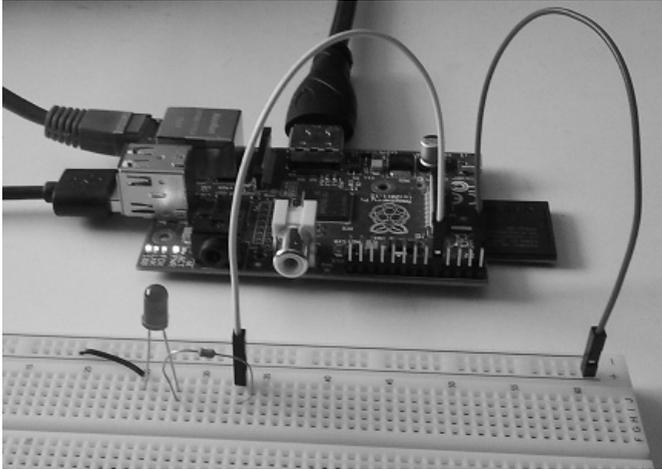


Abb. 1.8: Die Steckplatine trägt die Schaltung und ist über Jumperkabel mit dem GPIO verbunden.

Besser zu lesen ist meist ein Schaltdiagramm mit Schaltsymbolen wie in Abbildung 1.9.

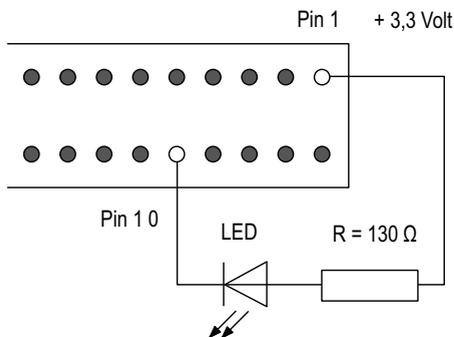


Abb. 1.9: Schaltdiagramm für den Anschluss einer LED an den GPIO des Raspberry Pi

Wenn Sie wie bei diesem Projekt elektronische Bauteile über Jumperkabel an den GPIO anschließen, sollten Sie genauestens aufpassen, nicht einen der 5-Volt-Pins zu berühren. Ein versehentlicher Kontakt kann die Elektronik Ihres RPi zerstören.

Eine Leuchtdiode (oder LED von engl. *light-emitting diode*) ist ein Halbleiterbauteil, das Licht aussendet und sich elektrisch wie eine Diode verhält. Das bedeutet insbesondere, dass LEDs nur in einer Richtung den Strom leiten. Sie wirken wie ein Ventil für Elektronen. Das Schaltsymbol ähnelt ein bisschen einem Pfeil und deutet die technische Fließrichtung des Stroms von Plus nach Minus an. Bei einer

neuen LED ist die Anode (+) etwas länger als die Kathode (-). Leuchtdioden haben nur einen äußerst geringen Widerstand und vertragen nur eine begrenzte Stromstärke. Bei den normalen LEDs, die Sie von Anzeigeleuchten in Elektrogeräten und Armaturenbletern kennen und die wir auch für Schaltungen mit dem RPi verwenden, beträgt die zulässige Maximalstromstärke etwa 20 mA (Milliampere). Um die Stromstärke zu begrenzen, muss ein Widerstand in den Stromkreis geschaltet werden. Ein guter Widerstandswert für unsere Zwecke ist $R = 130 \text{ Ohm}$. Das können Sie selbst nachrechnen: Nach dem ohmschen Gesetz gilt immer die Beziehung $R = U/I$. Nun haben wir an den steuerbaren Pins des GPIO eine Spannung von 3,3 Volt. An einer Leuchtdiode ist meist eine Spannung von 2 Volt. Damit liegt am Widerstand eine Spannung von 1,3 Volt an. Wenn wir den Strom auf 10 mA begrenzen wollen, ergibt sich für den Vorwiderstand: $R = 1,3\text{V}/0,01\text{A} = 130 \text{ }\Omega$.

1.7.2 Das Modul RPi.GPIO

Kommen wir nun zur Programmierung. Öffnen Sie eine Python-Shell (IDLE).

```
sudo idle3
```

Versuchen Sie zunächst, das Modul `Rpi.GPIO` zu importieren. Achten Sie dabei auf das kleine `i` bei `Rpi`.

```
>>> from Rpi import GPIO
```

Falls es nun eine Fehlermeldung gibt, ist das Modul noch nicht installiert. Öffnen Sie dann ein LX-Fenster und geben Sie das folgende Kommando ein:

```
sudo apt-get install python-rpi.gpio
```

Nun geht es wieder in der Python-Shell weiter. Stellen Sie den Modus der Pin-Nummerierung ein:

```
>>> GPIO.setmode(GPIO.BOARD)
```

Machen Sie Pin 10 zum Ausgabekanal.

```
>>> GPIO.setup(10, GPIO.OUT)
```

Schalten Sie die LED aus, indem Sie den Ausgang mit dem Wahrheitswert `True` belegen. Dann hat der Pin 10 den Spannungspegel von +3,3 V. Beide Anschlüsse der LED haben dann das gleiche elektrische Potenzial (+3,3 V gegenüber der Masse). Das heißt, an der LED liegt keine Spannung an und es fließt auch kein Strom.

```
>>> GPIO.output(10, True)
```

Wenn Sie den Wahrheitswert `False` an den Ausgang legen, ist am Pin 10 der Spannungspegel 0 V. Nun gibt es an der LED einen Spannungsabfall, es fließt Strom und die LED leuchtet.

```
>>> GPIO.output(10, False)
```

1.7.3 Steuern mit Relais

Die Ausgänge des GPIO können Sie mit maximal 50 mA belasten. Für die Steuerung größerer elektrischer Geräte (Motoren, Lampen) brauchen Sie Relais. Es gibt preiswerte Module (etwa ab 5 Euro) mit mehreren Relais, die Sie verwenden können. Oft werden sie für den Arduino (<http://www.arduino.cc/>) angeboten und sind deshalb auf TTL (Transistor-Transistor-Logik) zugeschnitten. Das heißt, diese Relais werden mit 5 Volt angesteuert. Zum Glück funktionieren sie auch mit 3,3 Volt, also der Spannung, die an einem programmierbaren GPIO-Ausgang anliegt, wenn er im Zustand `True` ist.

Abbildung 1.10 zeigt ein populäres Modul mit vier Relais. Eingezeichnet sind rechts die sechs Eingänge (GND, IN1, IN2, IN3, IN4, VCC). GND (Masse) verbinden Sie mit Pin 6 (GND) und VCC mit Pin 2 (+5 V) des GPIO. Die anderen Pins werden an programmierbare Pins des GPIO angeschlossen.

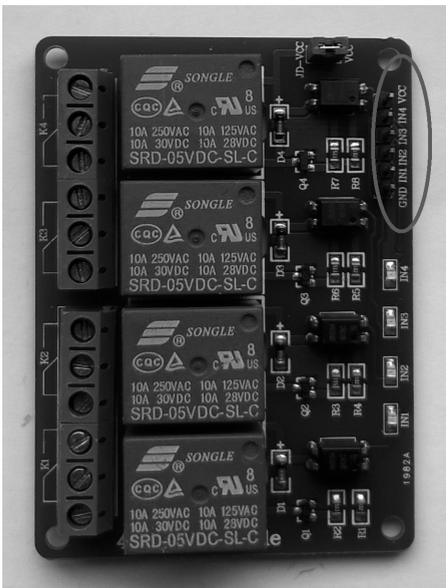


Abb. 1.10: 5-Volt-Relais-Modul für den Arduino. Es funktioniert auch mit dem RPi.

Die Relais auf dieser Platine sind vier Blöcke. Grundsätzlich ist ein Relais nichts weiter als ein elektrisch gesteuerter Schalter. Es kann einen Stromkreis öffnen und schließen. Links neben jedem Relais sind drei Schraubklemmen. Abbildung 1.11 zeigt die Arbeitsweise des Umschaltrelais. Wenn der Steuereingang des Relais im Zustand `True` ist (positiver Spannungspegel), wird der mittlere Anschluss mit dem unteren verbunden. Das ist übrigens der Grundzustand, in dem sich das Relais befindet, wenn am Eingang nichts angeschlossen ist. Meist ist an dem Relais eine kleine Grafik, die den Grundzustand anzeigt. Wenn der Steuereingang den Zustand `False` hat (0 V), wird der mittlere Anschluss mit dem oberen verbunden. Mit dem Umschaltrelais können Sie also zwei Stromkreise gleichzeitig steuern. Wenn der eine geöffnet ist, ist der andere geschlossen.

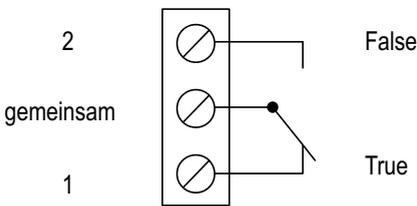


Abb. 1.11: Schaltung eines Umschaltrelais

Auf der Oberseite der Relais finden Sie Angaben zur maximalen Belastbarkeit. Bei den Relais in Abbildung 1.11 sind 10 A und 250 V zugelassen.

1.7.4 Projekt: Eine Taschenlampe an- und ausschalten

In diesem Abschnitt wird erklärt, wie man mit Alltagsmaterialien (Kabel, Pappe, Alufolie, Kreppklebeband und einem Gummiband) eine Taschenlampe mit einem Relais steuern kann. Es gibt natürlich verschiedene Typen von Taschenlampen. Diese Anleitung bezieht sich auf eine Taschenlampe mit Metallgehäuse. Der Schalter sitzt auf der Rückseite. Wenn Sie eine andere Art von Taschenlampe verwenden, müssen Sie etwas anderes erfinden. Abbildung 1.12 zeigt das Endergebnis. Sie erkennen, dass an das erste Relais zwei Kabel angeschlossen sind, die zur Taschenlampe führen.

Und so gehen Sie vor: Schrauben Sie die Verschlusskappe des Batteriefachs der Taschenlampe ab. Im Inneren sehen Sie einen Pol des Batteriesets. Wenn man ihn mit der Metallhülle der Taschenlampe verbindet, leuchten die LEDs.

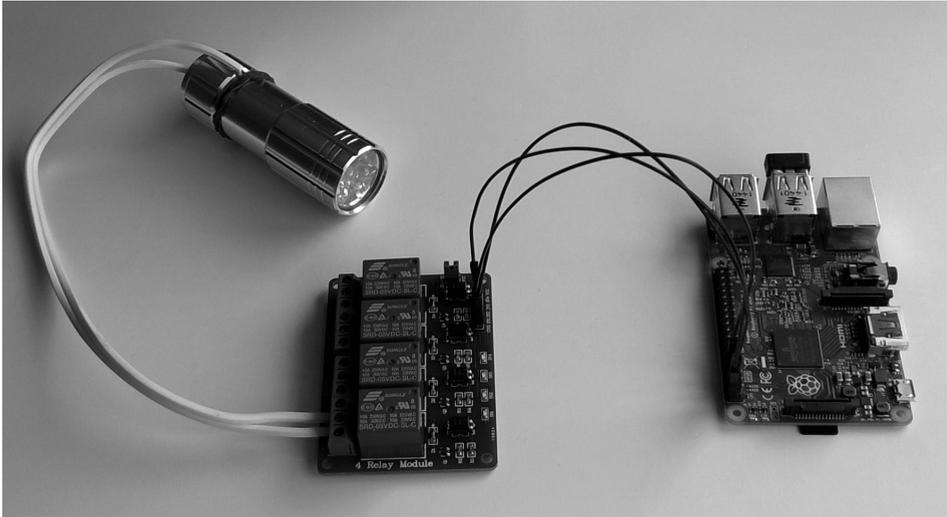


Abb. 1.12: Steuerung einer Taschenlampe mit einem Relais

Sie brauchen ein Kabel mit zwei Adern. Entfernen Sie am Ende einer Ader etwa 8 cm Isolierung. Wickeln Sie um die Mitte des entisolierten Bereichs etwas Alufolie, sodass dort eine Verdickung entsteht (Abbildung 1.13, linkes Bild).

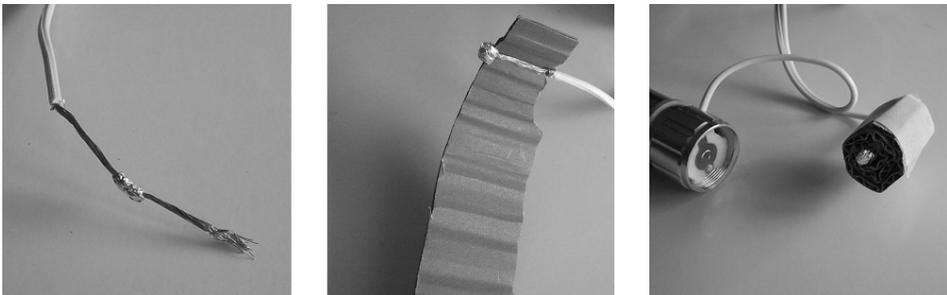


Abb. 1.13: Anschluss eines Steuerkabels an die Taschenlampe

Schneiden Sie aus Wellpappe einen etwa 3 cm schmalen Streifen aus und wickeln Sie ihn an einer Seite um das entisolerte Kabelende, sodass die Verdickung aus Alufolie an einer Kante der Pappe hervorragt (Abbildung 1.13, mittleres Bild). Sorgen Sie dafür, dass die Metallitze fest verdreht ist.

Rollen Sie die Pappe zu einem Zylinder zusammen. An der einen Seite ragt in der Mitte die Verdickung aus Aluminiumfolie hervor, auf der anderen Seite kommt das Kabel heraus (Abbildung 1.13, rechtes Bild).

Mit dem Gummiband befestigen Sie die zweite Ader an dem Metallgehäuse der Taschenlampe. Eventuell müssen Sie das Metall an der Berührstelle etwas ankratzen, damit es besser leitet.

Dann schließen Sie die beiden anderen Enden des Kabels wie in Abbildung 1.14 an das erste Relais an. Im Grundzustand sollte der Stromkreis mit der Lampe geöffnet sein. Verbinden Sie mit Jumperkabeln die Eingänge GND, IN1 und VCC des Moduls mit dem GPIO wie in Abbildung 1.14 rechts.

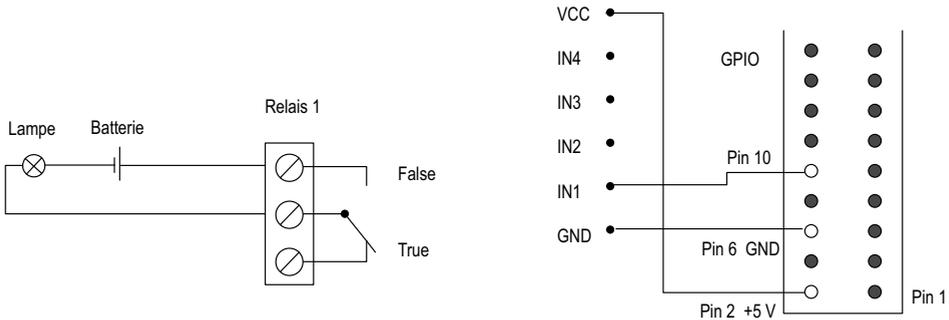


Abb. 1.14: Schaltplan für die Steuerung mit einem Umschaltrelais

Testen Sie Ihren Aufbau mit Python-Befehlen:

```
>>> from RPi import GPIO
>>> GPIO.setmode(GPIO.BOARD)
>>> GPIO.setup(10, GPIO.OUT) # Pin 10 ist Ausgang
>>> GPIO.output(10, False) # Lampe einschalten
>>> GPIO.output(10, True) # Lampe ausschalten
```

1.8 Aufgaben

Aufgabe 1: Formeln

Berechnen Sie die drei handschriftlichen Formeln in der Python-Shell.

$$(1 + e)^{97} \quad \sqrt{\frac{23}{14}} \quad \frac{2^8 \cdot 91^{12}}{11}$$

Abb. 1.15: Drei Formeln in mathematischer Schreibweise

Aufgabe 2: Gebäude

Berechnen Sie in der Python-Shell das Volumen des Gebäudes in Abbildung 1.16.

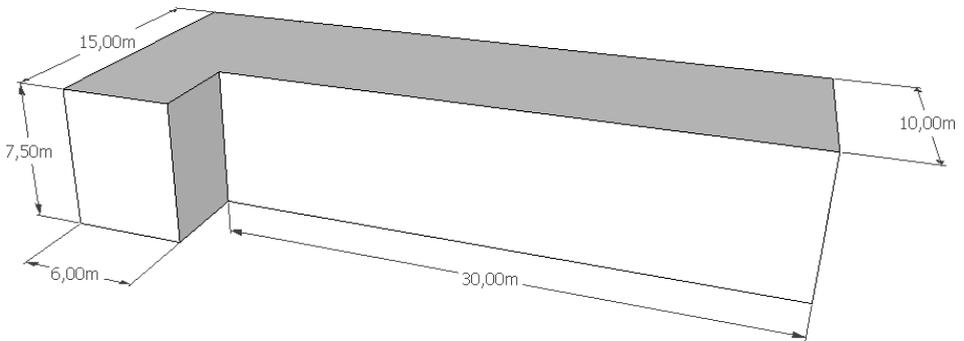


Abb. 1.16: Wohnblock

Aufgabe 3: Zylinder

Berechnen Sie in der Python-Shell das Volumen des Zylinders aus Abbildung 1.17.

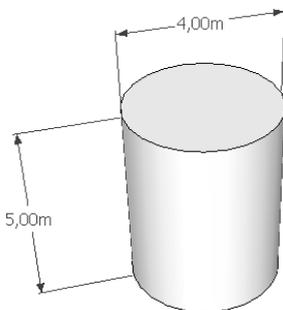


Abb. 1.17: Zylinder

Aufgabe 4: Anweisungen

Welche Ergebnisse liefern folgende Anweisungen?

```
>>> float(9)
>>> int(7/2)
>>> 0b100 + 0b100
>>> bin(2**4)
>>> bin(2**5 + 2**0)
>>> 10*10/10
>>> 2/2
>>> 3**2
>>> 9**0.5
```

Aufgabe 5: Visualisierungen interpretieren

Formulieren Sie zu den drei Bildern in Abbildung 1.18 passende Python-Anweisungen.

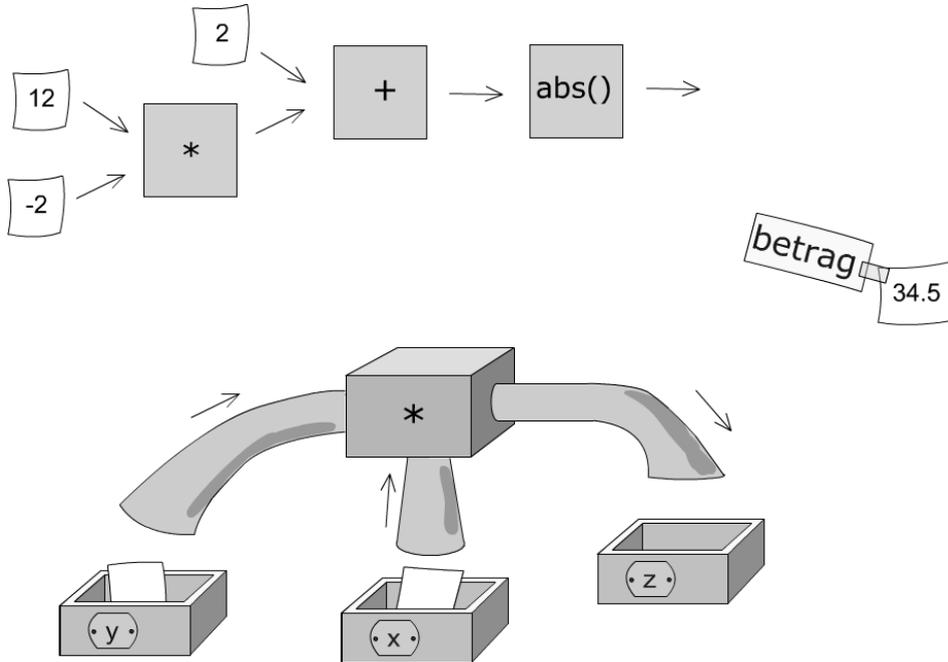


Abb. 1.18: Visualisierungen von Python-Anweisungen

Aufgabe 6

Welche der folgenden Wörter sind nicht als Bezeichner zulässig?

```
name
Erdumfang
ErdUmfang
False
false
round
x(1)
x1
```

1.9 Lösungen

Lösung 1

```
>>> from math import *
>>> (1 + e)**97
2.1047644793816324e+55

>>> sqrt(23/14)
1.2817398889233114

>>> (2**8 * 91**12)/pi
2.6277666738095865e+25
```

Lösung 2

```
>>> (30*10 + 6*15)*7.5
2925.0
```

Lösung 3

```
>>> pi*(4/2)**2 * 5
62.83185307179586
```

Lösung 4

```
>>> float(9)
9.0
```

```
>>> int(7/2)
3
```

Kommentar: $7/2$ ergibt 3,5, die `int()`-Funktion lässt die Nachkommastellen weg.

```
>>> 0b100 + 0b100
8
```

Kommentar: Die Summe zweier Binärzahlen wird als Dezimalzahl ausgegeben.

```
>>> bin(2**4)
'0b10000'
bin(2**5 + 2**0)
'0b100001'
```

Kommentar: An den Beispielen erkennt man den Aufbau von Binärzahlen als Summe von Zweierpotenzen.

```
>>> 10*10/10
10.0
>>> 2/2
1.0
>>> 3**2
9
>>> 9**0.5
3.0
```

Kommentar: Division / und Potenzieren ** mit einer Gleitkommazahl liefern immer ein Objekt vom Typ float, auch wenn – rein mathematisch – eine ganze Zahl herauskommt.

Lösung 5

```
abs(12*-2 + 2)
betrag = 34.5
z = y * x
```

Lösung 6

Nicht erlaubt sind False (Schlüsselwort) und x(1) (Klammern sind nicht erlaubt). Dagegen ist das Wort false als Name erlaubt, da es mit einem kleinen f beginnt und sich somit von dem Schlüsselwort False unterscheidet.

Stichwortverzeichnis

Numerisch

I-Wire-Bus 319

A

Absorptionsdiagramm 360

Absorptionsspektrometer 360

action 447

add_cascade() 201

add_checkbutton() 201

add_command() 201

add_radiobutton() 201

add_separator() 201

Aggregat 283

Aggregation 283

Akku 335

Aktueller Parameter 135

Alarmanlage (Projekt) 79

Alias 107

Alkoholgehalt messen (Projekt) 348

Alkoholisches Getränk (Projekt) 345

Alkoholsensor 344

Alphabet (Projekt) 280

anchor 181

Animierte Uhr 496

Anweisungsblock 138

Apache 431

Apache-Server

anhalten 432

starten 432

Application Launch Bar 503

Argument 31, 135

asctime() 249

Astro Pi 482

Ausdruck 24

regulärer 237

Ausführbar machen 440

Autosimulator 373

Autostart 257, 513

B

Background 300

backward() 155

Bedingung 64

Benutzungsoberfläche

grafische 175

Betriebssystem

installieren 501

Bewegung erfassen (Projekt) 390

Bezeichner 39

Syntaxregeln 39

Bildrate 411

Binärmodus 215

BINARY(n) 454

bind() 256

Birnen oder Tomaten (Projekt) 401

Blinken (Projekt) 85

Blinklicht 75

Body-Mass-Index 68

Boolescher Ausdruck 83

Botschaft 105

Breadboard 41

break 74, 95

Built-in function 30, 135

Bytestring 102

C

Callback-Funktion 163

Canvas (Klasse) 289

Methoden 292

Canvas (Objekt) 289

Casting 92

CGI-Skript 431, 439

ausführbar machen 440

Debugging 443

Probleme 443

cglib 443

Channel 121

chdir () 258

check 446

Checkbox 189

Clock 302

close() 211

Colorimetrie 408

Comprehension 111

CREATE TABLE 454

crop() 260

CSI-Verbinder 384

D

Dämon 433

Datei 211

Datenbank 453

relationale 453

Datenlogger (Projekt) 367

def 138

Dialogbox 223

Dictionary 90, 123

Operationen 125

Diffusion

Gase 369

Digitaler Bilderrahmen (Projekt) 257

Digitaler Lichtsensor (Projekt) 354

Digitaler Türgong (Projekt) 164

Digitales Thermometer (Projekt) 324

Digitaluhr (Projekt) 250

dir() 278

Display 302

Docstring 140

Drumscore 115

DS1820 319

dump() 215

Dynamic Domain Name System 450

E

Editor 217

Editor mit Pulldown-Menüs (Projekt) 200

Einschalten 502

Einzelbild 386

Elektrostatik 383

elif 70

else 69

Entry 187

Epoche 249

EPROM 212

Erweiterte Zuweisung 40

Escape-Sequenz 101

EVA-Prinzip 59

Event 256

Eventhandler 256

Event-Sequenz 256

except 216

Experiment zur Diffusion von Gasen (Projekt) 369

Exponentialschreibweise 29

F

Fallende Flanke 78

Fang das Pixel 496

Farbe

messen 360

Farbmixer (Projekt) 198

Farbwechsellampe 360

FieldStorage 447

File 211

Modus 211

File-Manager 57

fill 181

findall() 239

Flachbandkabel 510

Flanke

fallende 78

steigende 78

Flankensteuerung 78

Flash-Technik 212

float 28

for 94

Format 60

format() 103

Formatieren 103

Formular 445

Fotos per E-Mail verschicken (Projekt) 411

Freie Literatur 237

Freies Obst (Projekt) 456

frozenset 117

Fünf Sekunden stoppen und gewinnen (Projekt) 247

Funktion 135

Aufruf per Knopfdruck 163

Definition 138

Kopf 138

Körper 138

mathematische 30

Parameter 135

Parameterübergabe 141

voreingestellter Parameterwert 143

Funktionskopf 138

Funktionskörper 138

Fußschalter 287

G

Gas

Diffusion 369

Gerät

steuern 41

Gerichtete Bewegungen (Projekt) 394

getcwd() 258

Get-Methode 445

getPixel() 261
 Gleitkommazahl 28
 Glückskeks (Projekt) 192
 gmtime() 249
 Goethe oder Schiller (Projekt) 231
 GPIO 42, 74, 507
 Grafische Benutzungsoberfläche 175
 Grafische Oberfläche
 Raspberry Pi OS 503
 Graph 127
 Grid-Layout 180, 194
 Optionen 195
 GUI 175

H

Hangman 204
 Hardware
 Ausstattung 501
 Häufigkeit von Buchstaben (Projekt) 118
 Heißeste Stelle im Raum 372
 Hexadezimalsystem 28
 hidden 446
 Hilfe 35
 Holzklammer 75
 Hotkey siehe Tastenkombination
 HTML 434

I

IDLE 22
 if 67
 ifconfig 427
 ImageDraw 474
 ImageFont 477
 Indent Region 68
 Index 90, 97
 Indizierung 97
 input() 61
 INSERT INTO 454
 Instanz 106
 int 26
 Interaktive Webseite 444
 Interpolation
 lineare 352
 IP-Nummer 427
 Item 91, 124, 289
 Koordinaten 289
 Iteration 94
 Iterierbar 92

J

Joystick 490
 Jumperkabel 41

K

Kamera 383
 Kameramodul 383
 Kameraoptionen testen (Projekt) 387
 Karteikasten (Projekt) 216
 Klasse 106, 277, 279
 Klassendefinition 277, 279
 Klopffzeichen (Projekt) 114
 Knoten 128
 Koch-Schneeflocke (Projekt) 159
 Kohlendioxid-Sensor 366
 Kollektion
 Operationen 91
 Kommentar 61
 Komplexe Zahl 30
 Konkatenation 96
 Kontrollvariable 189, 190
 Kopie 107
 Krimiautomat (Projekt) 186
 Küchentablett 334
 Kundenberatung (Projekt) 93
 Kürzester Weg (Projekt) 127

L

Label 178
 Laden 212
 Lauf-Simulation (Projekt) 297
 Layout-Management 179
 LED (Projekt) 41
 LED-Matrix 485
 Lesbare Zufallspasswörter (Projekt) 98
 Lichtsensor
 digitaler 354
 listdir () 258
 Liste 90, 105
 Operationen 108
 Literal 26, 89
 Literatur
 freie 237
 load() 215
 localtime() 249
 Lostrommel (Projekt) 176
 LXTerminal. 504

M

MAC-Adresse 427
 Magic line 59, 176
 Magische Methode 278, 402
 Mathematische Funktion 30
 Memory-Stick 335
 Menge 90, 117

- Menu
 - Optionen der Choices 202
- Messagebox 224
- Messwertaufzeichnung 336
- Messwerterfassung 325
- Methode 105
 - magische 278, 402
- MIME 329
- Mobile Datenerfassung (Projekt) 334
- Mobile Stromversorgung 435
- Modul 32, 306
 - anlegen 306
 - Speicherort 307
- Modulo 25
- Morsen (Projekt) 126
- MPC3008 339
- Murmellabyrinth (Projekt) 487

- N**
- Name 36
- nobody 450

- O**
- Oberfläche
 - grafische 503
- Objekt 277
 - änderbar/nicht-änderbar 279
 - Wahrheitswert 71
- Objektorientierte Programmierung 277
- Oktalsystem 28
- OLED-Display 471
- omxplayer 411
- Online-Datenbank 456
- open() 211
- Operator 24
 - Plus 279
 - Vergleichsoperator 64
- os 258
- os.path 258

- P**
- Packer 180
- Parameter 31, 135
 - aktueller 135
 - beliebige Anzahl 144
- Parameterliste 138
- Parameterübergabe 141, 143
- Parameterwert
 - voreingestellter 143
- password 446
- Passwort 98
- pcmanfm 513
- pendown() 155
- Pfostenverbinder 510
- PhotoImage 181, 183
 - zoom() 185
- picamera (Modul) 408
- pickle 215
- PIL 414
 - ImageDraw 474
 - ImageFont 477
- PIL.Image
 - crop() 260
 - load() 261
 - resize() 261
 - save() 261
 - size 261
- Placer 180
- Plotter 331
- Plotter (Projekt) 330
- Plusoperator 279
- Polymorphie 402
- Pong (Projekt) 286
- Port 431
- Positionsgang 136
- Potenz 25
- Potenziometer 339
- PPM 182
- print() 62
 - Zeilenwechsel unterdrücken 63
- Privilegiertes Recht 450
- Programm
 - starten 55
- Programmierung
 - objektorientierte 277
- Programmverzweigung 63
- Projekt
 - Alarmanlage 79
 - Alkoholgehalt messen 348
 - Alkoholisches Getränk 345
 - Alphabet 280
 - Bewegung erfassen 390
 - Birnen oder Tomaten 401
 - Blinken 85
 - Datenlogger 367
 - Digitaler Bilderrahmen 257
 - digitaler Lichtsensor 354
 - Digitaler Türgong 164
 - Digitales Thermometer 324
 - Digitaluhr 250
 - Editor mit Pulldown-Menüs 200
 - Experiment zur Diffusion von Gasen 369

- Farbmixer 198
 - Fotos per E-Mail verschicken 411
 - Freies Obst 456
 - Fünf Sekunden stoppen und gewinnen 247
 - Gerichtete Bewegungen 394
 - Glückskeks 192
 - Goethe oder Schiller 231
 - Häufigkeit von Buchstaben 118
 - Kameraoptionen testen 387
 - Karteikasten 216
 - Klopzeichen 114
 - Koch-Schneeflocke 159
 - Krimiautomat 186
 - Kundenberatung 93
 - Kürzester Weg 127
 - Lauf-Simulation 297
 - LED 41
 - Lesbare Zufallspasswörter 98
 - Lostrommel 176
 - Mobile Datenerfassung 334
 - Morsen 126
 - Murmellabyrinth 487
 - Plotter 330
 - Pong 286
 - Pythagorasbaum 157
 - Quicksort 152
 - Quiz 84
 - Rechenquiz 195
 - Rekursive Spirale aus Quadraten 155
 - Rekursive Summe 151
 - Renn, Lola renn 297
 - Schiller 231
 - Schwarz-Weiß-Malerei 185
 - Sichere Kommunikation 93
 - Sierpinski-Teppich 161
 - Staumelder 239
 - Stoppuhr 267
 - Taschenlampe an- und ausschalten 45
 - Telefonliste 110
 - Thermometer 480
 - TrueType-Fonts darstellen 478
 - Überwachungskamera 389
 - Urlaubsgrüße 190
 - Verkehrsdichte auf der Autobahn 417
 - Verkehrszählungen 165
 - Visueller Zufallsgenerator 182
 - Wahrnehmungstest 263
 - Webcam 435
 - Zahlenraten 73
 - Zufallsnamen 109
 - Zufallssounds 119
 - Prozedur 146
 - Prozess 251
 - PSF 19
 - Pulldown-Menü 201
 - Pull-down-Widerstand 78
 - put() 184
 - PuTTY 429
 - pygame 121, 316
 - pygame.mixer 121
 - Pythagorasbaum (Projekt) 157
 - Python 19
 - Shell 22
 - Versionen 20
 - Python Imaging Library (PIL) 259, 414
 - Python Software Foundation 19
- Q**
- Querystring 445
 - Quicksort (Projekt) 152
 - Quiz (Projekt) 84
- R**
- Radiobutton 189
 - range() 112
 - Raspberry Pi OS 503
 - raspistill 385
 - raspid 385
 - Rasterbild 205
 - Rechenquiz (Projekt) 195
 - Recht
 - privilegiertes 450
 - Reed-Schalter 80
 - Regulärer Ausdruck 237
 - Rekursive Spirale aus Quadraten (Projekt) 155
 - Rekursive Summe (Projekt) 151
 - Relais 451
 - Relationale Datenbank 453
 - rename () 259
 - Renn, Lola renn (Projekt) 297
 - Rossum, Guido van 19
 - RPI.GPIO 43
 - RTC 251
 - run module 54
- S**
- Scale 199
 - Schalter 75
 - Schieberegler 199
 - Schiller oder Goethe (Projekt) 231
 - Schleife 72
 - Schlüssel 125

Schlüsselwort-Argument 63, 136
 Schriftart 477
 Schwarz-Weiß-Malerei (Projekt) 185
 Schwellenwertverfahren 185
 SDHC 212
 SD-Karte 212
 Seiteneffekt 150
 self 282
 Sense HAT 481
 Sensor 319
 Sequenz 90, 96
 Index 97
 Indizierung 97
 Slicing 97
 Vervielfältigung 96
 set 117
 Shebang 58
 Shell 22
 Shell-Fenster 54
 Shortcuts siehe Tastenkombination
 showturtle() 155
 Sichere Kommunikation (Projekt) 93
 side 181
 Sierpinski-Teppich (Projekt) 161
 Skript 53
 sleep() 250
 Slicing 97
 Sound 121
 Spannung messen 338
 speed() 155
 Speichern 211
 SPI-Protokoll 340
 SQL 454
 sqlite3 453
 SSD1306 472
 SSH 429
 start_new_thread() 253
 Starten 502
 Staumelder (Projekt) 239
 Steckplatine 41
 Steigende Flanke 78
 Stoppuhr (Projekt) 267
 String 101
 Methoden 232
 StringIO 395
 Stromversorgung
 mobile 435
 submit 447
 subprocess 395
 suid-Bit 451
 Switch 300
 Syntax-Highlighting 54

T

Taschenlampe an- und ausschalten (Projekt)
 45
 Taschenrechner 24
 Tastenkombination 23, 55
 Telefonliste (Projekt) 110
 Temperaturdaten
 E-Mail 328
 Temperatursensor 319
 Temperatur-Zeitdiagramm 330
 Term 24
 Text-Widget 187
 Thermoelement 319
 Thermometer (Projekt) 480
 Thread 251
 _thread 252
 time() 250
 tkinter 175
 tkinter.filedialog 224
 TrueType 478
 TrueType-Fonts darstellen (Projekt) 478
 try 216
 Tupel 91, 100
 Turtle-Grafik 153
 Typ 106
 type() 277
 Typhierarchie 89

U

Überladen 402
 Übersetzen 461
 Überwachungskamera (Projekt) 389
 UML-Klassendiagramm 282
 URL 230
 Urlaubsgrüße (Projekt) 190
 urllib.request 230
 urlopen() 230

V

VARCHAR 454
 Verbose mode 384
 Verdünnungsreihe
 lineare 350
 Vererbung 279, 284
 Vergleichsoperator 64
 Verkehrsdichte auf der Autobahn (Projekt)
 417
 Verkehrszählungen (Projekt) 165
 Verzweigung 67
 einseitig 67
 zweiseitig 69

Visueller Zufallsgenerator (Projekt) 182
Vollbildmodus 253
Voreingestellter Parameterwert 143

W

Wahrnehmungstest (Projekt) 263
Wear leveling 213
Webcam
 Bilder auswerten 413
 Internet anzapfen 413
Webcam (Projekt) 435
Webseite
 interaktive 444
Webserver 427
while 72
Widerstand 41
Widget 175, 178
 Aussehen 179
Wiederholung 72
WiFi-Adapter 427

WLAN 427
write() 211

Z

Zahl 26
 komplexe 30
Zahlenraten (Projekt) 73
Zahlenschloss 271
Zähler 77
Zeichenkette 101
Zeit 247
Zeiteinstellung 270
Zeit-Objekt 249
Zufallsfunktion 109
Zufallsnamen (Projekt) 109
Zufallssounds (Projekt) 119
Zusicherung 148
Zuweisung
 erweiterte 40
Zuweisungsoperator 36