

```
d1 = [format(i, '#010b')[2:] for i in data]
```



```
if not d1[i].startswith("0001"): i +=1  
    [i:i+14]
```

```
return d
```

```
__get_number(self, d):
```

```
"""Returns a float number representing the number  
of the digits on the multimeter display, if  
valid number. Otherwise -1 is returned. """
```

```
try:
```

```
A = d[1][7] + d[2][7] + d[2][5] + d[2][4] +
```

```
    d[1][5] + d[1][6] + d[2][6]
```

```
B = d[3][7] + d[4][7] + d[4][5] + d[4][4] +
```

```
    d[3][5] + d[3][6] + d[4][6]
```

```
C = d[5][7] + d[6][7] + d[6][5] + d[6][4] +
```

```
    d[5][5] + d[5][6] + d[6][6]
```

```
D = d[7][7] + d[8][7] + d[8][5] + d[8][4] +
```

```
    d[7][5] + d[7][6] + d[8][6]
```

```
n = int(DIGIT[A] + DIGIT[B] + DIGIT[C] + DI
```

```
# take the point position into account
```

```
if d[7][4] == "1": n/=10
```

```
elif d[5][4] == "1": n/=100
```

```
elif d[3][4] == "1": n/= 1000
```

```
# take prefix k, M, etc. into account
```

```
if d[9][4] == "1": n /= 10**6
```

```
elif d[9][5] == "1": n /= 10**9
```

```
elif d[9][6] == "1": n /= 1000
```

```
elif d[10][4] == "1": n /= 1000
```

```
elif d[10][5] == "1": n /= 1000
```

```
elif d[10][6] == "1": n /= 1000
```

```
# finally return the result
```

```
if d[11][4] == "1": n /= 1000
```

```
return n
```

```
except:
```

```
    return "Error"
```

```
__get_unit(self, d):
```

```
""" Returns
```

```
(A, V, C
```

```
if d[11][4] == "1": return "F"
```

```
elif d[11][5] == "1": return "Ohms"
```

```
elif d[12][4] == "1": return "A"
```

Michael
Weigend

9., erweiterte
Auflage

Python 3

Lernen und professionell anwenden

Das umfassende Praxisbuch

Hinweis des Verlages zum Urheberrecht und Digitalen Rechtemanagement (DRM)

Liebe Leserinnen und Leser,

dieses E-Book, einschließlich aller seiner Teile, ist urheberrechtlich geschützt. Mit dem Kauf räumen wir Ihnen das Recht ein, die Inhalte im Rahmen des geltenden Urheberrechts zu nutzen. Jede Verwertung außerhalb dieser Grenzen ist ohne unsere Zustimmung unzulässig und strafbar. Das gilt besonders für Vervielfältigungen, Übersetzungen sowie Einspeicherung und Verarbeitung in elektronischen Systemen.

Je nachdem wo Sie Ihr E-Book gekauft haben, kann dieser Shop das E-Book vor Missbrauch durch ein digitales Rechtemanagement schützen. Häufig erfolgt dies in Form eines nicht sichtbaren digitalen Wasserzeichens, das dann individuell pro Nutzer signiert ist. Angaben zu diesem DRM finden Sie auf den Seiten der jeweiligen Anbieter.

Beim Kauf des E-Books in unserem Verlagsshop ist Ihr E-Book DRM-frei.

Viele Grüße und viel Spaß beim Lesen,

Ihr mitp-Verlagsteam



Neuerscheinungen, Praxistipps, Gratiskapitel,
Einblicke in den Verlagsalltag –
gibt es alles bei uns auf Instagram und Facebook



[instagram.com/mitp_verlag](https://www.instagram.com/mitp_verlag)



[facebook.com/mitp.verlag](https://www.facebook.com/mitp.verlag)

Michael Weigend

Python 3

Lernen und professionell anwenden

Das umfassende Praxisbuch

9. Auflage



Bibliografische Information der Deutschen Nationalbibliothek

Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über <http://dnb.d-nb.de> abrufbar.

ISBN 978-3-7475-0545-8

9. Auflage 2022

www.mitp.de

E-Mail: mitp-verlag@sigloch.de

Telefon: +49 7953 / 7189 - 079

Telefax: +49 7953 / 7189 - 082

© 2022 mitp Verlags GmbH & Co. KG, Frechen

Dieses Werk, einschließlich aller seiner Teile, ist urheberrechtlich geschützt. Jede Verwertung außerhalb der engen Grenzen des Urheberrechtsgesetzes ist ohne Zustimmung des Verlages unzulässig und strafbar. Dies gilt insbesondere für Vervielfältigungen, Übersetzungen, Mikroverfilmungen und die Einspeicherung und Verarbeitung in elektronischen Systemen.

Die Wiedergabe von Gebrauchsnamen, Handelsnamen, Warenbezeichnungen usw. in diesem Werk berechtigt auch ohne besondere Kennzeichnung nicht zu der Annahme, dass solche Namen im Sinne der Warenzeichen- und Markenschutz-Gesetzgebung als frei zu betrachten wären und daher von jedermann benutzt werden dürften.

Lektorat: Sabine Schulz, Janina Bahlmann

Sprachkorrektur: Petra Heubach-Erdmann

Covergestaltung: Christian Kalkert

Coverbild: © Marc AZEMA / stock.adobe.com

Satz: III-satz, Kiel, www.drei-satz.de

Inhaltsverzeichnis

Einleitung	23
Warum Python?	23
Python 3	23
An wen wendet sich dieses Buch?	23
Inhalt und Aufbau	24
Hinweise zur Typographie	25
Programmbeispiele	26
1 Grundlagen	27
1.1 Was ist Programmieren?	27
1.2 Hardware und Software	28
1.3 Programm als Algorithmus	29
1.4 Syntax und Semantik	30
1.5 Interpreter und Compiler	30
1.6 Programmierparadigmen	32
1.7 Objektorientierte Programmierung	33
1.7.1 Strukturelle Zerlegung	33
1.7.2 Die Welt als System von Objekten	34
1.7.3 Objekte besitzen Attribute und beherrschen Methoden	35
1.7.4 Objekte sind Instanzen von Klassen	36
1.8 Hintergrund: Geschichte der objektorientierten Programmierung	36
1.9 Aufgaben	37
1.10 Lösungen	38
2 Der Einstieg – Python im interaktiven Modus	39
2.1 Python installieren	39
2.2 Python im interaktiven Modus	42
2.2.1 Start des Python-Interpreters in einem Konsolenfenster	42
2.2.2 Die IDLE-Shell	43
2.2.3 Die ersten Python-Befehle ausprobieren	43
2.2.4 Hotkeys	44
2.3 Objekte	45
2.4 Namen	47
2.5 Hintergrund: Syntax-Regeln für Bezeichner	47
2.6 Schlüsselwörter	48
2.7 Anweisungen	49
2.7.1 Ausdruckanweisungen	50
2.7.2 Import-Anweisungen	54
2.7.3 Zuweisungen	55
2.7.4 Erweiterte Zuweisungen	59

2.7.5	Hintergrund: Dynamische Typisierung	59
2.8	Aufgaben	60
2.9	Lösungen	62
3	Python-Skripte	65
3.1	Ausprobieren, nachmachen, besser machen!	65
3.2	Skripte editieren und ausführen mit IDLE	65
3.3	Ausführen eines Python-Skripts	67
3.4	Kommentare	69
3.5	Die Zeilenstruktur von Python-Programmen	70
3.6	Das EVA-Prinzip	73
3.7	Phasen der Programmentwicklung	75
3.8	Guter Programmierstil	76
3.9	Hintergrund: Die Kunst des Fehlerfindens.	78
3.10	Weitere Entwicklungsumgebungen für Python	80
3.10.1	Thonny – eine Entwicklungsumgebung für Python-Einsteiger . . .	80
3.10.2	Python in der Cloud	81
3.10.3	Jupyter Notebook und Google Colab.	82
3.10.4	Entwicklungsumgebungen für Profis	82
3.11	Aufgaben	83
3.12	Lösungen	84
4	Standard-Datentypen	87
4.1	Daten als Objekte	87
4.2	Fundamentale Datentypen im Überblick	89
4.3	Typen und Klassen	90
4.4	NoneType	91
4.5	Wahrheitswerte – der Datentyp bool	91
4.6	Ganze Zahlen	92
4.7	Gleitkommazahlen	94
4.8	Komplexe Zahlen	95
4.9	Arithmetische Operatoren für Zahlen	96
4.10	Sequenzen	101
4.10.1	Zeichenketten (Strings)	102
4.10.2	Bytestrings	104
4.10.3	Tupel	105
4.10.4	Liste	106
4.10.5	Bytearray	107
4.10.6	Einige Grundoperationen für Sequenzen.	107
4.10.7	Veränderbare und unveränderbare Sequenzen	109
4.11	Mengen	110
4.12	Dictionaries	111
4.13	Typumwandlungen	111
4.13.1	int()	113
4.13.2	float()	113

4.13.3	complex()	114
4.13.4	bool()	114
4.13.5	str()	114
4.13.6	dict(), list() und tuple()	115
4.14	Aufgaben	115
4.15	Lösungen	118
5	Kontrollstrukturen	123
5.1	Einfache Bedingungen	123
5.1.1	Vergleiche	123
5.1.2	Zugehörigkeit zu einer Menge (in, not in)	127
5.1.3	Beliebige Ausdrücke als Bedingungen	127
5.2	Zusammengesetzte Bedingungen – logische Operatoren	128
5.2.1	Negation (not)	128
5.2.2	Konjunktion (and)	129
5.2.3	Disjunktion (or)	130
5.2.4	Formalisierung von Bedingungen	131
5.2.5	Hinweis zum Programmierstil	132
5.3	Programmverzweigungen (bedingte Anweisungen)	132
5.3.1	Einseitige Verzweigung (if)	133
5.3.2	Zweiseitige Verzweigung (if-else)	133
5.3.3	Mehrfache Fallunterscheidung (elif)	134
5.3.4	Bedingte Ausdrücke	136
5.4	Bedingte Wiederholung (while)	136
5.4.1	Endlosschleifen	137
5.5	Iteration über eine Kollektion (for)	139
5.5.1	Zählschleifen – Verwendung von range()	140
5.5.2	Verschachtelte Iterationen	141
5.5.3	Vertiefung: Iterative Berechnung rekursiver Folgen	143
5.6	Abbruch einer Schleife mit break	143
5.6.1	Abbruch eines Schleifendurchlaufs mit continue	144
5.7	Abfangen von Ausnahmen mit try	145
5.7.1	try...except	146
5.8	Aufgaben	148
5.9	Lösungen	152
6	Funktionen	157
6.1	Aufruf von Funktionen	157
6.2	Definition von Funktionen	160
6.3	Schrittweise Verfeinerung	162
6.4	Ausführung von Funktionen	166
6.4.1	Globale und lokale Namen	166
6.4.2	Seiteneffekte – die global-Anweisung	169
6.4.3	Parameterübergabe	170

6.5	Voreingestellte Parameterwerte	172
6.5.1	Schlüsselwort-Argumente	174
6.6	Funktionen mit beliebiger Anzahl von Parametern	176
6.7	Lokale Funktionen.	177
6.8	Rekursive Funktionen.	178
6.9	Experimente zur Rekursion mit der Turtle-Grafik.	180
6.9.1	Turtle-Befehle im interaktiven Modus	180
6.9.2	Eine rekursive Spirale.	181
6.9.3	Baumstrukturen	183
6.9.4	Künstlicher Blumenkohl – selbstähnliche Bilder.	184
6.10	Rekursive Zahlenfunktionen	186
6.11	Hintergrund: Wie werden rekursive Funktionen ausgeführt?	187
6.11.1	Execution Frames	187
6.11.2	Rekursionstiefe	188
6.12	Funktionen als Objekte.	190
6.12.1	Hintergrund: Typen sind keine Funktionen	191
6.13	Lambda-Formen	191
6.14	Funktionsannotationen: Typen zuordnen	192
6.15	Hinweise zum Programmierstil.	193
6.15.1	Allgemeines.	193
6.15.2	Funktionsnamen.	193
6.15.3	Kommentierte Parameter.	194
6.15.4	Docstrings	194
6.16	Aufgaben	196
6.17	Lösungen	199
7	Sequenzen, Mengen und Generatoren	203
7.1	Gemeinsame Operationen für Sequenzen	203
7.1.1	Zugriff auf Elemente einer Sequenz.	204
7.1.2	Slicing von Sequenzen	205
7.1.3	Auspacken (unpacking)	206
7.2	Vertiefung: Rekursive Funktionen für Sequenzen.	207
7.2.1	Rekursives Summieren	207
7.2.2	Rekursive Suche	207
7.3	Tupel.	209
7.4	Listen	210
7.4.1	Eine Liste erzeugen.	210
7.4.2	Eine Liste verändern.	213
7.4.3	Flache und tiefe Kopien	215
7.4.4	Listen sortieren	216
7.4.5	Binäre Suche in einer sortierten Liste.	218
7.4.6	Zwei Sortierverfahren im Vergleich	219
7.4.7	Modellieren mit Listen – Beispiel: die Charts	223
7.5	Generatoren.	227
7.5.1	Generatorausdrücke	228

7.5.2	Generatorfunktionen	228
7.5.3	Iteratoren	230
7.5.4	Verwendung von Generatoren	231
7.6	Mengen	231
7.6.1	Operationen für Mengen	233
7.6.2	Modellieren mit Mengen – Beispiel: Graphen	234
7.7	Aufgaben	237
7.8	Lösungen	239
8	Dictionaries	241
8.1	Operationen für Dictionaries	241
8.2	Wie erstellt man ein Dictionary?	242
8.2.1	Definition mit einem Dictionary-Display	242
8.2.2	Schrittweiser Aufbau eines Dictionarys.	244
8.2.3	Ein Dictionary aus anderen Dictionaries zusammensetzen – update()	245
8.3	Zugriff auf Daten in einem Dictionary	245
8.3.1	Vergebliche Zugriffsversuche	245
8.4	Praxisbeispiel: Vokabeltrainer	246
8.5	Typische Fehler	248
8.6	Aufgaben	248
8.7	Lösungen	251
9	Ein- und Ausgabe	255
9.1	Streams	255
9.1.1	Die Rolle der Streams bei E/A-Operationen	255
9.1.2	Was ist ein Stream?	256
9.1.3	Eine Datei öffnen	257
9.1.4	Speichern einer Zeichenkette	258
9.1.5	Laden einer Zeichenkette aus einer Datei	260
9.1.6	Absolute und relative Pfade	260
9.1.7	Zwischenspeichern, ohne zu schließen	262
9.1.8	Zugriff auf Streams (lesen und schreiben)	263
9.2	Mehr Zuverlässigkeit durch try- und with-Anweisungen	265
9.2.1	try...finally	266
9.2.2	with-Anweisungen	267
9.3	Objekte speichern mit pickle	268
9.3.1	Funktionen zum Speichern und Laden	268
9.4	Die Streams sys.stdin und sys.stdout	270
9.5	Ausgabe von Werten mit der print()-Funktion	271
9.5.1	Anwendung: Ausgabe von Tabellen	272
9.6	Kommandozeilen-Argumente (Optionen)	273
9.6.1	Zugriff auf Optionen	274
9.6.2	Beispiel	274
9.6.3	Skripte mit Optionen testen	275

9.7	Aufgaben	275
9.8	Lösungen	278
10	Definition eigener Klassen	283
10.1	Klassen und Objekte	283
10.2	Definition von Klassen	285
10.3	Objekte (Instanzen)	287
10.4	Zugriff auf Attribute – Sichtbarkeit	290
	10.4.1 Öffentliche Attribute	290
	10.4.2 Private Attribute	291
	10.4.3 Properties	293
	10.4.4 Dynamische Erzeugung von Attributen	295
10.5	Methoden	295
	10.5.1 Polymorphismus – überladen von Operatoren	296
	10.5.2 Vertiefung: Objekte ausführbar machen – die Methode __call__()	299
	10.5.3 Statische Methoden	300
10.6	Abstraktion, Verkapselung und Geheimnisprinzip	302
	10.6.1 Abstraktion	302
	10.6.2 Verkapselung	302
	10.6.3 Geheimnisprinzip	302
10.7	Vererbung	303
	10.7.1 Spezialisierungen	303
	10.7.2 Beispiel: Die Klasse Konto – eine Spezialisierung der Klasse Geld	304
	10.7.3 Vertiefung: Standardklassen als Basisklassen	307
10.8	Hinweise zum Programmierstil	309
	10.8.1 Schreibweise	309
	10.8.2 Sichtbarkeit	309
	10.8.3 Dokumentation von Klassen	310
10.9	Typische Fehler	311
	10.9.1 Versehentliches Erzeugen neuer Attribute	311
	10.9.2 Verwechseln von Methoden und Attributen	311
10.10	Aufgaben	312
10.11	Lösungen	315
11	Klassen wiederverwenden – Module	321
11.1	Testen einer Klasse in einem lauffähigen Stand-alone-Skript	321
11.2	Module speichern und importieren	323
11.3	Den Zugang zu einem Modul sicherstellen	325
	11.3.1 Erweitern der Verzeichnisliste sys.path	325
	11.3.2 Anwendungsbeispiel: Eine interaktive Testumgebung	325
	11.3.3 Kompilieren von Modulen	326
11.4	Programmierstil: Verwendung und Dokumentation von Modulen	327

12	Objektorientiertes Modellieren	329
12.1	Phasen einer objektorientierten Software-Entwicklung	329
12.1.1	Objektorientierte Analyse (OOA)	329
12.1.2	Objektorientierter Entwurf (OOD)	330
12.1.3	Objektorientierte Programmierung (OOP)	330
12.2	Beispiel: Modell eines Wörterbuchs	330
12.2.1	OOA: Entwicklung einer Klassenstruktur	330
12.2.2	OOD: Entwurf einer Klassenstruktur zur Implementierung in Python	334
12.2.3	OOP: Implementierung der Klassenstruktur	336
12.3	Assoziationen zwischen Klassen	340
12.3.1	Reflexive Assoziationen	340
12.3.2	Aggregation	342
12.4	Beispiel: Management eines Musicals	343
12.4.1	OOA	343
12.4.2	OOD	345
12.4.3	OOP	345
12.5	Aufgaben	355
12.6	Lösungen	356
13	Textverarbeitung	361
13.1	Standardmethoden zur Verarbeitung von Zeichenketten	361
13.1.1	Formatieren	362
13.1.2	Schreibweise	362
13.1.3	Tests	363
13.1.4	Entfernen und Aufspalten	364
13.1.5	Suchen und Ersetzen	365
13.2	Codierung und Decodierung	365
13.2.1	Platonische Zeichen und Unicode	365
13.2.2	Vertiefung: Zeichenketten durch Bytefolgen darstellen	367
13.3	Automatische Textproduktion	369
13.3.1	Texte mit variablen Teilen – Anwendung der String-Methode format()	369
13.3.2	Vertiefung: Eine Tabelle erstellen	372
13.3.3	Mahnbriefe	373
13.3.4	Textuelle Repräsentation eines Objekts	374
13.3.5	F-Strings	376
13.4	Analyse von Texten	377
13.4.1	Chat-Bots	377
13.4.2	Textanalyse mit einfachen Vorkommenstests	378
13.5	Reguläre Ausdrücke	380
13.5.1	Die Funktion findall() aus dem Modul re	381
13.5.2	Aufbau eines regulären Ausdrucks	381
13.5.3	Objekte für reguläre Ausdrücke	384
13.5.4	Strings untersuchen mit search()	385

13.5.5	Textpassagen extrahieren mit findall()	386
13.5.6	Zeichenketten zerlegen mit split()	387
13.5.7	Teilstrings ersetzen mit sub()	388
13.5.8	Match-Objekte	389
13.6	Den Computer zum Sprechen bringen – Sprachsynthese	391
13.6.1	Buchstabieren	393
13.6.2	Den Klang der Stimme verändern	395
13.7	Aufgaben	398
13.8	Lösungen	401
14	Systemfunktionen	409
14.1	Das Modul sys – die Schnittstelle zum Laufzeitsystem	409
14.1.1	Informationen über die aktuelle Systemumgebung	410
14.1.2	Stander Eingabe und -ausgabe	411
14.1.3	Die Objektverwaltung beobachten mit getrefcount()	412
14.1.4	Ausführung eines Skripts beenden	413
14.2	Das Modul os – die Schnittstelle zum Betriebssystem	413
14.2.1	Dateien und Verzeichnisse suchen	414
14.2.2	Hintergrund: Zugriffsrechte abfragen und ändern (Windows und Unix)	415
14.2.3	Dateien und Verzeichnisse anlegen und modifizieren	417
14.2.4	Merkmale von Dateien und Verzeichnissen abfragen	418
14.2.5	Pfade verarbeiten	419
14.2.6	Hintergrund: Umgebungsvariablen	421
14.2.7	Systematisches Durchlaufen eines Verzeichnisbaumes	422
14.3	Datum und Zeit	424
14.3.1	Funktionen des Moduls time	425
14.3.2	Sekundenformat	425
14.3.3	Zeit-Tupel	426
14.3.4	Zeitstrings	427
14.3.5	Einen Prozess unterbrechen mit sleep()	428
14.4	Zeitberechnungen mit dem Modul datetime	428
14.4.1	Die Klasse datetime	428
14.4.2	Die Zeitzone	430
14.4.3	Die Klasse timedelta	431
14.5	Aufgaben	431
14.6	Lösungen	432
15	Grafische Benutzungsoberflächen mit tkinter	437
15.1	Ein einführendes Beispiel	438
15.2	Einfache Widgets	441
15.3	Die Master-Slave-Hierarchie	442
15.4	Optionen der Widgets	443
15.4.1	Optionen bei der Instanziierung setzen	443
15.4.2	Widget-Optionen nachträglich konfigurieren	444

15.4.3	Fonts	445
15.4.4	Farben	446
15.4.5	Rahmen	446
15.4.6	Die Größe eines Widgets	447
15.4.7	Leerraum um Text.	449
15.5	Gemeinsame Methoden der Widgets.	450
15.6	Die Klasse Tk	450
15.7	Die Klasse Button.	451
15.8	Die Klasse Label.	451
15.8.1	Dynamische Konfiguration der Beschriftung.	452
15.8.2	Verwendung von Kontrollvariablen.	453
15.9	Die Klasse Entry.	455
15.10	Die Klasse Radiobutton	457
15.11	Die Klasse Checkbutton.	459
15.12	Die Klasse Scale	461
15.13	Die Klasse Frame.	463
15.14	Aufgaben	463
15.15	Lösungen	464
16	Layout	469
16.1	Der Packer	469
16.2	Layout-Fehler	471
16.3	Raster-Layout	472
16.4	Vorgehensweise bei der GUI-Entwicklung	476
16.4.1	Die Benutzungsoberfläche gestalten	479
16.4.2	Funktionalität hinzufügen	482
16.5	Aufgaben	483
16.6	Lösungen	486
17	Grafik	497
17.1	Die tkinter-Klasse Canvas	497
17.1.1	Generierung grafischer Elemente – ID, Positionierung und Display-Liste.	498
17.1.2	Grafische Elemente gestalten.	500
17.1.3	Visualisieren mit Kreisdiagrammen	502
17.2	Die Klasse PhotoImage	505
17.2.1	Eine Pixelgrafik erzeugen.	506
17.2.2	Fotos analysieren und verändern.	508
17.3	Bilder in eine Benutzungsoberfläche einbinden.	511
17.3.1	Icons auf Schaltflächen.	511
17.3.2	Hintergrundbilder.	512
17.3.3	Hintergrund: Das PPM-Format	514
17.4	Die Python Imaging Library (PIL)	515
17.4.1	Installation eines Moduls mit pip	515
17.4.2	Mit PIL beliebige Bilddateien einbinden.	516

17.4.3	Steganografie – Informationen in Bildern verstecken	517
17.5	Aufgaben	519
17.6	Lösungen	520
18	Event-Verarbeitung	525
18.1	Einführendes Beispiel	526
18.2	Event-Sequenzen	528
18.2.1	Event-Typen	528
18.2.2	Qualifizierer für Maus- und Tastatur-Events	528
18.2.3	Modifizierer	530
18.3	Beispiel: Tastaturereignisse verarbeiten	530
18.4	Programmierung eines Eventhandlers	532
18.4.1	Beispiel für eine Event-Auswertung	533
18.5	Bindemethoden	534
18.6	Aufgaben	534
18.7	Lösungen	537
19	Komplexe Benutzungsoberflächen	543
19.1	Text-Widgets	543
19.1.1	Methoden der Text-Widgets	544
19.2	Rollbalken (Scrollbars)	546
19.3	Menüs	547
19.3.1	Die Klasse Menu	548
19.3.2	Methoden der Klasse Menu	548
19.4	Texteditor mit Menüleiste und Pulldown-Menü	550
19.5	Dialogboxen	552
19.6	Applikationen mit mehreren Fenstern	556
19.7	Aufgaben	559
19.8	Lösungen	560
20	Threads	565
20.1	Funktionen in einem Thread ausführen	566
20.2	Thread-Objekte erzeugen – die Klasse Thread	568
20.3	Aufgaben	571
20.4	Lösungen	572
21	Fehler finden und vermeiden	577
21.1	Testen von Bedingungen	577
21.1.1	Ausnahmen (Exceptions)	577
21.1.2	Testen von Vor- und Nachbedingungen mit assert	578
21.1.3	Vertiefung: Programmabstürze ohne Fehlermeldung	581
21.2	Debugging-Modus und optimierter Modus	583
21.3	Ausnahmen gezielt auslösen	584
21.4	Selbstdokumentation	585
21.5	Dokumentation eines Programmlaufs mit Log-Dateien	587
21.5.1	Grundfunktionen	587

21.5.2	Beispiel: Logging in der GUI-Programmierung	588
21.6	Vertiefung: Professionelles Arbeiten mit Logging	589
21.6.1	Logging-Levels	589
21.6.2	Logger-Objekte	594
21.6.3	Das Format der Logging-Meldungen konfigurieren	594
21.7	Debugging	596
21.7.1	Schaltflächen des Debug-Control-Fensters	597
21.7.2	Breakpoints	597
22	Dynamische Webseiten – CGI und WSGI	599
22.1	Wie funktionieren dynamische Webseiten?	599
22.2	Wie spät ist es? Aufbau eines CGI-Skripts	601
22.2.1	Die Ausgabe eines CGI-Skripts	601
22.2.2	Wie ist ein CGI-Skript aufgebaut?	602
22.2.3	Verwendung von Schablonen	603
22.2.4	Aufruf mit dem Webbrowser	604
22.2.5	Ein einfacher HTTP-Server	605
22.3	Kommunikation über interaktive Webseiten	605
22.3.1	Aufbau eines HTML-Formulars	606
22.3.2	Eingabekomponenten in einem HTML-Formular	608
22.4	Verarbeitung von Eingabedaten mit FieldStorage	610
22.5	Sonderzeichen handhaben	612
22.6	CGI-Skripte debuggen	613
22.7	Der Apache-Webserver	614
22.7.1	Den Apache-Server installieren	615
22.7.2	CGI-Skripte auf dem Apache-Server	616
22.8	Dynamische Webseiten mit WSGI	616
22.8.1	Einfacher geht's nicht: Ein Stand-alone-WSGI-Webserver mit wsgiref	617
22.9	mod_wsgi	618
22.9.1	Installation	618
22.9.2	Vorbereitung	619
22.9.3	Den Apache-Server konfigurieren	619
22.9.4	Ein WSGI-Skript für den Apache-Server	621
22.9.5	Tipps zum Debuggen	621
22.9.6	Zugriff von einem entfernten Rechner im WLAN	622
22.10	Verarbeitung von Eingabedaten aus Formularen	623
22.11	Objektorientierte WSGI-Skripte – Beispiel: ein Chatroom	625
22.11.1	Die HTML-Seiten	627
22.11.2	Die Klassen für den Chatroom	629
22.11.3	Skript (Teil 2):	629
22.12	WSGI-Skripte mit Cookies	632
22.12.1	Besuche zählen	633

23	Internet-Programmierung	641
23.1	Was ist ein Protokoll?	641
23.2	Übertragung von Dateien mit FTP	642
	23.2.1 Das Modul ftplib	642
	23.2.2 Navigieren und Downloaden	643
	23.2.3 Ein Suchroboter für FTP-Server	645
23.3	Zugriff auf Webseiten mit HTTP und HTTPS	649
	23.3.1 Automatische Auswertung von Webseiten	651
23.4	Zugriff auf Ressourcen im Internet über deren URL	653
	23.4.1 Webseite herunterladen und verarbeiten	653
	23.4.2 Projekt: Wie warm wird es heute?	654
	23.4.3 Datei herunterladen und speichern	655
	23.4.4 Projekt: Filme herunterladen	655
23.5	E-Mails senden mit SMTP	657
23.6	Aufgaben	660
23.7	Lösungen	661
24	Datenbanken	669
24.1	Was ist ein Datenbanksystem?	669
24.2	Entity-Relationship-Diagramme (ER-Diagramme)	670
24.3	Relationale Datenbanken	671
24.4	Darstellung von Relationen als Mengen oder Dictionaries	672
24.5	Das Modul sqlite3	673
	24.5.1 Beispiel: Telefonbuch	673
	24.5.2 Eine Tabelle anlegen	674
	24.5.3 Anfragen an eine Datenbank	675
	24.5.4 Datensuche im interaktiven Modus	676
	24.5.5 SQL-Anweisungen mit variablen Teilen	678
	24.5.6 Vertiefung: SQL-Injection	680
24.6	Online-Redaktionssystem mit Datenbankanbindung	681
	24.6.1 Objektorientierte Analyse (OOA)	683
	24.6.2 Objektorientierter Entwurf des Systems (OOD)	684
	24.6.3 Hintergrund: Authentifizieren mit SHA-256	686
	24.6.4 Implementierung des Redaktionssystems mit Python (OOP)	687
24.7	Aufgaben	697
24.8	Lösungen	698
25	Testen und Tuning	701
25.1	Automatisiertes Testen	701
25.2	Testen mit Docstrings – das Modul doctest	701
25.3	Praxisbeispiel: Suche nach dem Wort des Jahres	704
25.4	Klassen testen mit doctest	711
	25.4.1 Wie testet man eine Klasse?	711
	25.4.2 Normalisierte Whitespaces – doctest-Direktiven	712
	25.4.3 Ellipsen verwenden	712

25.4.4	Dictionaries testen	713
25.5	Gestaltung von Testreihen mit unittest	713
25.5.1	Einführendes Beispiel mit einem Testfall	714
25.5.2	Klassen des Moduls unittest	715
25.5.3	Weiterführendes Beispiel	718
25.6	Tuning	721
25.6.1	Performance-Analyse mit dem Profiler	721
25.6.2	Praxisbeispiel: Auswertung astronomischer Fotografien	723
25.6.3	Performance-Analyse und Tuning	729
25.7	Aufgaben	730
25.8	Lösungen	732
26	XML und JSON	739
26.1	Was ist XML?	739
26.2	XML-Dokumente	740
26.3	Ein XML-Dokument als Baum	742
26.4	DOM	743
26.5	Das Modul xml.dom.minidom	746
26.5.1	XML-Dokumente und DOM-Objekte	746
26.5.2	Die Basisklasse Node	748
26.5.3	Die Klassen Document, Element und Text	750
26.6	Attribute von XML-Elementen	752
26.7	Anwendungsbeispiel 1: Eine XML-basierte Klasse	752
26.8	Anwendungsbeispiel 2: Datenkommunikation mit XML	755
26.8.1	Überblick	756
26.8.2	Das Client-Programm	757
26.8.3	Das Server-Programm	760
26.9	JSON	764
26.9.1	JSON-Texte decodieren	765
26.9.2	Decodierungsfehler	766
26.9.3	Ein Dictionary als JSON-Objekt speichern: Kompakt oder gut lesbar?	766
26.9.4	Projekt: Verarbeitung von Wetterdaten	769
26.10	Aufgaben	772
26.11	Lösungen	773
27	Modellieren mit Kellern, Schlangen und Graphen	775
27.1	Stack (Keller, Stapel)	775
27.2	Queue (Schlange)	778
27.3	Graphen	779
27.4	Aufgaben	789
27.5	Lösungen	791
28	Benutzungsoberflächen mit Qt	795
28.1	Was bietet PyQt5?	795
28.2	PyQt5 erkunden	796

28.3	Wie arbeitet PyQt? Applikation und Fenster	796
28.4	Eine objektorientierte Anwendung mit PyQt5	797
28.5	Ein Webbrowser	798
28.6	Interaktive Widgets	802
28.7	Label – Ausgabe von Text und Bild	803
28.8	Signale	804
28.9	Checkboxen und Radiobuttons	805
28.10	Auswahlliste (ComboBox)	808
28.11	Gemeinsame Operationen der Widgets	810
28.12	Spezielle Methoden eines Fensters	811
28.13	Events	813
28.14	Fonts	814
28.15	Stylesheets	816
28.16	Icons	819
28.17	Messageboxen	819
28.18	Timer	820
28.19	Das Qt-Layout unter der Lupe	822
	28.19.1 Absolute Positionierung und Größe	822
	28.19.2 Raster-Layout	824
	28.19.3 Form-Layout	825
28.20	Browser für jeden Zweck	827
	28.20.1 Die Klasse QWebEngineView	827
28.21	Ein Webbrowser mit Filter	828
28.22	Surfen mit Geschichte – der Verlauf einer Sitzung	830
28.23	Aufgaben	832
28.24	Lösungen	833
29	Multimediaanwendungen mit Qt	837
29.1	Kalender und Textfeld – ein digitales Tagebuch	837
	29.1.1 Programmierung	838
29.2	Kamerabilder	843
29.3	Dialoge	845
	29.3.1 Projekt: Ansichtskarte	847
29.4	Videoplayer	851
	29.4.1 Ein einfacher Videoplayer	851
	29.4.2 Videoplayer mit Playlist	855
	29.4.3 Regeln zur Änderung der Größe (Size Policy)	858
	29.4.4 Das Dashboard bei Mausbewegungen einblenden	859
29.5	Aufgaben	862
29.6	Lösungen	867
30	Rechnen mit NumPy	875
30.1	NumPy installieren	875
30.2	Arrays erzeugen	875
	30.2.1 Arrays	875

30.2.2	Matrizen und Vektoren.....	878
30.2.3	Zahlenfolgen.....	878
30.2.4	Zufallsarrays.....	879
30.2.5	Spezielle Arrays.....	880
30.3	Indizieren.....	881
30.4	Slicing.....	882
30.5	Arrays verändern.....	883
30.6	Arithmetische Operationen.....	885
30.7	Funktionen, die elementweise ausgeführt werden.....	886
30.8	Einfache Visualisierung.....	887
30.9	Matrizenmultiplikation mit dot().....	888
30.10	Array-Funktionen und Achsen.....	889
30.11	Projekt: Diffusion.....	891
30.12	Vergleiche.....	894
30.13	Projekt: Wolken am Himmel.....	894
30.14	Projekt: Wie versteckt man ein Buch in einem Bild?.....	897
30.15	Datenanalyse mit Histogrammen.....	900
30.16	Wie funktioniert ein Medianfilter?.....	903
30.17	Rechnen mit SciPy.....	906
30.17.1	Lineare Gleichungssysteme lösen.....	906
30.17.2	Integration.....	908
30.18	Aufgaben.....	909
30.19	Lösungen.....	912
31	Messdaten verarbeiten.....	917
31.1	Messwerte in einem Diagramm darstellen – Matplotlib und tkinter.....	917
31.1.1	Basisprojekt.....	917
31.1.2	Erweiterung: Den letzten Wert löschen.....	921
31.1.3	Das Aussehen eines Diagramms gestalten.....	923
31.2	Messwerte aus einem Multimeter lesen und darstellen.....	926
31.2.1	Vorbereitung.....	926
31.2.2	Werte auslesen.....	927
31.2.3	Welche Ziffern zeigt das Display des Multimeters?.....	930
31.3	Anzeige der Temperatur.....	934
31.4	Messreihen aufzeichnen.....	936
31.5	Aufgabe.....	939
31.6	Lösung.....	939
32	Parallele Datenverarbeitung.....	943
32.1	Was sind parallele Programme?.....	943
32.2	Prozesse starten und abbrechen.....	944
32.3	Funktionen in eigenen Prozessen starten.....	945
32.4	Prozesse zusammenführen – join().....	947
32.5	Wie können Prozesse Objekte austauschen?.....	948
32.5.1	Objekte als Argumente übergeben.....	948

32.5.2	Objekte über eine Pipe senden und empfangen	949
32.5.3	Objekte über eine Queue austauschen	950
32.6	Daten im Pool bearbeiten	951
32.6.1	Mit dem Pool geht's schneller – ein Zeitexperiment	951
32.6.2	Forschen mit Big Data aus dem Internet	953
32.7	Synchronisation	956
32.8	Produzenten und Konsumenten	958
32.8.1	Sprücheklopfer	959
32.9	Aufgaben	961
32.10	Lösungen	962
33	Django	967
33.1	Django aus der Vogelperspektive	967
33.2	Ein Projekt anlegen	968
33.2.1	Den Server starten	970
33.2.2	Eine neue Applikation anlegen	970
33.2.3	Startseite und View einrichten	971
33.3	Datenbankanbindung	974
33.4	Modelle erstellen	974
33.5	Modelle aktivieren	976
33.6	In der Python-Shell die Datenbank bearbeiten	979
33.6.1	Objekte durch Aufruf der Klasse erzeugen	980
33.6.2	Auf Attribute eines Objekts zugreifen	981
33.6.3	Objekte finden	981
33.6.4	Objekte erzeugen und Beziehungen herstellen	982
33.6.5	Den Beziehungsmanager nutzen	983
33.6.6	Objekte löschen	983
33.7	Django-Modelle unter der Lupe	984
33.7.1	Grenzwerte	984
33.7.2	Leere Felder	985
33.7.3	Voreingestellte Werte	985
33.7.4	Einmaligkeit	985
33.7.5	Auswahlmöglichkeiten	985
33.8	Der Manager unter der Lupe – Objekte erzeugen und suchen	986
33.8.1	Objekte erzeugen	986
33.8.2	Objekte finden	986
33.8.3	Mehrere Bedingungen	987
33.8.4	Suchen über Beziehungen	988
33.8.5	Weitere Suchmethoden	988
33.9	Administration	989
33.9.1	Eine Applikation der Website-Verwaltung zugänglich machen	991
33.10	Views einrichten – die Grundstruktur	995
33.10.1	Was sind Views?	995
33.10.2	Funktionen für Views	995
33.10.3	URL-Patterns	996

33.11	View-Funktionen erweitern.	997
33.11.1	Startseite	998
33.11.2	Auflistung der Ideen zu einer Frage – question_index	1001
33.11.3	Die Templates verbessern: Namen statt expliziter URLs	1003
33.12	Interaktive Webseiten – Views mit Formularen	1004
33.12.1	Eingabe einer neuen Frage	1004
33.12.2	Eingabe einer neuen Idee	1010
33.12.3	View-Funktion für das Speichern einer neuen Idee	1012
33.12.4	Fertig!	1013
33.13	Die nächsten Schritte	1013
33.14	Aufgabe	1014
33.15	Lösung	1015
A	Anhang	1019
A.1	Codierung von Sonderzeichen in HTML	1019
A.2	Quellen im WWW	1019
A.3	Standardfunktionen und Standardklassen	1020
A.4	Mathematische Funktionen	1022
A.4.1	Das Modul math	1022
A.4.2	Das Modul random	1023
A.5	EBNF-Grammatik	1024
B	Glossar	1029
C	Download der Programmbeispiele	1043
D	Ein Python-Modul veröffentlichen: PyPI	1045
D.1	Bei PyPI und TestPyPI registrieren	1046
D.2	Ein Paket für die Veröffentlichung vorbereiten	1047
D.2.1	Die Programmdatei setup.py	1047
D.2.2	Die Lizenz	1048
D.2.3	Die Datei README.txt	1049
D.2.4	Die Datei __init__.py	1050
D.3	Das Paket auf PyPI veröffentlichen	1050
D.3.1	Das Paket aktualisieren	1051
	Stichwortverzeichnis	1053



Einleitung

Warum Python?

Es gibt triftige Argumente für die Verwendung der Programmiersprache Python.

- Python ist einfach. Man könnte auch sagen minimalistisch. Auf Sprachelemente, die nicht unbedingt notwendig sind, wurde verzichtet. Mit Python kann man kurze Programme schreiben, die viel leisten.
- Python besitzt einen interaktiven Modus. Sie können einzelne Befehle direkt eingeben und ihre Wirkung beobachten. Python unterstützt das Experimentieren und Ausprobieren. Das erleichtert das Erlernen neuer Programmierkonzepte und hilft vor allem Anfängern bei den ersten »Gehversuchen«.
- Dennoch ist Python kein Spielzeug. Zusammen mit vielen Zusatzkomponenten, sogenannten Modulen, ist es eine sehr mächtige Programmiersprache.
- Python ist nichtkommerziell. Alle Software, die Sie benötigen, ist kostenlos und für jede Plattform verfügbar.
- Hinter Python steht eine wachsende internationale Community aus Wissenschaftlern und Praktikern, die die Sprache pflegen und weiterentwickeln.

Python 3

Im Jahre 2008 fand in der Python-Welt eine kleine Revolution statt. Python 3 wurde veröffentlicht. Eine neue Version, die mit den Vorgängerversionen 2.X nicht mehr kompatibel ist. Ein Programm, das z.B. in Python 2.5 geschrieben worden ist, läuft (in der Regel) nicht mehr mit einem Python-3-Interpreter. Das ist natürlich schade, war aber notwendig, weil es einige sehr tief gehende Änderungen gab. Doch das neue Python 3 ist noch konsistenter und führt zu schönerem Programmtext als die früheren Versionen.

An wen wendet sich dieses Buch?

Dieses Buch ist für jeden, der die Programmierung mit Python lernen möchte. Besondere Vorkenntnisse werden nicht erwartet. Für die hinteren Kapitel ist es allerdings hilfreich, wenn man sich mit HTML auskennt. Das Buch wendet sich sowohl an Anfänger als auch an Leserinnen und Leser, die bereits mit einer höheren Programmiersprache vertraut sind, und ihr Wissen erweitern und vertiefen wollen. Für Neulinge gibt es zahlreiche Passagen, in denen grundlegende Konzepte anschaulich erklärt werden. Insbesondere das erste Kapitel ist zum überwiegenden Teil eine allgemeine Einführung für diejenigen, die sich bisher noch nie ausführlicher mit der Computertechnik beschäftigt haben. Wenn Sie sich eher zu

den Fortgeschrittenen zählen, dürfen Sie getrost diese Textabschnitte überspringen und sich dem zuwenden, das Sie interessiert.

Auf der anderen Seite enthält das Buch auch Stellen, die eine Herausforderung darstellen. Einige Abschnitte tragen Überschriften, die mit *Hintergrund:* oder *Vertiefung:* beginnen. Sie enthalten Ausblicke und Hintergrundinformationen oder gehen vertiefend auf speziellere Aspekte der jeweiligen Thematik ein, die nicht jeden interessieren.

Generell ist der Theorieanteil dieses Buches gering. Die praktische Arbeit steht im Vordergrund. In der Regel ist es möglich, theoretische Passagen (wie die über formale Grammatiken) zu überspringen, wenn man nun gar nicht damit zurechtkommt. Alle wichtigen Dinge werden zusätzlich auch auf anschauliche Weise erklärt. Und Sie werden erleben, dass beim Nachvollziehen und praktischen Ausprobieren der Programmbeispiele auch zunächst schwierig erscheinende Konzepte verständlich werden. Lassen Sie sich also nicht abschrecken.

Inhalt und Aufbau

Im Zentrum steht die Kunst der Programmentwicklung nach dem objektorientierten Paradigma. Dabei machen wir einen Rundgang durch verschiedene Gebiete der Informatik. Wir werfen einen Blick hinter die Kulissen von Software-Systemen, die Sie als Anwender aus dem Alltag kennen. Wie gestaltet man eine grafische Benutzeroberfläche? Wie funktioniert E-Mail? Wie programmiert man einen Chatroom? Darüber hinaus werden eine Reihe fundamentaler Ideen der Informatik angesprochen. Das Buch orientiert sich an den üblichen Curricula von Universitätskursen zur Einführung in die Programmierung. In vielen Fällen dürfte es deshalb eine sinnvolle Ergänzung zu einem Vorlesungsskript sein.

Dieses Buch ist so angelegt, dass man es von vorne nach hinten lesen kann. Wir fangen mit einfachen Dingen an und nachfolgende Kapitel knüpfen an den vorhergehenden Inhalt an. Idealerweise sollte jeder Begriff bei seiner ersten Verwendung erklärt werden. Doch lässt sich dieses Prinzip nur schwer in Perfektion umsetzen. Manchmal gehen wir von einem intuitiven Vorverständnis aus und erläutern die Begrifflichkeit erst kurz darauf ausführlich.

Im vorderen Teil des Buches finden Sie an verschiedenen Stellen Hinweise zum Programmierstil und zu typischen Fehlern. Am Ende jedes Kapitels gibt es Übungsaufgaben, die in der Regel nach Schwierigkeitsgrad sortiert sind. Einige Programmieraufgaben sind so komplex, dass man sie (insbesondere als Anfänger) eigentlich gar nicht eigenständig lösen kann. Sie sind dann eher als Erweiterung gedacht und es wurde ins Kalkül gezogen, dass Sie »mögeln« und während der Bearbeitung in die Lösung gucken.

Unterkapitel, deren Überschriften mit dem Wort »Vertiefung« beginnen, wenden sich an besonders interessierte Leser und können in der Regel übersprungen werden.

Der vordere Teil des Buches befasst sich mit den grundlegenden Konzepten der Programmierung mit Python. Herausgestellt werden die syntaktischen Besonderheiten gegenüber anderen Programmiersprachen. Sie finden an verschiedenen Stellen Hinweise zum Programmierstil und zu typischen Fehlern. Angesprochen werden unter anderem folgende Punkte:

- Aufbau von Anweisungen in einem Python Programm
- Umgang mit der Standard-Entwicklungsumgebung IDLE

- Standard-Datentypen
- Modellieren mit Datenstrukturen: Tupel, Listen, Dictionaries, Mengen
- Kontrollstrukturen: Wiederholungen, Verzweigungen, Abfangen von Ausnahmen (try ... except)
- Funktionen: Arten von Parametern, Voreinstellungen, Lambda-Ausdrücke, Rekursion, Docstrings
- Ein- und Ausgabe: Dateien, pickle
- Konzepte der Objektorientierung: Klassen, Objekte, Vererbung, statische Methoden, Polymorphie, Properties
- Techniken der objektorientierten Modellierung: Analyse (OOA) und Design (OOD), UML, Objekt- und Klassendiagramme, Assoziationen
- Modularisieren
- Verarbeitung von Zeichenketten: String-Methoden, Codierung und Decodierung, Formatierung, reguläre Ausdrücke, Sprachsynthese, Chat-Bots
- Systemfunktionen: Schnittstelle zum Betriebssystem, Datum und Zeit
- Grundprinzipien der Gestaltung von grafischen Benutzeroberflächen mit tkinter: Widgets, Event-Verarbeitung, Layout, Threads
- Debugging-Techniken

Im hinteren Teil des Buches werden die Kapitel immer spezieller. Hier kommen dann gelegentlich auch Module von Drittanbietern ins Spiel, die nicht zur Standardinstallation von Python gehören (z.B. PIL, PyQt, NumPy). Sie müssen erst heruntergeladen und installiert werden. Zu diesen spezielleren Themen gehören:

- Internet-Programmierung: CGI-Skripte, WSGI, Webserver, E-Mail-Clients
- Datenbanken und XML
- Testen und Performance-Analyse: doctest, unittest
- Benutzeroberflächen für Multimedia-Anwendungen mit PyQt: Video-Player, Webbrowser, Kalender
- Wissenschaftliches Rechnen mit NumPy und SciPy: Arrays, Vektoren und Matrizen, digitale Bildbearbeitung, Datenvisualisierung, lineare Gleichungssysteme, Integralrechnung
- Parallele Datenverarbeitung: Prozesse und Synchronisation, Queues, Pipes, Pools
- Messdaten eines externen digitalen Multimeters erfassen und verarbeiten
- Webentwicklung mit Django.

Hinweise zur Typographie

Achten Sie beim Lesen auf den Schrifttyp. Formale Texte, wie Python-Programmtext, Funktions- und Variablenamen, Operatoren, Grammatik. Regeln, Zahlen und mathematische Ausdrücke, werden in einem Zeichenformat mit fester Breite gesetzt. Beispiele:

```
x = y + 1
print()
```

In solchen formalen Texten tauchen gelegentlich Wörter auf, die kursiv gesetzt sind. Hierbei handelt es sich um Platzhalter, die man nicht Buchstabe für Buchstabe aufschreibt, sondern z.B. durch Zahlen oder andere Zeichenfolgen ersetzt. Beispiel:

```
range(zahl)
```

Hier bezeichnet *zahl* eine (ganze) Zahl. Ein korrekter Aufruf der Funktion `range()` lautet z.B. `range(10)`, während `range(zahl)` zu Problemen führen kann.

In Programmtexten sind wichtige Passagen fett gedruckt, damit man sie schneller finden kann.

Programmbeispiele

Das Buch enthält zahlreiche Programmbeispiele, die zum Ausprobieren, Nachmachen und Weiterentwickeln ermuntern sollen. Sie können alle Skripte und einige zusätzliche Dateien als ZIP-Archiv von der Website des mitp-Verlages herunterladen. Der URL ist:

<http://www.mitp.de/0544>

Klicken Sie im Kasten **DOWNLOADS** auf den Link **PROGRAMMBEISPIELE**.

Außerdem sind die Programmbeispiele in einem GitHub-Repository veröffentlicht. URL:

<https://github.com/mweigend/python3/>

Weitere Hinweise zum Download finden Sie im Anhang C.

Beim Design der Beispiele wurde darauf geachtet, dass sie möglichst kurz und übersichtlich sind. Häufig sind die Skripte Spielzeugversionen richtiger Software, die man im Alltag zu sinnvollen Dingen nutzen kann. Sie sind Modelle – etwa so wie Häuser aus Legosteinen. Modelle richtiger Häuser sind. Sie sind auf das Wesentliche reduziert und sollen nur bestimmte Aspekte verdeutlichen. Sie genügen deshalb nicht den Qualitätsanforderungen, die man üblicherweise an professionelle Software stellt, aber sie dienen vielleicht als Anregung und Inspiration für eigene Projekte.

Grundlagen

Bitte noch etwas Geduld! Im ersten Kapitel bleibt der Computer noch ausgeschaltet. Hier wird zunächst eine anschauliche Vorstellung von einigen Grundideen der Programmierung vermittelt. Sie helfen, den Rest des Buches besser zu verstehen. Im Mittelpunkt stehen folgende Fragen:

- Was sind Programme und Algorithmen?
- Worin unterscheiden sich Programmierparadigmen?
- Was ist die Philosophie der objektorientierten Programmierung?

1.1 Was ist Programmieren?

Es ist eigentlich ganz einfach: Programmieren ist das Schreiben eines Programms. Nun gibt es den Begriff »Programm« auch in unserer Alltagssprache – fernab von jeder Computertechnik. Sie kennen Fernseh- und Kinoprogramme, planen ein Programm für Ihre Geburtstagsparty, genießen im Urlaub vielleicht Animationsprogramme (sofern Sie nichts Besseres zu tun haben) und lesen als gewissenhafter Staatsbürger vor den Bundestagswahlen Parteiprogramme. In diesen Zusammenhängen versteht man unter einem Programm eigentlich recht unterschiedliche Dinge: Ein Parteiprogramm ist so etwas wie ein strukturiertes Konzept politischer Ziele, ein Kinoprogramm ein Zeitplan für Filmvorstellungen und ein Animationsprogramm ein Ablauf von Unterhaltungsveranstaltungen.

In der Informatik – der Wissenschaft, die hinter der Programmierertechnik steht – ist der Begriff Programm natürlich enger und präziser gefasst. Allerdings gibt es auch hier unterschiedliche Sichtweisen.

Die älteste und bekannteste Definition basiert auf dem Begriff *Algorithmus*. Grob gesprochen ist ein Algorithmus eine Folge von Anweisungen (oder militärisch formuliert: Befehlen), die man ausführen muss, um ein Problem zu lösen. Unter einem Programm versteht man in dieser Sichtweise einen Algorithmus,

- der in einer Sprache geschrieben ist, die auch Maschinen verstehen können (Programmiersprache), und
- der das Verhalten von Maschinen steuert.

Daraus folgt: Wer ein Computerprogramm schreibt, muss zumindest zwei Dinge tun:

- Er oder sie muss einen Algorithmus erfinden, der in irgendeiner Weise nützlich ist und zum Beispiel bei der Lösung eines Problems helfen kann.
- Der Algorithmus muss fehlerfrei in einer Programmiersprache formuliert werden. Man spricht dann von einem Programmtext.

Ziel einer Programmentwicklung ist korrekter Programmtext.

1.2 Hardware und Software

Ein Computer ist eine universelle Maschine, deren Verhalten durch ein Programm bestimmt wird. Ein Computersystem besteht aus Hardware und Software. Ersteres ist das englische Wort für »Eisenwaren« und meint alle Komponenten des Computers, die man anfassen kann – Arbeitsspeicherbausteine, Prozessor, Peripheriespeicher (Festplatte, Diskette, CD), Monitor, Tastatur usw. Software dagegen ist ein Kunstwort, das als Pendant zu Hardware gebildet wurde. Mit Software bezeichnet man die Summe aller Programme, die die Hardware steuern.

Man kann die gesamte Software eines Computers grob in zwei Gruppen aufteilen:

Das *Betriebssystem* regelt den Zugriff auf die Hardware des Computers und verwaltet Daten, die im Rechner gespeichert sind. Es stellt eine Umgebung bereit, in der Benutzer Programme ausführen können. Bekannte Betriebssysteme sind Unix, MS Windows oder macOS. Python-Programme laufen unter allen drei genannten Betriebssystemen. Man nennt sie deshalb portabel.

Anwendungs- und Systemsoftware dient dazu, spezifische Probleme zu lösen. Ein Textverarbeitungsprogramm z.B. unterstützt das Erstellen, Verändern und Speichern von Textdokumenten. Anwendungssoftware ist also auf Bedürfnisse des Benutzers ausgerichtet, während das Betriebssystem nur für ein möglichst störungsfreies und effizientes Zusammenspiel der verschiedenen Komponenten des Computersystems sorgt.

Ein Computersystem wird häufig durch ein Schichtenmodell wie in Abbildung 1.1 beschrieben. Die unterste Schicht ist die Computer-Hardware, darüber liegt das Betriebssystem und zuoberst befinden sich schließlich die Anwendungs- und Systemprogramme, die eine Benutzungsschnittstelle enthalten. Nur über diese oberste Software-Schicht kommunizieren Menschen mit einem Computersystem.

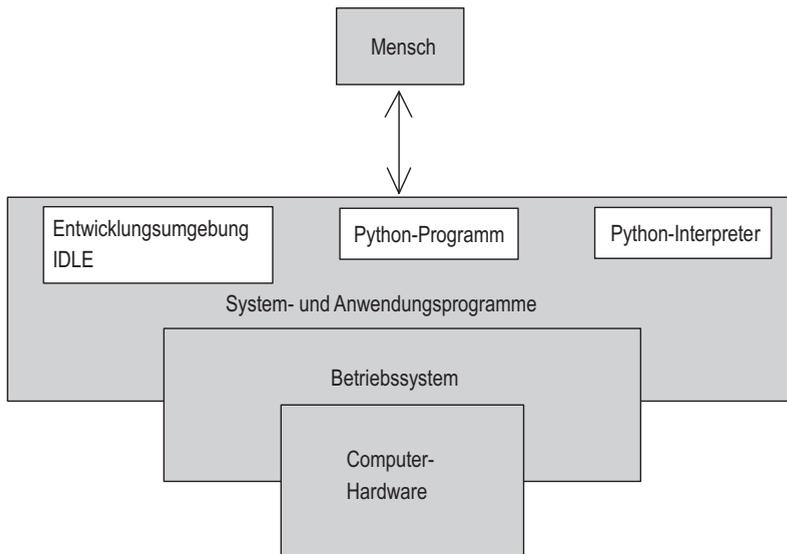


Abb. 1.1: Komponenten eines Computer-Systems