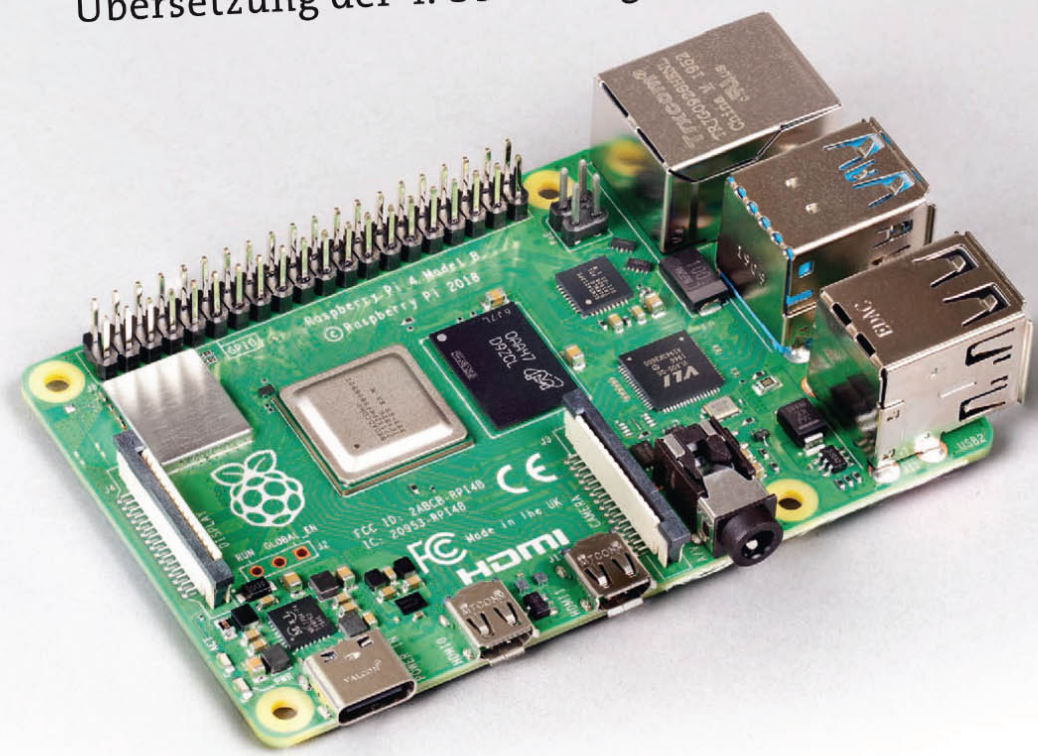
Richardson · Wallace · Donat

RASPBERRY PI DEIN EINSTIEG

Der vielseitige Linux-Computer für Smarthome, Entertainment, Experimente

Übersetzung der 4. US-Auflage



Make:

dpunkt.verlag

Inhalt

Cover

Über die Autoren

Titel

Impressum

Inhaltsverzeichnis

Vorwort

- Die Raspberry Pi Foundation
- Was können Sie mit ihm tun?
  - Ein universell einsetzbarer Computer
  - Programmieren lernen
  - Projektplattform
  - Produkt-Prototyping
- Raspberry Pi für Maker
- Linux und Raspberry Pi
- Was andere mit dem Raspberry Pi gemacht haben
- In diesem Buch genutzte Konventionen
- Danksagung
- Materialliste

1 Fahrt aufnehmen

- Die Boards im Überblick
- Das passende Zubehör
- Das Gehäuse
- Wählen Sie eine Distribution aus
- Die SD-Karte flashen
  - Für fortgeschrittene Nutzer: Erstellen Sie Ihr eigenes Disk Image

Booten
Konfigurieren Sie Ihren Pi
Online gehen
Herunterfahren
Den Pi »headless« betreiben
Fehlerbehebung
Welches Board besitzen Sie?
Weitere Informationen
2 Linux auf dem Raspberry Pi
Arbeiten an der Befehlszeile
Dateien und das Dateisystem
Weitere Linux-Befehle
Prozesse
Sudo und Berechtigungen
Das Netzwerk
/etc
Datum und Uhrzeit setzen
Neue Software installieren
Sound in Linux
Upgraden Ihrer Firmware
Weitere Informationen
3 Andere Betriebssysteme und Linux-Distributionen
Distributionen fürs Heimkino
Distributionen für Musik
Retrocomputing und Retrogaming
Das Internet of Things
Andere nützliche Distributionen
Weitere Informationen

4 Python auf dem Pi

Hallo Python

Ein bisschen mehr Python

Objekte und Module

Noch mehr Module

Andere Programme von Python aus starten

Fehlerbehebung

Weitere Informationen

5 Arduino und der Pi

Den Arduino im Raspberry Pi OS installieren

Den seriellen Port herausfinden

Serielle Kommunikation

Verwenden von Firmata

Weitere Informationen

6 Die grundlegenden Ein- und Ausgänge

Eingangs- und Ausgangsanschlüsse nutzen

Digitale Ausgabe: eine LED zum Leuchten bringen

Digitaler Eingang: einen Taster auslesen

Projekt: Cron-Lampenzeitschaltuhr

Befehle skripten

Eine Lampe anschließen

Befehle zeitgesteuert über cron ausführen

Mehr zu Cron

Weitere Informationen

7 Ein- und Ausgänge mit Python programmieren

Installation

GPIO in Python testen

Eine LED blinken lassen

Einen Taster auslesen
Projekt: ein einfaches Soundboard
Weitere Informationen
8 Analoge Ein- und Ausgänge
Ausgang: Konvertieren von digital in analog
Testdurchlauf mit PWM
Weitere Möglichkeiten mit PWM
Eingang: Konvertieren von analog in digital
Variable Widerstände
Weitere Informationen
9 Einsatz von Kameras
Anschließen und Testen des Kameramoduls
Projekt: ein GIF erstellen
Herstellen von Videoaufnahmen
USB-Webcams testen
Installation und Test von OpenCV
Zusätzlicher Schritt beim Kameramodul für den Raspberry Pi
Ein Bild anzeigen
Ein Bild verändern
Zugriff auf die Kamera
Gesichtserkennung
Projekt: Raspberry Pi Fotostudio
Weitere Informationen
10 Python und das Internet
Daten von einem Webserver laden
Die Wettervorhersage abfragen
Serving Pi (ein Webserver sein)
Grundlagen zu Flask

Das Web mit der realen Welt verbinden
Projekt: WebLamp
Weitere Informationen
Anhang A: Ein SD-Karten-Image schreiben
Eine SD-Karte unter macOS schreiben
Eine SD-Karte unter Linux schreiben
Eine SD-Karte unter Windows schreiben
Anhang B: Der Raspberry Pi Pico
Der Pico selbst
MicroPython
MicroPython auf dem Pico installieren
Linux und Mac
Microsoft Windows
MicroPython auf dem Pico verwenden
Eine LED auf dem Pico blinken lassen
Anhang C: Noch ein Raspberry Pi?!
Index

4 Python auf dem Pi

Python ist eine tolle erste Programmiersprache – sie ist klar strukturiert und man kommt schnell zu Ergebnissen. Wichtiger ist noch, dass es viele andere Anwender gibt, die man bei Problemen fragen und mit denen man Code teilen kann.

Guido van Rossum hat Python im Jahr 1991 entworfen und sehr schnell ihre Eignung als erste Programmiersprache erkannt. 1999 hat van Rossum ein viel gelesenes Proposal namens »Computer Programming for Everybody« (*https://www.python.org/doc/essays/cp4e/*) geschrieben, in dem sich eine Vision für ein ambitioniertes Programm zum Programmieren-Lernen mit Python an Grund- und Mittelschulen findet. Mehr als ein Jahrzehnt später sieht es so aus, als ob sich diese Vision durch die große Beliebtheit und den Einsatz des Raspberry Pi in der Schule erfüllt.

Python ist eine *Interpreter-Sprache*, Sie können also ein Programm oder Skript schreiben und es direkt ausführen lassen, statt es erst in Maschinencode zu kompilieren. Mit Interpreter-Sprachen kann man schneller Programme schreiben, und Sie haben ein paar angenehme Nebeneffekte. So müssen Sie zum Beispiel in Python dem Computer nicht explizit mitteilen, ob es sich bei einer Variablen um eine Zahl, eine Liste oder einen String handelt. Der Interpreter ermittelt den Datentyp selbst, wenn Sie das Skript ausführen.

Der Python-Interpreter kann auf zwei Arten ausgeführt werden: als interaktive Shell, um einzelne Befehle zu starten, oder als Befehlszeilenprogramm, um Skripten laufen zu lassen. Die integrierte Entwicklungsumgebung (Integrated Development Environment, IDE), die für Python auf dem Raspberry Pi zur Verfügung steht, nennt sich IDLE. Die aktuellste Version des Raspberry Pi OS bringt IDLE nicht mehr länger mit, sondern setzt stattdessen auf Geany und Thonny. Beide IDEs sind meiner Meinung nach etwas robuster als IDLE, bei der es sich mehr um eine einfache Befehlszeilenumgebung handelt.

Das große Python-Versionsrätsel

Experimentieren Sie mit dem Pi, werden Sie vielleicht feststellen, dass es zwei Python-Versionen auf dem Pi gibt. Das ist gängige Praxis (allerdings auch ein bisschen verwirrend). Zum Zeitpunkt der Entstehung dieses Buches ist Python 3 die neueste Version und auch die

einzige, die aktiv weiterentwickelt wird – die Unterstützung von Python v2.7 wurde im Januar 2021 eingestellt. Leider gab es beim Wechsel von Version 2 auf Version 3 Änderungen an der Sprache, sodass sie nicht abwärtskompatibel ist. Und obwohl Python 3 schon seit ein paar Jahren im Einsatz ist, hat es eine ganze Zeit gedauert, bis sich diese neue Version auch verbreitet hat. Viele Pakete, die von Anwendern kommen, sind immer noch nicht auf Python 3 aktualisiert. Wenn Sie nach Python-Dokumentationen suchen, wird es noch verwirrender – achten Sie darauf, die richtige Version zu verwenden!

Sie können Python 3 explizit so starten:

```
python3
```

Die Beispiele in diesem Buch funktionieren sowohl mit Python 2.7 als auch 3.X, sofern nicht explizit darauf hingewiesen wird, aber sie sind komplett in der Syntax von Python 3 geschrieben (zum Beispiel mit dem Verwenden von öffnenden und schließenden Klammern in print-Anweisungen).

Hallo Python

Am besten steigt man in Python ein, indem man direkt loslegt. Sie können zwar einen beliebigen Texteditor nutzen, um Skripten zu schreiben, aber wir werden mit der IDLE-3-Anwendung einsteigen. Klicken Sie dazu auf das Desktop-Menü unten links und wählen Sie dann Programming→Python (IDLE 3).

Wenn IDLE geladen ist, sehen Sie ein Fenster mit der interaktiven Shell. Das dreifache Größer-Zeichen (>>>) ist der interaktive Prompt – wenn Sie es sehen, heißt das, dass der Interpreter auf Ihre Eingabe wartet. Geben Sie dort Folgendes ein:

```
>>> print("SalutonMondo!")
```

Drücken Sie nun die Eingabetaste. Python führt diese Anweisung aus, und Sie werden das Ergebnis im Shell-Fenster sehen. Sie können die Shell als eine Art Taschenrechner nutzen, um Anweisungen auszuprobieren oder Berechnungen durchzuführen:

```
>>> 3+4+5

12
```

Stellen Sie sich die Anweisungen, die in der interaktiven Shell ausgeführt werden, als ein Programm vor, das Sie Zeile für Zeile laufen lassen. Sie können sogar Variablen einrichten oder Module importieren:

```
>>> import math

>>> (1 + math.sqrt(5)) / 2

1.618033988749895
```

Die `import`-Anweisung stellt Ihrem Programm alle mathematischen Funktionen von Python zur Verfügung (mehr zu Modulen in Abschnitt »Objekte und Module«, S. 75). Um eine Variable einzurichten, nutzen Sie den Zuweisungsoperator (=):

```
>>> import math

>>> radius = 20

>>> radius *2* math.pi

1256.6370614359173
```

Wollen Sie alle Variablen abräumen und in einem sauberen Status neu beginnen, wählen Sie aus dem Menü Shell→Restart Shell. Sie können auch die interaktive Shell verwenden, um Informationen über den Einsatz einer bestimmten Anweisung, eines Moduls oder anderer Python-Themen zu erhalten. Dazu greifen Sie auf den Befehl `help()` zurück:

```
help ("print")
```

Um eine Liste aller verfügbaren Themen zu erhalten, probieren Sie einmal Folgendes aus:

```
help ("topics")
```

```
help ("keywords")

help ("modules")
```

Der Python-Interpreter ist ideal zum Austesten von Anweisungen oder einfachen Operationen. Häufig werden Sie aber Ihr Python-Skript wie eine eigenständige Anwendung ausführen wollen. Um ein neues Python-Programm aufzusetzen, wählen Sie File→New Window, dann bietet Ihnen IDLE ein Fenster zur Skript-Bearbeitung (siehe Abb. 4–1).

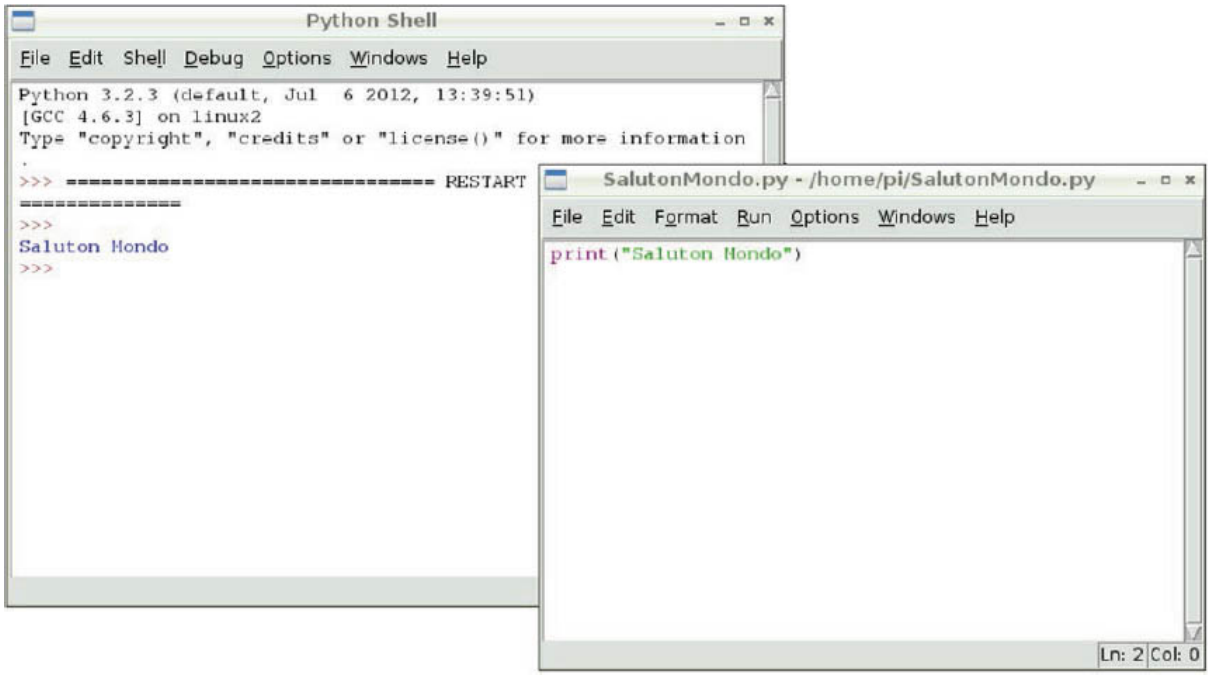


Abb. 4–1 Die interaktive Shell von IDLE (links) und ein Editor-Fenster (rechts)

Geben Sie einmal eine Codezeile ein und rufen Sie dann Run→Run Module auf. Sie werden eine Warnung erhalten: »Source Must Be Saved OK To Save?«. Sichern Sie Ihr Skript in Ihrem Home-Verzeichnis als *Saluton-Mondo.py* und sehen Sie dann, wie es in der Shell ausgeführt wird.

Manchmal werden Sie die IDLE-Umgebung nicht nutzen wollen. Um ein Skript von der Befehlszeile auszuführen, öffnen Sie das Terminal und tippen Sie:

```
python3 SalutonMondo.py
```

Das sind schon alle grundlegenden Mechanismen, die Sie kennen müssen, um mit der Umgebung vertraut zu sein. Als Nächstes müssen Sie noch die Sprache

erlernen.

Ein bisschen mehr Python

Wenn Sie sich Python aus der Arduino-Welt nähern, sind Sie es gewohnt, Programme (in Arduino *Sketches* genannt, in Python eher *Skripten*) in einem setup/loop-Format zu schreiben, wobei `setup()` eine Funktion ist, die nur einmal ausgeführt wird, während die Funktion `loop()` wieder und wieder läuft. Das folgende Beispiel zeigt, wie man das in Python erreichen kann. Wählen Sie an der Shell in IDLE 3 den Befehl New Window und geben Sie das Folgende ein:

```
# Setup

n=0

# Loop

while True:

    n = n + 1

    # % ist der Modulo-Operator

    if ((n % 2) == 0):

    print(n)
```

Wählen Sie Run Module und geben Sie Ihrem Skript einen Namen (wie zum Beispiel *CountEvens.py*). Während es läuft, sollten alle geraden Zahlen ausgegeben werden (drücken Sie aber irgendwann Strg-C, um die Ausgabe abzubrechen, sonst läuft es bis in alle Ewigkeit).

Sie können dies auch mittels der `for`-Schleife, bei der nur die ersten 100 Zahlen vom Typ Integer gezählt werden, implementieren:

```
for n in range (0, 100):

    if ((n % 2) == 0):

    print(n)
```

Im voranstehenden Beispiel ist Ihnen vielleicht aufgefallen, dass jede Einrückstufe aus vier Leerzeichen besteht, nicht aus einem Tabulatorzeichen. (Sie können in IDLE allerdings Tab drücken, es werden dann vier Leerzeichen eingefügt.) In Python ist das Einrücken nicht nur eine Lesehilfe, sondern es hat auch eine strukturelle Bedeutung – ähnlich den Codeblöcken innerhalb von geschweiften Klammern in C oder JavaScript.

Dies ist einer der größten Fallstricke für Einsteiger, insbesondere beim Kopieren und Einfügen von Code. Wir sind allerdings trotzdem der Meinung, dass das verpflichtende Einrücken Python zu einer sehr gut lesbaren Sprache macht. In den Python Style Guidelines (*bit.ly/1ne4xO5*) finden Sie Tipps zum Schreiben von lesbarem Code.

Es ist wichtig, auf Ihre Leerzeichen zu achten. Python ist eine sehr strukturierte Sprache, bei der Whitespace (also Leerzeichen, Tabs und so weiter) die Struktur festlegen. Im nächsten Beispiel ist alles, was unter der `loop()`-Funktion um eine Ebene eingerückt ist, Teil dieser Funktion. Das Ende einer Schleife wird durch die Zeile bestimmt, in der die Einrückung wieder eine Ebene zurückgesetzt ist (oder durch das Ende der Datei). Das unterscheidet Python von Sprachen wie C, bei denen Blöcke durch geschweifte Klammern oder andere Kennzeichen abgeschlossen werden.

Verwenden Sie Funktionen, um Codeabschnitte in einen Codeblock umzuwandeln, der an anderer Stelle in Ihrem Skript aufgerufen werden kann. Um das vorige Beispiel mit Funktionen umzuschreiben, nutzen Sie den folgenden Code (wenn Sie es ausführen wollen, speichern Sie es als *CountEvens.py*):

```
# Globale Variablen deklarieren

n=0  (1)

# Setup-Funktion
```

```
def setup():    (2)
    global n

n = 100

def loop():    (3)
    global n
    n = n + 1
    if ((n % 2) == 0):
    print(n)

# Main    (4)

setup()

while True:
    loop()
```

In diesem Beispiel wird jede gerade Zahl ab 102 ausgegeben. So funktioniert das Programm:

1. Als Erstes wird die Variable `n` als globale Variable definiert, die in allen Blöcken des Skripts genutzt werden kann.
2. Hier wird die Funktion `setup()` definiert (aber noch nicht ausgeführt).
3. Genauso findet sich hier die Definition der Funktion `loop()`.
4. Im Hauptcodeblock wird `setup()` einmalig aufgerufen und danach `loop()`.

Die Verwendung des Schlüsselworts `global` in der ersten Zeile jeder Funktion ist wichtig – sie weist den Interpreter an, die globale Variable `n` zu verwenden und keine zweite (lokale, oder für diese Funktion private) Variable `n` zu erzeugen, die nur hier genutzt werden kann.

Dieses Tutorial ist zu kurz für eine vollständige Python-Referenz. Um die Sprache wirklich zu erlernen, schauen Sie sich vielleicht einmal *Learning Python the Hard Way* (*learnpythonthehardway.org*), *Think Python* (O'Reilly) oder *Python – kurz & gut* (O'Reilly) an. Der Rest dieses Kapitels wird Sie mit genug Wissen ausstatten, dass Sie die späteren Beispiele verstehen und ausführen können. Dabei lernen Sie die grundlegenden Features und verfügbaren Module kennen.

Befehlszeile oder IDLE?

Ihnen wird vielleicht auffallen, dass die Ausgabe von IDLE beim Ausführen von Beispielcode, der Text in der Shell ausgibt, sehr langsam ist. Für einen Vergleich können Sie IDLE und ein neues Terminal parallel offen haben. Führen Sie in IDLE das CountEvens-Skript per Run Module aus – es benötigt einen Vorsprung. Dann geben Sie an der Befehlszeile im Terminal Folgendes ein:

```
python3 CountEvens.py
```

Sie werden schnell merken, wie stark die IDE das Skript auf dem doch nur begrenzt schnellen Pi ausbremst. Die Beispiele weiter unten im Buch werden alle an der Befehlszeile ausgeführt, Sie können aber IDLE natürlich trotzdem als Editor nutzen.

Objekte und Module

Sie müssen die Syntax-Grundlagen der Objekte und Module verstehen, um die Beispiele in diesem Buch nutzen zu können. Python ist eine aufgeräumte Sprache mit nur 35 reservierten *Schlüsselwörtern* (siehe Tab. 4–1). Diese Schlüsselwörter bilden den Kern der Sprache, mit dem Sie die Struktur und den Ablauf in Ihrem Skript definieren. So gut wie alles, was kein Schlüsselwort ist, können Sie sich als ein *Objekt* vorstellen. Dabei handelt es sich um eine Kombination aus Daten und Verhaltensweisen mit einem Namen. Sie können die Daten eines Objekts ändern, Informationen darüber auslesen und sogar andere Objekte beeinflussen.

Bedingte Anweisungen	Schleifen	Eingebaute Funktionen	Klassen, Module, Funktionen	Fehler-behandlung
if	for	print	class	try
else	in	pass	def	def
elif	while	del	global	finally
not	break	lambda	break	raise
or	as		nonlocal	assert
and	continue		yield	with
is			import	
True			return	
False			from	
None				

Tab. 4–1 Python besitzt nur 35 reservierte Schlüsselwörter

In Python sind Strings, Listen, Funktionen, Module und selbst Zahlen Objekte. Ein Python-Objekt kann man sich als gekapselte Sammlung von Attributen und Methoden vorstellen. (Wenn Sie sich nicht ganz sicher sind: Attribute werden manchmal auch als Variablen bezeichnet, Methoden als Funktionen. In der Welt der Software kommen diese Begriffe häufig synonym zum Einsatz, insbesondere Methoden und Funktion.) Sie erhalten Zugriff auf diese Attribute und Methoden über eine einfache Punkt-Syntax. Geben Sie zum Beispiel den folgenden Code am interaktiven Shell-Prompt ein, um ein String-Objekt zu erstellen und die Methode aufzurufen, mit der es bei sich selbst den Wortanfang in Großbuchstaben umwandelt:

```
>>> myString = "quux"

>>> myString.capitalize()

'Quux'
```

Oder nutzen Sie `reverse()`, um die Reihenfolge einer Liste umzukehren:

```
>>> myList = ['a', 'man', 'a', 'plan', 'a', 'canal']

>>> myList.reverse()

>>> print(myList)

['canal', 'a', 'plan', 'a', 'man', 'a']
```

Sowohl String als auch List sind eingebaute Module der *Standardbibliothek*, die jedem Python-Programm zur Verfügung stehen. In den beiden Modulen finden sich sehr viele Funktionen für den Umgang mit Strings und Listen, wie zum Beispiel `capitalize()` und `reverse()`.

Manche der Module der Standardbibliothek sind nicht eingebaut. Sie müssen Python mit der Anweisung `import` explizit mitteilen, dass Sie sie nutzen wollen. Um das Modul `time` aus der Standardbibliothek zu nutzen und Zugriff auf hilfreiche Funktionen für den Umgang mit Zeitwerten und Timestamps zu erhalten, verwenden Sie:

```
import time
```

Sie werden auch einmal die Anweisung `import as` sehen, mit der sich das Modul in Ihrem Programm umbenennen lässt:

```
import time as myTime
```

Dazu kommt noch `from import`, um ausgewählte Funktionen aus einem Modul zu laden:

```
from time import clock
```

Hier ein kurzes Beispiel für ein Python-Skript, das die Module `time` und `datetime` aus der Standardbibliothek nutzt, um in jeder Sekunde die aktuelle Uhrzeit auszugeben:

```
from datetime import date time

from time import sleep

while True:

now = str(datetime.now())

print(now)

sleep(1)
```

Die Funktion `sleep` stoppt die Ausführung des Programms für eine Sekunde. Beim Ausführen des Codes wird Ihnen auffallen, dass sich die Zeit auf Dauer ein wenig verschiebt.

Das hat zwei Gründe:

- Der Code berücksichtigt nicht, dass das Berechnen der aktuellen Uhrzeit auch Zeit kostet.
- Andere Prozesse nutzen ebenfalls die CPU des Pi und nehmen Ihrem Programm Rechenzeit weg. Daran sollten Sie immer denken: Wenn Sie auf dem Raspberry Pi programmieren, arbeiten Sie nicht in einer *Real-Time-Umgebung*.

Wenn Sie die Funktion `sleep()` verwenden, werden Sie bemerken, dass sie eine Genauigkeit von ungefähr 5 ms besitzt. Sie können das auch so lesen, dass die Funktion `sleep()` auf etwa 5 ms genau ist, aber nicht genauer. Müssen Sie Ihr Programm für weniger als 5 ms pausieren lassen, sollten Sie sich eventuell Gedanken über den Einsatz eines Real-Time-Betriebssystems oder eines Microcontrollers für Ihre Anwendung machen, weil der Pi diese Anforderungen eher nicht erfüllen kann.

Als Nächstes wollen wir das Beispiel anpassen, eine Textdatei öffnen und regelmäßig Daten protokollieren. Wenn es um Textdateien geht, ist alles in Python ein String.

Mit der Funktion `str()` wandeln Sie Zahlen in Strings um (mit `int()` können Sie den umgekehrten Weg gehen). Das wird als *Casting* einer Variable

bezeichnet:

```
from datetime import datetime

from time import sleep

import random

log = open ("log.txt", "w")

for i in range(5):

    now = str(datetime.now())

    # Generate some random data in the range 0-1024

    data = random.randint(0, 1024)

    log.write(now + " " + str(data) + "\n")

    print(".")

    sleep(.9)

log.flush()

log.close()
```

Achten Sie bei einer echten Anwendung, die Daten protokolliert, darauf, dass der Raspberry Pi Datum und Uhrzeit korrekt eingestellt hat (siehe Abschnitt »Datum und Uhrzeit setzen«, S. 55).

Hier ein weiteres Beispiel (*ReadFile.py*), bei dem ein Dateiname als Argument eingelesen wird (in der Shell starten Sie das Programm mit `python3 ReadFile.py` *`Dateiname`*). Das Programm öffnet die Datei, liest jede Zeile als String ein und gibt ihn aus. Beachten Sie, dass `print()` wie `println()` in anderen Sprachen funktioniert – die Funktion fügt bei der Ausgabe des Strings einen Zeilenumbruch ein. Mit dem Argument `end` bei `print()` wird dieser Zeilenumbruch verhindert:

```
# Eine Datei als Befehlszeilenparameter öffnen und einlesen

import sys

if (len(sys.argv) != 2):

    print("Usage: python3 ReadFile.py filename")

    sys.exit()

scriptname = sys.argv[0]

filename = sys.argv[1]

file = open(filename, "r")

lines = file.readlines()

file.close()

for line in lines:

print(line,end = '')
```

Noch mehr Module

Einer der Gründe für die Beliebtheit von Python ist die große Zahl an von Anwendern bereitgestellten Modulen, die auf der Standardbibliothek aufbauen. Der Python Package Index oder PyPI (*pypi.org*) ist die umfassendste Liste mit Paketen (oder Modulen), die für die Sprache zur Verfügung stehen. Manche der beliebteren Module, die auch für den Raspberry Pi sehr nützlich sind, finden Sie in Tab. 4–2 aufgeführt. Sie werden einige dieser Module später noch verwenden, insbesondere das Modul GPIO für den Zugriff auf die Ein- und Ausgabe-Pins des Raspberry Pi.

Modul	Beschreibung	URL	Paketname
`RPi.GPIO`	Zugriff auf die GPIO-Pins	*sourceforge.net/projects/raspberry-gpio-python*	python-rpi.gpio
`GPIOzero`	vereinfachter Zugriff auf GPIO-Pins	*gpiozero.readthedocs.org*	python-gpiozero
`Pygame`	Gaming-Framework	*pygame.org*	pythonpygame
`OpenCV`	einfache API für Computer Vision	*opencv.org/*	opencv-python
`Scipy`	wissenschaftliches Rechnen	*www.scipy.org*	python-scipy
`Numpy`	die numerischen Grundlagen zu Scipy	*numpy.scipy.org*	python-numpy
`Flask`	Microframework für Webentwicklung	*flask.palletsprojects.com*	python-flask
`Feedparser`	Parser für Atom- und RSS-Feeds	*pypi.python.org/pypi/feedparser*	Kein Paket
`Requests`	»HTTP für Menschen«	*docs.python-requests.org*	python-requests
`PIL`	Bildverarbeitung	*https://python-pillow.org*	python-imaging
`wxPython`	GUI-Framework	*wxpython.org*	python-wxgtk2.8
`pySerial`	Zugriff auf den seriellen Port	*github.com/pyserial/pyserial*	python-serial

PyUSB	FTDI-USB-Schnittstelle	*bleyer.org/pyusb*	Kein Paket

Tab. 4–2 Für Pi-Anwender besonders interessante Pakete

Um eines dieser Module zu nutzen, müssen Sie den Code herunterladen, das Paket konfigurieren und installieren. Das Modul NumPy beispielsweise kann folgendermaßen installiert werden:

```
sudo apt install python-numpy
```

Wurde ein Paket von seinem Ersteller auf dem üblichen Weg (mit Pythons distutils-Tool) aus seinen Modulen gebaut, müssen Sie es nur herunterladen, entpacken und das Folgende eingeben:

```
python setup.py install
```

Einfaches Installieren von Modulen mit Pip

Viele Module lasse sich mit apt installieren. Möglicherweise ist für Sie auch der Pip Package Installer (*www.pypa.io*) interessant. Hierbei handelt es sich um ein Tool, mit dem sich auf sehr einfache Weise Packages von der PyPI-Bibliothek installieren lassen. Installieren Sie Pip mittels `apt`:

```
sudo apt install python3-pip
```

Anschließend können Sie bei der Installation der meisten Module Pip einsetzen, um die Downloads und Abhängigkeiten zu verwalten. Hier ein Beispiel:

```
pip3 install flask
```

In späteren Kapiteln werden Sie sehr ausgiebig anwendungsspezifische Module nutzen, aber hier kommt schon einmal ein Beispiel, das zeigt, wie mächtig einige Module sein können. Das Feedparser-Modul ist ein universeller Parser, der RSS- oder Atom-Feeds erfasst und einen einfachen Zugriff auf deren Inhalt erlaubt. Weil die meisten Streams mit Informationen im Web mit RSS- oder Atom-Ausgabe arbeiten, ist der Feedparser eine der Möglichkeiten, Ihren Raspberry Pi im Internet of Things einzubinden.

Installieren Sie zunächst das Feedparser-Modul mittels Pip (siehe Kasten: »Einfaches Installieren von Modulen mit Pip«):

```
pip3 install –user feedparser
```

Mit Feedparser können Sie RSS-Feeds in Ihrem Python-Code herunterladen und parsen – daher der Name.

Um ihn zu nutzen, übergeben Sie einfach die URL eines RSS-Feeds an die Parse-Funktion. Feedparser wird das XML des Feeds abrufen, entsprechend parsen und in eine spezielle Datenlistenstruktur, ein sogenanntes *Verzeichnis* (*Dictionary*), umwandeln. Ein Python-Verzeichnis ist eine Liste mit Schlüssel/Wert-Paaren, die manchmal auch als *Hash* oder *assoziatives Array* bezeichnet werden. Der geparste Feed ist ein Verzeichnis, und bei den geparsten Elementen handelt es sich ebenfalls um ein Verzeichnis, wie in diesem Beispiel gezeigt wird, in dem das aktuelle Wetter in Providence, Rhode Island, von *weather.gov* abgerufen wird:

```
import feedparser

feed_url = "http://w1.weather.gov/xml/current_obs/KPVD.rss"

feed = feedparser.parse(feed_url)

RSSitems = feed["items"]

for item in RSSitems:

    weather = item["title"]

    print(weather)
```

Andere Programme von Python aus starten

Bei Python ist es recht einfach, mittels des Moduls `sys.subprocess` andere Programme auf Ihrem Pi anzustoßen. Versuchen Sie Folgendes:

```
from datetime import datetime
```

```
from time import sleep

import subprocess

for count in range(0, 60):

filename = str(datetime.now()) + ".jpg"

subprocess.call(["fswebcam", filename])

sleep(60)
```

Dies ist ein einfaches Zeitraffer-Skript, bei dem eine Stunde lang einmal pro Minute ein Foto mit einer Webcam geschossen wird. Auf Kameras wird detaillierter in Kapitel 9 eingegangen. Dieses Programm sollte funktionieren, wenn eine USB-Kamera angeschlossen und das Programm `fswebcam` installiert ist. Für diese Installation führen Sie zunächst den folgenden Befehl aus:

```
sudo apt install fswebcam
```

Die Funktion `subprocess.call` übernimmt eine Liste von Strings, verknüpft sie (mit Leerzeichen dazwischen) und versucht, das angegebene Programm auszuführen. Wenn es sich um ein gültiges Programm handelt, erzeugt der Raspberry Pi dafür einen separaten Prozess, der neben dem Python-Skript läuft. Der Befehl im voranstehenden Beispiel entspricht der folgenden Eingabe an einem Terminal-Prompt:

```
fswebcam 20220812.jpg
```

Um beim Programmaufruf weitere Parameter hinzuzufügen, ergänzen Sie die Liste in der Funktion `subprocess.call` um entsprechende Strings:

```
subprocess.call(["fswebcam", "-r", "1280x720", filename])
```

Mittels dieses Codes wird dem Programm `fswebcam` ein Parameter geschickt, um die Auflösung des erzeugten Bildes zu ändern.

Ein Python-Skript automatisch beim Start ausführen

Da der Pi als Stand-alone-Gerät fungieren kann, stellt sich häufig die Frage, wie ein Python-Skript automatisch gestartet werden kann, wenn der Pi hochgefahren wird. Die Antwort besteht darin, einen Zugang zur Datei */etc/rc.local* hinzuzufügen, die genau für diesen Zweck in der Linux-Welt genutzt wird. Bearbeiten Sie einfach die Datei

```
sudo nano /etc/rc.local
```

und fügen Sie einen Befehl zum Ausführen des Skripts zwischen dem kommentierten Abschnitt und `exit 0` ein – ungefähr so:

```
/usr/bin/python3 /home/pi/foo.py&
```

Durch das Kaufmanns-Und am Ende wird das Skript als Hintergrundprozess ausgeführt, und für alle anderen Dienste auf dem Pi kann der Boot-Prozess fortgesetzt werden. Beachten Sie auch, dass Sie den vollständigen Pfad für die ausführbare Python-Datei angeben – das kann notwendig sein, wenn Ihr Pi mehrere Benutzer kennt.

Fehlerbehebung

Sie werden mit Ihrem Code ganz unvermeidlich auf Probleme stoßen und einen Fehler suchen müssen. Der IDLE Interactive Mode kann hier sehr hilfreich sein – das Debug-Menü bietet eine Reihe von Tools, mit deren Hilfe Sie herausbekommen können, was Ihr Code tatsächlich tut. Sie haben auch die Möglichkeit, alle Ihre Variablen anzeigen und den Code Zeile für Zeile ausführen zu lassen.

Syntaxfehler lassen sich am einfachsten beheben – normalerweise handelt es sich nur um einen Tippfehler oder ein falsches Verständnis der Programmiersprache. *Semantische Fehler*, bei denen das Programm an sich korrekt ist, aber nicht das tut, was es soll, können schwieriger zu finden sein. Hier kann der Debugger sehr hilfreich sein. Es braucht Jahre, effektives Debuggen zu erlernen, aber hier eine kurze Checkliste, die Sie bei der Arbeit mit Python auf dem Pi berücksichtigen sollten:

- Nutzen Sie `print()`, um anzuzeigen, dass das Programm einen bestimmten Punkt erreicht hat.

- Mit `print()` können Sie auch die Werte von Variablen ausgeben lassen.
- Achten Sie ganz besonders auf Whitespace, um sicherzustellen, dass die Blöcke so definiert sind, wie Sie sich das vorgestellt haben.
- Denken Sie beim Debuggen von Syntaxfehlern daran, dass der tatsächliche Fehler weit vor der Stelle liegen kann, die der Interpreter meldet.
- Prüfen Sie noch einmal alle Ihre globalen und lokalen Variablen.
- Achten Sie auf korrekt schließende Klammerpaare.
- Schauen Sie sich die Reihenfolge der Operationen bei Rechnungen an – wenn Sie sich nicht sicher sind, nutzen Sie Klammern. So liefern zum Beispiel `3 + 4 * 2` und `(3 + 4) * 2` verschiedene Ergebnisse.

Nachdem Sie sich mit Python vertraut gemacht und Erfahrung gesammelt haben, sollten Sie einmal einen Blick auf die Module `code` und `logging` werfen – dort finden Sie weitere Debugging-Tools.

Weitere Informationen

Es gibt noch viel mehr zu Python zu sagen. Hier ein paar nützliche Ressourcen:

Programmieren lernen mit Python. Einstieg in die Programmierung

von Allen Downey

Dies ist ein klares und recht kompaktes Buch zum Programmieren-Lernen (in Python). O'Reilly-Verlag, 2. Ausgabe 2014. ISBN 978-3-95561-806-3

Python – kurz & gut

von Mark Lutz

Weil es manchmal besser ist, ein Buch durchzublättern, als sich durch ein Dutzend Posts bei Stack Overflow durchzuklicken. O'Reilly-Verlag, 5. Auflage 2014. ISBN 978-3-95561-770-7

Stack Overflow (*stackoverflow.com****)***

Stack Overflow ist trotzdem eine ausgezeichnete Quelle kollektiven Wissens. Das funktioniert besonders gut, wenn Sie nach einer bestimmten Lösung oder Fehlermeldung suchen – bestimmt hatte schon mal jemand anderes das gleiche Problem.

Learn Python the Hard Way *(learnpythonthehardway.org)*

von Zed Shaw

Ein tolles Buch und eine tolle Online-Ressource – lesen Sie zumindest die Einführung »The Hard Way Is Easier«.

Python kinderleicht! Einfach programmieren lernen – nicht nur für Kids

von Jason R. Briggs

Auch eher ein allgemeines Buch zum Programmieren-Lernen (in Python), das für jüngere Leser geschrieben ist. dpunkt.verlag, 2. Auflage 2016. ISBN 978-3-86490-344-1

Index

> (Redirection) 49
| (Pipe-Operator) 47
~ (Tilde) 44
4K-Auflösung 6
802.11 (WiFi-Stick) 20
1080p 6
—help (Option) 46
.. (Punkte) 45
 SD-Karten
  flashen
    Linux 201
    Windows 203

A

ADC (Analog Digital Converter) 101, 136
ADS1x15 136
Advanced Linux Sound Architecture (ALSA) 57
AlaMode 97
ALSA (Advanced Linux Sound Architecture) 57
alsamixer 57
Amiga 2
analog
  vs. digital 131
Analog Digital Converter (ADC) 101, 136

analoge Audio-Signale 14
analoge Ein- und Ausgänge 131
analoge Sensoren 101
Android 7
API (Application Programming Interface) 175
aplay 57
Application Programming Interface (API) 175
apt 81
apt-get 56
Arbeitsverzeichnis 44
Arch Linux 62
Arduino 5, 73, 87
  IDE 91
  installieren 89
  kompatible Geräte 95
  serielle Kommunikation 91
  serieller Port 90
  Stromversorgung 89
  Uno 5
  USB-Kabel 89
ARM 2
ARM-A72 14
assoziatives Array 82
Atom-Feeds 81
Attribute 76
Audio 126
Ausschalter 32
Autovervollständigen 41
avahi-daemon 185

B

balenaEtcher 200, 203
Bare Metal 6
bash 41
BBC Micro 2
BCM-Nummer 121
Befehlspuffer 41
Befehlszeile 38, 41
Berechtigungen 52
Beschleunigungssensor 23
Betriebssysteme 61
Bewegung 99
Binary Blob 25
Bittorrent 26
Board-Version 35
Bonjour Services 185
Booten 27, 38
Breadboard s. Steckplatinen
Broadcom-Nummer 121
BUB I 34

C

Camera Serial Interface (CSI) 17, 148
Cascades 164
Casting 78
cat 50
cd 44
chgrp 52
chmod 53
chown 52

Chrome 216
Chromium 39
chsh 41
CLI 29
Commodore 64 2
Composite Video 6, 14
Compute Module 12, 218
Computer Vision 155
Configuration Tool 27
CoolTerm 211
cron 111, 115
  Kommentare 116
  leere Zeile 116
crontab 116
Cross-Over-Kabel 32
CSI (Camera Serial Interface) 148

D

dash 41
date 56
Dateien 42
Dateimanager 39
Dateisystem 42
  virtuell 107
Daten aus dem Netz laden 173
Datum 55
dd 199
Debian 7
Debuggen 84
Dictionary 81

digital
  vs. analog 131
Dimmen 133
Disk Image 26, 197
diskutil 198
Display Serial Interface 17
Distributionen 24, 61
Django 184
dmesg 27, 155
Dock 22
Druck 23
DS1307 21
DSI 17

E

Echtzeituhr 21
Ein- und Ausgänge 99, 119
  analog 131
ELEC 6
Electrodoc 121
Emacs 40
Entfernung 99
etc (Verzeichnis) 55
Ethernet 20
Ethernet-Port 15

F

Feedparser 81
Fehler
  Semantische Fehler 84
  Syntaxfehler 84

Fehlerbehebung 34, 83
Firmata 95
Firmware 58
Flask 183
Forum 101
Fotostudio 167
Fotozelle 142
FreeRTOS 6
Fritzing 127
Frotz 65
fswebcam 82
FTDI 95
  Kabel 33
Funktionen 74

G

Geany 70
Gehäuse 23
General Purpose Input/Output 17
Gesichtserkennung 164
Gleichstrommotor 146
  PWM 135
global 75
GPIO 17
GPIO-Pins 99
  Python 119
GPIO Zero 119
  installieren 119
grep 50
Gyroskop 23

gzip 47

H

Haar Cascades 164

Hash 81

HATs 11, 22

HDMI 6, 16, 19

head 50

Headless 32

Heimkino 62

Herunterfahren 32

history 41

Home-Verzeichnis 44

HTTP-Status-Code 174

I

I2C (Inter-Integrated Circuit) 136

IDE (Integrated Development Environment) 69

IDLE 69

ifconfig 54

ImageMagick 152

import 71, 77

Infrarot-Kameras 21, 148

int 78

Integrated Development Environment (IDE) 69

Inter-Integrated Circuit (I2C) 136

Internet 173

- Client 173
- mit realer Welt verbinden 188
- Wettervorhersage 175

Internet of Things (IoT) 65

Interrupts 125
IoT (Internet of Things) 65

J

Jinja2 186
Job Scheduler 115
Joystick 23
JSON 177
json (Bibliothek) 179

K

Kameramodul 21
Kameras 147
  anschließen 150
  Board 148
  GIF erstellen 152
  Infrarot 148
kill 51
Kodi 62
Konfigurationsdateien 55
Konfigurieren 27
Konvertieren
  analog nach digital 135
  digital nach analog 132
Kühlkörper 20

L

Lampe 113
Lampenschalter 191
Lampenzeitschaltuhr 111
LCDs 21
Leafpad 40

LEDs 99, 103
  blinken 122
  dimmen 133
  RGB 135
Leerzeichen 74
less 47, 49
LibreOffice-Suite 4
Licht 99
Lightweight X11 Desktop Environment (LXDE) 37
Linux 37
Locale 30
ls 45
lsblk 201
Luftfeuchtigkeit 23
luvcview 154
LXDE (Lightweight X11 Desktop Environment) 37
LXTerminal 37, 41

M

Magnetometer 23
man 46
Materialliste 9
Mathematica 40
Maus 19
Media Center 6
Media Player 40
Methoden 76
MicroPython 207
  installieren 209
  Unterschiede zu Python 208

MicroSD-Card 16
MicroSD-Karte 19
Mikrocontroller 5
minicom 211
mkdir 47
Module 77, 79
more 49
Motoren 99
Musik 64
mv 46

N

Nano 40
Network Time Protocol (NTP) 55
Netzwerk 54
Node.js 97
NOOBS-Installer 61
NOOBS-Tool 25
NTP (Network Time Protocol) 55

O

Objekte 75
OctoPrint 67
omxplayer 153
Online 31
Openbox 38
OpenCV 155
  Bild anzeigen 157
  Fotostudio 167
  Gesichtserkennung 164
  Kamera-Modul 157

Zugriff auf Kamera 162
OpenELEC 63
Openwrt 66
OSMC 6, 62
OV5647 147
Overscan 29

P

Paketmanager 56
Parameter 45
passwd 54
Performance 29
PHATs 97
picamera 157
Pi Cobbler Plus 102
PiCore Player 64
Pidora 62
pigpio 119
PiMusic Box 64
ping 54
Pi NoIR 148
pip 81
—user (Flag) 120
Pipes 47
PiPlay 65
Pip Package Installer 81
Plex Media Service 6
Polling 124
Port Forwarding 187
Portsplus Port-ID 103

Potentiometer 137
PowerSwitch Tail II 111, 191
print 79
Programmiersprachen 101
Prompt 44, 70
Protokolle 95
Prozesse 50
Prozessor 14
ps 50
Pulsweitenmodulation (PWM) 132
Punkt-Syntax 76
PuTTY 211
PWM (Pulsweitenmodulation) 132, 206
PWM/Servo Driver 135
pyFirmata 96
PyPI (Python Package Index) 79
pySerial 91
Python 69
  GPIO-Pins 119
  Internet 173
  LEDs 122
  MicroPython 207
    Unterschiede 208
  Taster 124
  Versionen 70
Python Package Index (PyPI) 79
Python Style Guidelines 73

Q

Qt on Pi 66

R

Raspberry-Menü 38

Raspberry Pi

- Betriebssystem 7
- Compute Module 4, 12
- Pico 205
  - PWM 206
- Pico RP2040 5
- Versionen 12
- Zero 5, 12
- Zero W 5, 12
- Zero W2 217

Raspberry Pi Foundation 2

Raspberry Pi Imager 200, 203

Raspberry Pi OS 7, 25, 61

Raspbmc 62

raspi-config (Tool) 30

raspistill 152

raspivid 153

Read-Evaluate-Print-Loop (REPL) 207

Redirection 48

Reguläre Ausdrücke 50

Relais 99

REPL (Read-Evaluate-Print-Loop) 207

Requests 173

Response 174

Retro Computing 65

Retro Gaming 6, 65

RetroPie 6, 65

RISCOS 65
rm 46
rmdir 46
Root-User 52
RP2040-Chip 205
RP2041 219
RPi.GPIO 119
rpi-update 58
RSS-Feeds 81

S

Satellite CCRMA 64
Schalter 45, 99
Schlüssel/Wert-Paare 81, 178
Schlüsselwörter 75
scp 55
Scratch 4
Screenshot 57
scrot 57
SD-Karten
  flashen 25
    macOS 197
Secure Shell (SSH) 29, 32, 55
Secure Sockets Layer (SSL) 55
Semantische Fehler 84
Sense HAT 23
Sensoren 23, 99
Serial Peripheral Interface 21
serielle Kommunikation 91
sftp 55

SHA-Checksum 34
Shell 41
Shell-Skript 112
Shields 22
Sketches 73
sleep 77
Software installieren 56
Sonic Pi 4, 64
Soundboard 126
Soundkarten 21
Spannungsteiler 143
Spectrum ZX 2
SPI 21
SSH (Secure Shell) 29, 32, 55
SSL (Secure Sockets Layer) 55
Stack Overflow 101
Standard-Bibliothek 77
Standard Error 47
Standard Input 47
Standard Output 47
Standardpasswort 29
Start-Skripten 55
startx 29, 38
Status-Code 174
Status-LEDs 15
stderr 47
stdin 47
stdout 47
Steckplatinen 104

Stellknöpfe 99
Steuertasten 49
str 78
Stromversorgung 16, 18, 27
sudo 48, 52
Sun Vox 64
Super-User 48, 52
Syntaxfehler 84
sys 93
System-on-a-Chip 5

T

Tabulatorzeichen 73
tail 50
tar 47
Tastatur 19, 29
Taster 99
  auslesen 108
  Python 124
Tastgrad 132
Temperatur 23, 99
Template Engine 186
Terminal-Emulator 210
Texas Instruments 136
Texteditor 40
TFT-Display 21
Thonny 70
Tilde 44
time 77
Timestamps 77

top 50

U

Ubuntu 7
UbuntuCore 66
Ubuntu MATE 62
U-Control 21
Uhrzeit 55
USB-C-Adapter 18
USB-C-Buchse 16
USB-Hub 20
USB-Ports 15
usermod 90
Userspace 106

V

Variablen 71
vcgencmd 58
Versionen 12
vim 40
virtuelles Dateisystem 107
VLC 40, 57
VNC 33
Volumio 64

W

Weather Underground 175
Webbrowser 39
Webcams 147
  testen 154
Webframework 183
Web kiosk 66

WebLamp 190
Weborina 65
Webserver
  Daten laden 173
  Server 183
WebSocket 97
Wettervorhersage 175
Whitespace 74
Widerstand
  druckempfindlich 145
Widerstände 109
  druckempfindlich 142
  variabel 142
WiFi 20
Win32 Disk Imager 203
Windows 10 Core 7, 66
Wolfram 40

X

XBMC 62
X-Window-System 37

Z

Zeitplan 115
Zeitwerte 77
Zeitzone 30
Zubehör 18