

```
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 /= 10**6
```

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

```
# take prefix to account
```

```
if d[11][4] == "1": n /= 10**6
```

```
elif d[11][5] == "1": n /= 10**9
```

```
elif d[11][6] == "1": n /= 10**12
```

```
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)

# Inhaltsverzeichnis

## Python 3

## Impressum

## Einleitung

Warum Python?

Python 3

An wen wendet sich dieses Buch?

Inhalt und Aufbau

Hinweise zur Typographie

Programmbeispiele

## Kapitel 1: Grundlagen

1.1 Was ist Programmieren?

1.2 Hardware und Software

1.3 Programm als Algorithmus

1.4 Syntax und Semantik

1.5 Interpreter und Compiler

1.6 Programmierparadigmen

1.7 Objektorientierte Programmierung

1.7.1 Strukturelle Zerlegung

1.7.2 Die Welt als System von Objekten

1.7.3 Objekte besitzen Attribute und beherrschen Methoden

1.7.4 Objekte sind Instanzen von Klassen

1.8 Hintergrund: Geschichte der objektorientierten Programmierung

1.9 Aufgaben

1.10 Lösungen

## **Kapitel 2: Der Einstieg - Python im interaktiven Modus**

2.1 Python installieren

2.2 Python im interaktiven Modus

2.2.1 Start des Python-Interpreters in einem Konsolenfenster

2.2.2 Die IDLE-Shell

2.2.3 Die ersten Python-Befehle ausprobieren

2.2.4 Hotkeys

2.3 Objekte

2.4 Namen

2.5 Hintergrund: Syntax-Regeln für Bezeichner

2.6 Schlüsselwörter

2.7 Anweisungen

2.7.1 Ausdruckanweisungen

2.7.2 Import-Anweisungen

2.7.3 Zuweisungen

2.7.4 Erweiterte Zuweisungen

2.7.5 Hintergrund: Dynamische Typisierung

2.8 Aufgaben

2.9 Lösungen

## **Kapitel 3: Python-Skripte**

3.1 Ausprobieren, nachmachen, besser machen!

- 3.2 Skripte editieren und ausführen mit IDLE
- 3.3 Ausführen eines Python-Skripts
- 3.4 Kommentare
- 3.5 Die Zeilenstruktur von Python-Programmen
- 3.6 Das EVA-Prinzip
- 3.7 Phasen der Programmentwicklung
- 3.8 Guter Programmierstil
- 3.9 Hintergrund: Die Kunst des Fehlerfindens
- 3.10 Weitere Entwicklungsumgebungen für Python
  - 3.10.1 Thonny – eine Entwicklungsumgebung für Python-Einsteiger
  - 3.10.2 Python in der Cloud
  - 3.10.3 Jupyter Notebook und Google Colab
  - 3.10.4 Entwicklungsumgebungen für Profis
- 3.11 Aufgaben
- 3.12 Lösungen

## **Kapitel 4: Standard-Datentypen**

- 4.1 Daten als Objekte
- 4.2 Fundamentale Datentypen im Überblick
- 4.3 Typen und Klassen
- 4.4 NoneType
- 4.5 Wahrheitswerte – der Datentyp bool
- 4.6 Ganze Zahlen
- 4.7 Gleitkommazahlen
- 4.8 Komplexe Zahlen
- 4.9 Arithmetische Operatoren für Zahlen

## 4.10 Sequenzen

4.10.1 Zeichenketten (Strings)

4.10.2 Bytestrings

4.10.3 Tupel

4.10.4 Liste

4.10.5 Bytearray

4.10.6 Einige Grundoperationen für Sequenzen

4.10.7 Veränderbare und unveränderbare Sequenzen

## 4.11 Mengen

## 4.12 Dictionaries

## 4.13 Typumwandlungen

4.13.1 int()

4.13.2 float()

4.13.3 complex()

4.13.4 bool()

4.13.5 str()

4.13.6 dict(), list() und tuple()

## 4.14 Aufgaben

## 4.15 Lösungen

# **Kapitel 5: Kontrollstrukturen**

## 5.1 Einfache Bedingungen

5.1.1 Vergleiche

5.1.2 Zugehörigkeit zu einer Menge (in, not in)

5.1.3 Beliebige Ausdrücke als Bedingungen

## 5.2 Zusammengesetzte Bedingungen – logische Operatoren

5.2.1 Negation (not)

- 5.2.2 Konjunktion (and)
- 5.2.3 Disjunktion (or)
- 5.2.4 Formalisierung von Bedingungen
- 5.2.5 Hinweis zum Programmierstil
- 5.3 Programmverzweigungen (bedingte Anweisungen)
  - 5.3.1 Einseitige Verzweigung (if)
  - 5.3.2 Zweiseitige Verzweigung (if-else)
  - 5.3.3 Mehrfache Fallunterscheidung (elif)
  - 5.3.4 Bedingte Ausdrücke
- 5.4 Bedingte Wiederholung (while)
  - 5.4.1 Endlosschleifen
- 5.5 Iteration über eine Kollektion (for)
  - 5.5.1 Zählschleifen – Verwendung von range()
  - 5.5.2 Verschachtelte Iterationen
  - 5.5.3 Vertiefung: Iterative Berechnung rekursiver Folgen
- 5.6 Abbruch einer Schleife mit break
  - 5.6.1 Abbruch eines Schleifendurchlaufs mit continue
- 5.7 Abfangen von Ausnahmen mit try
  - 5.7.1 try...except
- 5.8 Aufgaben
- 5.9 Lösungen

## **Kapitel 6: Funktionen**

- 6.1 Aufruf von Funktionen
- 6.2 Definition von Funktionen
- 6.3 Schrittweise Verfeinerung

- 6.4 Ausführung von Funktionen
  - 6.4.1 Globale und lokale Namen
  - 6.4.2 Seiteneffekte – die global-Anweisung
  - 6.4.3 Parameterübergabe
- 6.5 Voreingestellte Parameterwerte
  - 6.5.1 Schlüsselwort-Argumente
- 6.6 Funktionen mit beliebiger Anzahl von Parametern
- 6.7 Lokale Funktionen
- 6.8 Rekursive Funktionen
- 6.9 Experimente zur Rekursion mit der Turtle-Grafik
  - 6.9.1 Turtle-Befehle im interaktiven Modus
  - 6.9.2 Eine rekursive Spirale
  - 6.9.3 Baumstrukturen
  - 6.9.4 Künstlicher Blumenkohl – selbstähnliche Bilder
- 6.10 Rekursive Zahlenfunktionen
- 6.11 Hintergrund: Wie werden rekursive Funktionen ausgeführt?
  - 6.11.1 Execution Frames
  - 6.11.2 Rekursionstiefe
- 6.12 Funktionen als Objekte
  - 6.12.1 Hintergrund: Typen sind keine Funktionen
- 6.13 Lambda-Formen
- 6.14 Funktionsannotationen: Typen zuordnen
- 6.15 Hinweise zum Programmierstil
  - 6.15.1 Allgemeines
  - 6.15.2 Funktionsnamen
  - 6.15.3 Kommentierte Parameter
  - 6.15.4 Docstrings

6.16 Aufgaben

6.17 Lösungen

## **Kapitel 7: Sequenzen, Mengen und Generatoren**

7.1 Gemeinsame Operationen für Sequenzen

7.1.1 Zugriff auf Elemente einer Sequenz

7.1.2 Slicing von Sequenzen

7.1.3 Auspacken (unpacking)

7.2 Vertiefung: Rekursive Funktionen für Sequenzen

7.2.1 Rekursives Summieren

7.2.2 Rekursive Suche

7.3 Tupel

7.4 Listen

7.4.1 Eine Liste erzeugen

7.4.2 Eine Liste verändern

7.4.3 Flache und tiefe Kopien

7.4.4 Listen sortieren

7.4.5 Binäre Suche in einer sortierten Liste

7.4.6 Zwei Sortierverfahren im Vergleich

7.4.7 Modellieren mit Listen – Beispiel: die Charts

7.5 Generatoren

7.5.1 Generatorausdrücke

7.5.2 Generatorfunktionen

7.5.3 Iteratoren

7.5.4 Verwendung von Generatoren

7.6 Mengen

7.6.1 Operationen für Mengen

7.6.2 Modellieren mit Mengen – Beispiel: Graphen

7.7 Aufgaben

7.8 Lösungen

## **Kapitel 8: Dictionaries**

8.1 Operationen für Dictionaries

8.2 Wie erstellt man ein Dictionary?

8.2.1 Definition mit einem Dictionary-Display

8.2.2 Schrittweiser Aufbau eines Dictionarys

8.2.3 Ein Dictionary aus anderen Dictionaries zusammensetzen - update()

8.3 Zugriff auf Daten in einem Dictionary

8.3.1 Vergebliche Zugriffsversuche

8.4 Praxisbeispiel: Vokabeltrainer

8.5 Typische Fehler

8.6 Aufgaben

8.7 Lösungen

## **Kapitel 9: Ein- und Ausgabe**

9.1 Streams

9.1.1 Die Rolle der Streams bei E/A-Operationen

9.1.2 Was ist ein Stream?

9.1.3 Eine Datei öffnen

9.1.4 Speichern einer Zeichenkette

9.1.5 Laden einer Zeichenkette aus einer Datei

9.1.6 Absolute und relative Pfade

9.1.7 Zwischenspeichern, ohne zu schließen

9.1.8 Zugriff auf Streams (lesen und schreiben)

9.2 Mehr Zuverlässigkeit durch try- und with-Anweisungen

- 9.2.1 try...finally
- 9.2.2 with-Anweisungen
- 9.3 Objekte speichern mit pickle
  - 9.3.1 Funktionen zum Speichern und Laden
- 9.4 Die Streams sys.stdin und sys.stdout
- 9.5 Ausgabe von Werten mit der print()-Funktion
  - 9.5.1 Anwendung: Ausgabe von Tabellen
- 9.6 Kommandozeilen-Argumente (Optionen)
  - 9.6.1 Zugriff auf Optionen
  - 9.6.2 Beispiel
  - 9.6.3 Skripte mit Optionen testen
- 9.7 Aufgaben
- 9.8 Lösungen

## **Kapitel 10: Definition eigener Klassen**

- 10.1 Klassen und Objekte
- 10.2 Definition von Klassen
- 10.3 Objekte (Instanzen)
- 10.4 Zugriff auf Attribute – Sichtbarkeit
  - 10.4.1 Öffentliche Attribute
  - 10.4.2 Private Attribute
  - 10.4.3 Properties
  - 10.4.4 Dynamische Erzeugung von Attributen
- 10.5 Methoden
  - 10.5.1 Polymorphismus – überladen von Operatoren
  - 10.5.2 Vertiefung: Objekte ausführbar machen – die Methode `__call__()`
  - 10.5.3 Statische Methoden

- 10.6 Abstraktion, Verkapselung und Geheimnisprinzip
  - 10.6.1 Abstraktion
  - 10.6.2 Verkapselung
  - 10.6.3 Geheimnisprinzip
- 10.7 Vererbung
  - 10.7.1 Spezialisierungen
  - 10.7.2 Beispiel: Die Klasse Konto – eine Spezialisierung der Klasse Geld
  - 10.7.3 Vertiefung: Standardklassen als Basisklassen
- 10.8 Hinweise zum Programmierstil
  - 10.8.1 Schreibweise
  - 10.8.2 Sichtbarkeit
  - 10.8.3 Dokumentation von Klassen
- 10.9 Typische Fehler
  - 10.9.1 Versehentliches Erzeugen neuer Attribute
  - 10.9.2 Verwechseln von Methoden und Attributen
- 10.10 Aufgaben
- 10.11 Lösungen

## **Kapitel 11: Klassen wiederverwenden - Module**

- 11.1 Testen einer Klasse in einem lauffähigen Stand-alone-Skript
- 11.2 Module speichern und importieren
- 11.3 Den Zugang zu einem Modul sicherstellen
  - 11.3.1 Erweitern der Verzeichnisliste sys.path
  - 11.3.2 Anwendungsbeispiel: Eine interaktive Testumgebung
  - 11.3.3 Kompilieren von Modulen

11.4 Programmierstil: Verwendung und Dokumentation von Modulen

## **Kapitel 12: Objektorientiertes Modellieren**

12.1 Phasen einer objektorientierten Software-Entwicklung

12.1.1 Objektorientierte Analyse (OOA)

12.1.2 Objektorientierter Entwurf (OOD)

12.1.3 Objektorientierte Programmierung (OOP)

12.2 Beispiel: Modell eines Wörterbuchs

12.2.1 OOA: Entwicklung einer Klassenstruktur

12.2.2 OOD: Entwurf einer Klassenstruktur zur Implementierung in Python

12.2.3 OOP: Implementierung der Klassenstruktur

12.3 Assoziationen zwischen Klassen

12.3.1 Reflexive Assoziationen

12.3.2 Aggregation

12.4 Beispiel: Management eines Musicals

12.4.1 OOA

12.4.2 OOD

12.4.3 OOP

12.5 Aufgaben

12.6 Lösungen

## **Kapitel 13: Textverarbeitung**

13.1 Standardmethoden zur Verarbeitung von Zeichenketten

13.1.1 Formatieren

13.1.2 Schreibweise

13.1.3 Tests

13.1.4 Entfernen und Aufspalten

13.1.5 Suchen und Ersetzen

## 13.2 Codierung und Decodierung

13.2.1 Platonische Zeichen und Unicode

13.2.2 Vertiefung: Zeichenketten durch Bytefolgen darstellen

## 13.3 Automatische Textproduktion

13.3.1 Texte mit variablen Teilen - Anwendung der String-Methode format()

13.3.2 Vertiefung: Eine Tabelle erstellen

13.3.3 Mahnbriefe

13.3.4 Textuelle Repräsentation eines Objekts

13.3.5 F-Strings

## 13.4 Analyse von Texten

13.4.1 Chat-Bots

13.4.2 Textanalyse mit einfachen Vorkommenstests

## 13.5 Reguläre Ausdrücke

13.5.1 Die Funktion findall() aus dem Modul re

13.5.2 Aufbau eines regulären Ausdrucks

13.5.3 Objekte für reguläre Ausdrücke

13.5.4 Strings untersuchen mit search()

13.5.5 Textpassagen extrahieren mit findall()

13.5.6 Zeichenketten zerlegen mit split()

13.5.7 Teilstrings ersetzen mit sub()

13.5.8 Match-Objekte

## 13.6 Den Computer zum Sprechen bringen - Sprachsynthese

13.6.1 Buchstabieren

13.6.2 Den Klang der Stimme verändern

13.7 Aufgaben

13.8 Lösungen

## **Kapitel 14: Systemfunktionen**

14.1 Das Modul sys – die Schnittstelle zum Laufzeitsystem

14.1.1 Informationen über die aktuelle Systemumgebung

14.1.2 Standardeingabe und -ausgabe

14.1.3 Die Objektverwaltung beobachten mit `getrefcount()`

14.1.4 Ausführung eines Skripts beenden

14.2 Das Modul os – die Schnittstelle zum Betriebssystem

14.2.1 Dateien und Verzeichnisse suchen

14.2.2 Hintergrund: Zugriffsrechte abfragen und ändern (Windows und Unix)

14.2.3 Dateien und Verzeichnisse anlegen und modifizieren

14.2.4 Merkmale von Dateien und Verzeichnissen abfragen

14.2.5 Pfade verarbeiten

14.2.6 Hintergrund: Umgebungsvariablen

14.2.7 Systematisches Durchlaufen eines Verzeichnisbaumes

14.3 Datum und Zeit

14.3.1 Funktionen des Moduls `time`

14.3.2 Sekundenformat

14.3.3 Zeit-Tupel

14.3.4 Zeitstrings

- 14.3.5 Einen Prozess unterbrechen mit sleep()
- 14.4 Zeitberechnungen mit dem Modul datetime
  - 14.4.1 Die Klasse datetime
  - 14.4.2 Die Zeitzone
  - 14.4.3 Die Klasse timedelta
- 14.5 Aufgaben
- 14.6 Lösungen

## **Kapitel 15: Grafische Benutzungsoberflächen mit tkinter**

- 15.1 Ein einführendes Beispiel
- 15.2 Einfache Widgets
- 15.3 Die Master-Slave-Hierarchie
- 15.4 Optionen der Widgets
  - 15.4.1 Optionen bei der Instanziierung setzen
  - 15.4.2 Widget-Optionen nachträglich konfigurieren
  - 15.4.3 Fonts
  - 15.4.4 Farben
  - 15.4.5 Rahmen
  - 15.4.6 Die Größe eines Widgets
  - 15.4.7 Leerraum um Text
- 15.5 Gemeinsame Methoden der Widgets
- 15.6 Die Klasse Tk
- 15.7 Die Klasse Button
- 15.8 Die Klasse Label
  - 15.8.1 Dynamische Konfiguration der Beschriftung
  - 15.8.2 Verwendung von Kontrollvariablen
- 15.9 Die Klasse Entry

- 15.10 Die Klasse Radiobutton
- 15.11 Die Klasse Checkbutton
- 15.12 Die Klasse Scale
- 15.13 Die Klasse Frame
- 15.14 Aufgaben
- 15.15 Lösungen

## **Kapitel 16: Layout**

- 16.1 Der Packer
- 16.2 Layout-Fehler
- 16.3 Raster-Layout
- 16.4 Vorgehensweise bei der GUI-Entwicklung
  - 16.4.1 Die Benutzungsoberfläche gestalten
  - 16.4.2 Funktionalität hinzufügen
- 16.5 Aufgaben
- 16.6 Lösungen

## **Kapitel 17: Grafik**

- 17.1 Die tkinter-Klasse Canvas
  - 17.1.1 Generierung grafischer Elemente - ID, Positionierung und Display-Liste
  - 17.1.2 Grafische Elemente gestalten
  - 17.1.3 Visualisieren mit Kreisdiagrammen
- 17.2 Die Klasse PhotoImage
  - 17.2.1 Eine Pixelgrafik erzeugen
  - 17.2.2 Fotos analysieren und verändern
- 17.3 Bilder in eine Benutzungsoberfläche einbinden
  - 17.3.1 Icons auf Schaltflächen

- 17.3.2 Hintergrundbilder
- 17.3.3 Hintergrund: Das PPM-Format
- 17.4 Die Python Imaging Library (PIL)
  - 17.4.1 Installation eines Moduls mit pip
  - 17.4.2 Mit PIL beliebige Bilddateien einbinden
  - 17.4.3 Steganografie – Informationen in Bildern verstecken
- 17.5 Aufgaben
- 17.6 Lösungen

## **Kapitel 18: Event-Verarbeitung**

- 18.1 Einführendes Beispiel
- 18.2 Event-Sequenzen
  - 18.2.1 Event-Typen
  - 18.2.2 Qualifizierer für Maus- und Tastatur-Events
  - 18.2.3 Modifizierer
- 18.3 Beispiel: Tastaturereignisse verarbeiten
- 18.4 Programmierung eines Eventhandlers
  - 18.4.1 Beispiel für eine Event-Auswertung
- 18.5 Bindemethoden
- 18.6 Aufgaben
- 18.7 Lösungen

## **Kapitel 19: Komplexe Benutzungsoberflächen**

- 19.1 Text-Widgets
  - 19.1.1 Methoden der Text-Widgets
- 19.2 Rollbalken (Scrollbars)
- 19.3 Menüs

- 19.3.1 Die Klasse Menu
- 19.3.2 Methoden der Klasse Menu
- 19.4 Texteditor mit Menüleiste und Pulldown-Menü
- 19.5 Dialogboxen
- 19.6 Applikationen mit mehreren Fenstern
- 19.7 Aufgaben
- 19.8 Lösungen

## **Kapitel 20: Threads**

- 20.1 Funktionen in einem Thread ausführen
- 20.2 Thread-Objekte erzeugen – die Klasse Thread
- 20.3 Aufgaben
- 20.4 Lösungen

## **Kapitel 21: Fehler finden und vermeiden**

- 21.1 Testen von Bedingungen
  - 21.1.1 Ausnahmen (Exceptions)
  - 21.1.2 Testen von Vor- und Nachbedingungen mit assert
  - 21.1.3 Vertiefung: Programmabstürze ohne Fehlermeldung
- 21.2 Debugging-Modus und optimierter Modus
- 21.3 Ausnahmen gezielt auslösen
- 21.4 Selbstdokumentation
- 21.5 Dokumentation eines Programmlaufs mit Log-Dateien
  - 21.5.1 Grundfunktionen
  - 21.5.2 Beispiel: Logging in der GUI-Programmierung

## 21.6 Vertiefung: Professionelles Arbeiten mit Logging

### 21.6.1 Logging-Levels

### 21.6.2 Logger-Objekte

### 21.6.3 Das Format der Logging-Meldungen konfigurieren

## 21.7 Debugging

### 21.7.1 Schaltflächen des Debug-Control-Fensters

### 21.7.2 Breakpoints

## **Kapitel 22: Dynamische Webseiten - CGI und WSGI**

### 22.1 Wie funktionieren dynamische Webseiten?

### 22.2 Wie spät ist es? Aufbau eines CGI-Skripts

#### 22.2.1 Die Ausgabe eines CGI-Skripts

#### 22.2.2 Wie ist ein CGI-Skript aufgebaut?

#### 22.2.3 Verwendung von Schablonen

#### 22.2.4 Aufruf mit dem Webbrowser

#### 22.2.5 Ein einfacher HTTP-Server

### 22.3 Kommunikation über interaktive Webseiten

#### 22.3.1 Aufbau eines HTML-Formulars

#### 22.3.2 Eingabekomponenten in einem HTML-Formular

### 22.4 Verarbeitung von Eingabedaten mit FieldStorage

### 22.5 Sonderzeichen handhaben

### 22.6 CGI-Skripte debuggen

### 22.7 Der Apache-Webserver

#### 22.7.1 Den Apache-Server installieren

#### 22.7.2 CGI-Skripte auf dem Apache-Server

### 22.8 Dynamische Webseiten mit WSGI

22.8.1 Einfacher geht's nicht: Ein Stand-alone-WSGI-Webserver mit wsgiref

22.9 mod\_wsgi

22.9.1 Installation

22.9.2 Vorbereitung

22.9.3 Den Apache-Server konfigurieren

22.9.4 Ein WSGI-Skript für den Apache-Server

22.9.5 Tipps zum Debuggen

22.9.6 Zugriff von einem entfernten Rechner im WLAN

22.10 Verarbeitung von Eingabedaten aus Formularen

22.11 Objektorientierte WSGI-Skripte – Beispiel: ein Chatroom

22.11.1 Die HTML-Seiten

22.11.2 Die Klassen für den Chatroom

22.11.3 Skript (Teil 2):

22.12 WSGI-Skripte mit Cookies

22.12.1 Besuche zählen

## **Kapitel 23: Internet-Programmierung**

23.1 Was ist ein Protokoll?

23.2 Übertragung von Dateien mit FTP

23.2.1 Das Modul ftplib

23.2.2 Navigieren und Downloaden

23.2.3 Ein Suchroboter für FTP-Server

23.3 Zugriff auf Webseiten mit HTTP und HTTPS

23.3.1 Automatische Auswertung von Webseiten

23.4 Zugriff auf Ressourcen im Internet über deren URL

23.4.1 Webseite herunterladen und verarbeiten

- 23.4.2 Projekt: Wie warm wird es heute?
- 23.4.3 Datei herunterladen und speichern
- 23.4.4 Projekt: Filme herunterladen

23.5 E-Mails senden mit SMTP

23.6 Aufgaben

23.7 Lösungen

## **Kapitel 24: Datenbanken**

24.1 Was ist ein Datenbanksystem?

24.2 Entity-Relationship-Diagramme (ER-Diagramme)

24.3 Relationale Datenbanken

24.4 Darstellung von Relationen als Mengen oder Dictionaries

24.5 Das Modul sqlite3

- 24.5.1 Beispiel: Telefonbuch

- 24.5.2 Eine Tabelle anlegen

- 24.5.3 Anfragen an eine Datenbank

- 24.5.4 Datensuche im interaktiven Modus

- 24.5.5 SQL-Anweisungen mit variablen Teilen

- 24.5.6 Vertiefung: SQL-Injection

24.6 Online-Redaktionssystem mit Datenbankanbindung

- 24.6.1 Objektorientierte Analyse (OOA)

- 24.6.2 Objektorientierter Entwurf des Systems (OOD)

- 24.6.3 Hintergrund: Authentifizieren mit SHA-256

- 24.6.4 Implementierung des Redaktionssystems mit Python (OOP)

24.7 Aufgaben

## 24.8 Lösungen

### **Kapitel 25: Testen und Tuning**

#### 25.1 Automatisiertes Testen

#### 25.2 Testen mit Docstrings – das Modul doctest

#### 25.3 Praxisbeispiel: Suche nach dem Wort des Jahres

#### 25.4 Klassen testen mit doctest

##### 25.4.1 Wie testet man eine Klasse?

##### 25.4.2 Normalisierte Whitespaces – doctest-Direktiven

##### 25.4.3 Ellipsen verwenden

##### 25.4.4 Dictionaries testen

#### 25.5 Gestaltung von Testreihen mit unittest

##### 25.5.1 Einführendes Beispiel mit einem Testfall

##### 25.5.2 Klassen des Moduls unittest

##### 25.5.3 Weiterführendes Beispiel

#### 25.6 Tuning

##### 25.6.1 Performance-Analyse mit dem Profiler

##### 25.6.2 Praxisbeispiel: Auswertung astronomischer Fotografien

##### 25.6.3 Performance-Analyse und Tuning

#### 25.7 Aufgaben

#### 25.8 Lösungen

### **Kapitel 26: XML und JSON**

#### 26.1 Was ist XML?

#### 26.2 XML-Dokumente

#### 26.3 Ein XML-Dokument als Baum

26.4 DOM

26.5 Das Modul xml.dom.minidom

26.5.1 XML-Dokumente und DOM-Objekte

26.5.2 Die Basisklasse Node

26.5.3 Die Klassen Document, Element und Text

26.6 Attribute von XML-Elementen

26.7 Anwendungsbeispiel 1: Eine XML-basierte Klasse

26.8 Anwendungsbeispiel 2: Datenkommunikation mit XML

26.8.1 Überblick

26.8.2 Das Client-Programm

26.8.3 Das Server-Programm

26.9 JSON

26.9.1 JSON-Texte decodieren

26.9.2 Decodierungsfehler

26.9.3 Ein Dictionary als JSON-Objekt speichern:  
Kompakt oder gut lesbar?

26.9.4 Projekt: Verarbeitung von Wetterdaten

26.10 Aufgaben

26.11 Lösungen

## **Kapitel 27: Modellieren mit Kellern, Schlangen und Graphen**

27.1 Stack (Keller, Stapel)

27.2 Queue (Schlange)

27.3 Graphen

27.4 Aufgaben

27.5 Lösungen

## **Kapitel 28: Benutzungsoberflächen mit Qt**

28.1 Was bietet PyQt5?

28.2 PyQt5 erkunden

28.3 Wie arbeitet PyQt? Applikation und Fenster

28.4 Eine objektorientierte Anwendung mit PyQt5

28.5 Ein Webbrowser

28.6 Interaktive Widgets

28.7 Label – Ausgabe von Text und Bild

28.8 Signale

28.9 Checkboxes und Radiobuttons

28.10 Auswahlliste (ComboBox)

28.11 Gemeinsame Operationen der Widgets

28.12 Spezielle Methoden eines Fensters

28.13 Events

28.14 Fonts

28.15 Stylesheets

28.16 Icons

28.17 Messageboxen

28.18 Timer

28.19 Das Qt-Layout unter der Lupe

28.19.1 Absolute Positionierung und Größe

28.19.2 Raster-Layout

28.19.3 Form-Layout

28.20 Browser für jeden Zweck

28.20.1 Die Klasse QWebEngineView

28.21 Ein Webbrowser mit Filter

28.22 Surfen mit Geschichte – der Verlauf einer Sitzung

28.23 Aufgaben

28.24 Lösungen

## **Kapitel 29: Multimediaanwendungen mit Qt**

29.1 Kalender und Textfeld – ein digitales Tagebuch

29.1.1 Programmierung

29.2 Kamerabilder

29.3 Dialoge

29.3.1 Projekt: Ansichtskarte

29.4 Videoplayer

29.4.1 Ein einfacher Videoplayer

29.4.2 Videoplayer mit Playlist

29.4.3 Regeln zur Änderung der Größe (Size Policy)

29.4.4 Das Dashboard bei Mausbewegungen einblenden

29.5 Aufgaben

29.6 Lösungen

## **Kapitel 30: Rechnen mit NumPy**

30.1 NumPy installieren

30.2 Arrays erzeugen

30.2.1 Arrays

30.2.2 Matrizen und Vektoren

30.2.3 Zahlenfolgen

30.2.4 Zufallsarrays

30.2.5 Spezielle Arrays

30.3 Indizieren

- 30.4 Slicing
- 30.5 Arrays verändern
- 30.6 Arithmetische Operationen
- 30.7 Funktionen, die elementweise ausgeführt werden
- 30.8 Einfache Visualisierung
- 30.9 Matrizenmultiplikation mit dot()
- 30.10 Array-Funktionen und Achsen
- 30.11 Projekt: Diffusion
- 30.12 Vergleiche
- 30.13 Projekt: Wolken am Himmel
- 30.14 Projekt: Wie versteckt man ein Buch in einem Bild?
- 30.15 Datenanalyse mit Histogrammen
- 30.16 Wie funktioniert ein Medianfilter?
- 30.17 Rechnen mit SciPy
  - 30.17.1 Lineare Gleichungssysteme lösen
  - 30.17.2 Integration
- 30.18 Aufgaben
- 30.19 Lösungen

## **Kapitel 31: Messdaten verarbeiten**

- 31.1 Messwerte in einem Diagramm darstellen - Matplotlib und tkinter
  - 31.1.1 Basisprojekt
  - 31.1.2 Erweiterung: Den letzten Wert löschen
  - 31.1.3 Das Aussehen eines Diagramms gestalten

31.2 Messwerte aus einem Multimeter lesen und darstellen

31.2.1 Vorbereitung

31.2.2 Werte auslesen

31.2.3 Welche Ziffern zeigt das Display des Multimeters?

31.3 Anzeige der Temperatur

31.4 Messreihen aufzeichnen

31.5 Aufgabe

31.6 Lösung

## **Kapitel 32: Parallele Datenverarbeitung**

32.1 Was sind parallele Programme?

32.2 Prozesse starten und abbrechen

32.3 Funktionen in eigenen Prozessen starten

32.4 Prozesse zusammenführen - join()

32.5 Wie können Prozesse Objekte austauschen?

32.5.1 Objekte als Argumente übergeben

32.5.2 Objekte über eine Pipe senden und empfangen

32.5.3 Objekte über eine Queue austauschen

32.6 Daten im Pool bearbeiten

32.6.1 Mit dem Pool geht's schneller - ein Zeitexperiment

32.6.2 Forschen mit Big Data aus dem Internet

32.7 Synchronisation

32.8 Produzenten und Konsumenten

32.8.1 Sprücheklopfer