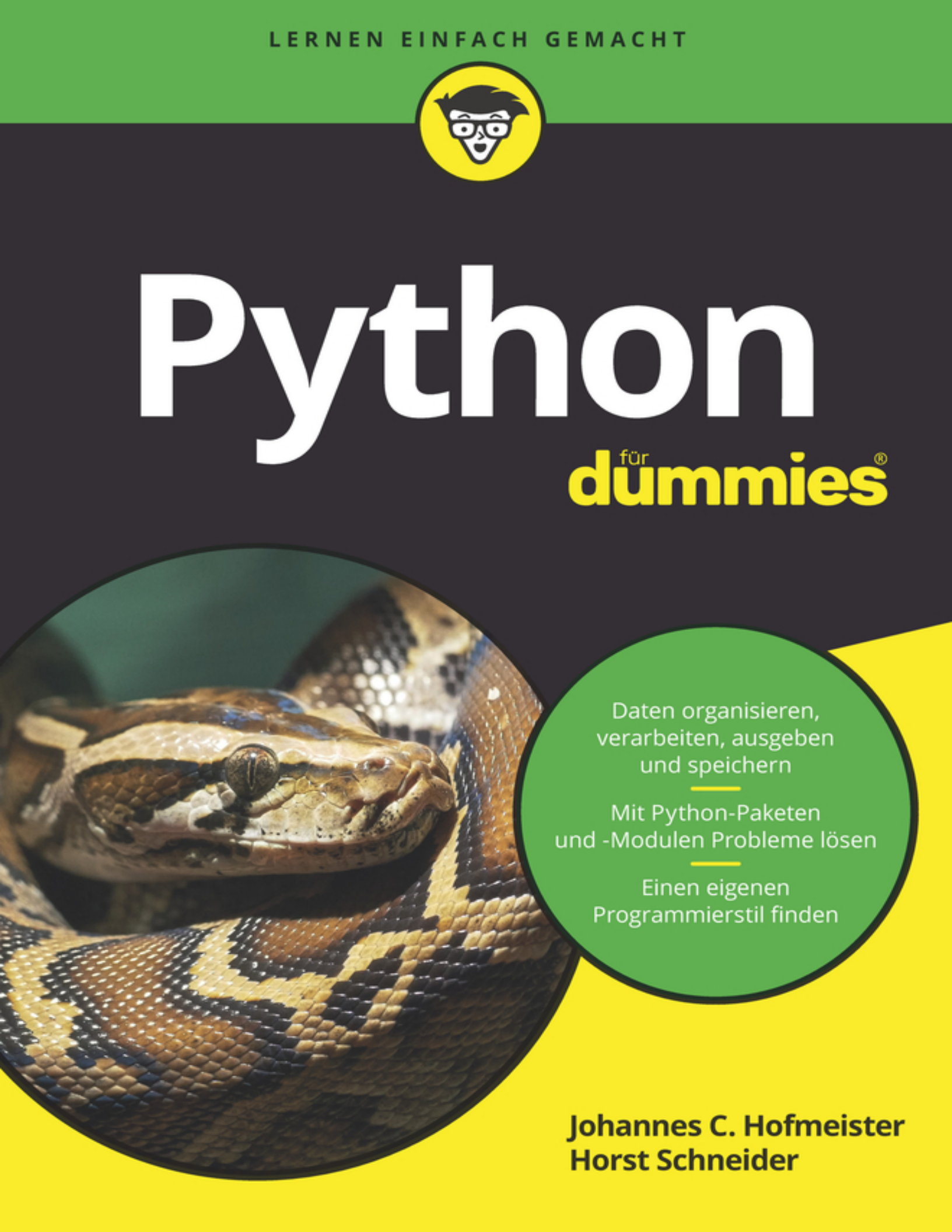
LERNEN EINFACH GEMACHT



# Python

## für dummies®



Daten organisieren, verarbeiten, ausgeben und speichern

Mit Python-Paketen und -Modulen Probleme lösen

Einen eigenen Programmierstil finden

**Johannes C. Hofmeister**
**Horst Schneider**

# Python für Dummies

# Schummelseite

## THE ZEN OF PYTHON VON TIM PETERS

Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one – and preferably only one – obvious way to do it.
Although that way may not be obvious at first unless you're Dutch.
Now is better than never.
Although never is often better than *right* now.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
Namespaces are one honking great idea – let's do more of those!

## COMPREHENSIONS

| | |
|---|---|
| Liste | `[i for i in iterable]` |
| Set | `{i for i in iterable}` |

| | |
|---|---|
| Dictionary | `{key:value for key,value in iterable}` |
| Generator Expression | `(i for i in iterable)` |

## CONTAINER

| | |
|---|---|
| Liste | `[1, 2, 3]` |
| Tupel | `(1, 2, 3)` |
| Set | `{1, 2, 3}` |
| Dictionary | `{'a': 1, 'b': 2 }` |

## STRINGS

| | |
|---|---|
| String | `“Hello”` |
| | `'Hello'` |
| Mehrzeilig | `“““Hello”””` |
| | `’’'Hello'’’` |
| F-String | `f'Hello, {variable}'` |
| Bytes | `b'Hello'` |
| Raw | `r'Hello'` |

## RECHNEN

| | |
|---|---|
| Grundrechenarten | `6 + 3` |
| | `12 - 3` |
| | `3 * 3` |
| | `18 / 2` |

| | |
|---|---|
| Division ohne Rest | `19 // 2` |
| Modulo | `19 % 10` |
| Potenzieren | `3 ** 2` |

## LOGIK

| | |
|---|---|
| Boolean | `True, False` |
| Und | `True and True` |
| Oder | `True or False` |
| Negation | `not False` |

## ZAHLENLITERALE

| | |
|---|---|
| Floats | `-39.23` |
| Wissenschaftlich | `1.90e-3` |
| Integer | `23` |
| Mit Trennzeichen | `10_000_000` |
| Binär | `0b1100110011` |
| Oktal | `0o123` |
| Hexadezimal | `0xff00ff` |

## VERGLEICHEN

| | |
|---|---|
| Identität | `a is a` |
| Inklusion | `'a' in 'abc'` |
| Vergleiche | `3 == 3` |
| | `1 != 2` |

```
1 < 2

2 > 1

2 <= 2

2 >= 2
```

## FORMATIERUNG: STRINGS

| | | |
|---|---|---|
| Linksbündig | `f'{pi:<10.4f}'` | `'3.1416 '` |
| Zentriert | `f'{pi:^10.4f}'` | `' 3.1416 '` |
| Rechtsbündig | `f'{pi:>10.4f}'` | `' 3.1416'` |
| Zeichen auffüllen | `f'{.5: ^10.2%}'` | `'~ 50.00%  '` |
| Runden | `f'{12.5865:.3f}'` | `'12.586'` |
| Wissenschaftlich | `f'{pi:E}'` | `'3.141593E+00'` |
| Prozent | `f'{.5:.2%}'` | `'50.00%'` |
| Maskieren | `f'{iban:*<22.8}'` | `'DE893704**********'` |

## DATUMSFORMATE

| | |
|---|---|
| `%d` | Tag |
| `%m` | Monat |
| `%Y / %y` | Jahr vier-/zweistellig |
| `%A / %a` | Wochentag (lang/kurz) |
| `%B / %b` | Monatsname (lang/kurz) |
| `%W` | Kalenderwoche |
| `%x` | Komplettes Datum |

## ZEITFORMATE

| | |
|---|---|
| `%H / %I` | Stunde (24/12) |
| `%M` | Minute |
| `%S` | Sekunde |
| `%f` | Mikrosekunde |
| `%z` | Zeitverschiebung (ab UTC) |
| `%Z` | Name der Zeitzone |
| `%X` | Komplette Zeit |

## Datum und Zeit Beispiele

| | | |
|---|---|---|
| Uhrzeit | `now.strftime('%H:%M:%S.%f')` | `'17:23:20.011664'` |
| Datum | `now.strftime('%A, der %d. %B %Y')` | `'Montag, der 30. März 2020'` |

Falls nötig, Locale setzen:

```
import datetime, locale

locale.setlocale(locale.LC_ALL, 'de_de')

now = datetime.datetime.now()
```

## ZAHLENDARSTELLUNG

| | |
|---|---|
| `oct(int)` | Oktal |
| `bin(int)` | Binär |
| `hex(int)` | Hexadezimal |
| `chr(int)` | Unicode-Zeichen |

## MATHE

| | |
|---|---|
| `min(iterable)` | Kleinster Wert |
| `max(iterable)` | Größter Wert |
| `sum(iterable)` | Summe |
| `round(number, [n])` | Runden |
| `abs(number)` | Betrag |
| `pow(number, exp)` | Potenz |
| `divmod(a, b)` | Teilen mit Rest |

## ELEMENTARE TYPEN

| | |
|---|---|
| `str(object)` | String |
| `bytes(object)` | Bytes-Objekt |
| `int(str)` | Ganzzahl |
| `bool(object)` | Wahrheitswert |
| `float(str)` | Gleitkommazahl |
| `list(iterable)` | Liste |
| `tuple(iterable)` | Tupel |
| `set(iterable)` | Menge (Set) |
| `dict(iterable)` | Dictionary |

## REPL

| | |
|---|---|
| `help([object])` | Hilfe anzeigen |
| `dir([object])` | Attribute auflisten |

## INTERVALLE ' ITERATION

| | |
|---|---|
| `enumerate(iterable, start=0)` | Werte aufzählen |
| `sorted(iterable)` | Werte sortieren |
| `reversed(sequence)` | Sequenz umkehren |
| `iter(object)` | Iterator erzeugen |
| `next(iterator)` | Nächstes Element aus Iterator nehmen |
| `range(start, stop, step)` | Zahlen im Intervall ausgeben |
| `zip(*iterables)` | Werte gruppiert iterieren |

## AGGREGATION

| | |
|---|---|
| `len(sequence)` | Länge einer Sequenz |
| `any(iterable)` | Ist einer der Werte truthy? |
| `all(iterable)` | Sind alle Wert truthy? |
| `map(function, iterable)` | Funktion auf alle Werte anwenden |
| `filter(function, iterable)` | Nur Werte nehmen, die truthy ergeben |

## INPUT ' OUTPUT

| | |
|---|---|
| `input([prompt])` | Tastatureingabe |
| `open(path, mode='r', encoding=None)` | Dateien öffnen |
| `print(*objects, sep=' ', end='\n', file='sys.stdout')` | Ausgabe |

## DEKORATOREN

```
def log(print_function, prefix=__file__):
    def decorate(function):
        # Funktion mit allgemeingültiger Signatur
        def call(*args, **kwargs):
            name = function.__name__
            message = f'{prefix}: {name}({args}, {kwargs})'
            print_function(message)
            # Eigentliche Funktion aufrufen
            return function(*args, **kwargs)
        return call
    return decorate

@log(print, 'DEBUG')
def add(a, b):
    return a + b

add(1, 2)
```

## SCHWEIZER TASCHENMESSER

| | |
|---|---|
| Unit-Tests ausführen | `$ python3 -m unittest` |
| Webserver starten | `$ python3 -m http.server 8001` |
| Mailserver starten | `$ python3 -m smtpd -n -d` |
| Zip-Datei erstellen | `$ python3 -m zipfile -c backup.zip documents/` |
| Zip-Datei extrahieren | `$ python3 -m zipfile -e backup.zip` |
| Kalender ausgeben | `$ python3 -m calendar 2019 12` |
| JSON formatieren | `$ python3 -m json.tool file.json` |

| | |
|---|---|
| UUID generieren | `$ python3 -c “import uuid;print(uuid.uuid4().hex)”` |

Johannes C. Hofmeister
Horst Schneider

# Python für dummies®

Fachkorrektur von Prof. Thomas Smits

WILEY
WILEY-VCH Verlag GmbH & Co. KGaA

**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.



**Print ISBN:** 978-3-527-71414-8

**ePub ISBN:** 978-3-527-81116-8

Coverfoto: © chris – stock.adobe.com

Korrektur: Matthias Delbrück, Dossenheim/Bergstraße

# Inhaltsverzeichnis

# Tabellenverzeichnis

### Kapitel 7



### Kapitel 8



### Kapitel 12



### Kapitel 17



### Kapitel 19



# Illustrationsverzeichnis

### Kapitel 2

Abbildung 17.1 Die Verteilung der Punkte nach Geschlecht aufgeschlüsselt

# Über den Autoren

## Über Johannes

Johannes C. Hofmeister lebt in Heidelberg. Am Psychologischen Institut der Universität Heidelberg arbeitet er als Administrator und erforscht nebenher, was Code verständlich macht. Als Softwareentwickler und Berater hat er in großen und kleinen Unternehmen C#, Java und JavaScript eingesetzt, aber am liebsten liest und schreibt er Python-Code.

## Über Horst

Horst Schneider lebt in Mannheim und arbeitet seit mehr als zehn Jahren als Softwareentwickler. Sein Schwerpunkt ist die Anwendungsentwicklung in verschiedenen Sprachen, von Java über C# bis zu JavaScript und Python. Aktuell arbeitet er als Coach und Berater im Raum Heidelberg und löst vielfältige Probleme – vorzugsweise mit Python.

## Johannes' Widmung

Für Katha.

## Horsts Widmung

Ich möchte dieses Buch Sabine und Rosa widmen. Danke für eure Unterstützung und die viele Geduld, die ihr mit mir hattet.

## Danksagung

Eine frühe Fassung dieses Buches enthielt eine fiktive Anekdote darüber, wie der Mathematiker Alonzo Church Familie, Freunde und körperliche Hygiene vernachlässigte, um an seinen Theorien zum Lambda-Kalkül zu arbeiten. Inzwischen ist uns klar, dass wir wohl unsere eigene Situation auf Church projiziert haben. Wir bedanken uns daher bei allen, die uns in der Zeit unterstützt oder zumindest wohlwollend ertragen und uns den Rücken freigehalten haben.

Ganz besonderer Dank gilt Lars Kumbier für Raum, Zeit und aufmerksame Korrekturen. Kim Almasan wollen wir für die Installationsschritte auf dem Mac danken, die wir uns mangels Equipments sonst hätten ausdenken müssen. Dank gilt unserem enthusiastischen Erstleser Volker Lueg, dessen Anmerkungen unseren Code verbessert und uns sehr motiviert haben. Vielen Dank an Marion Lammarsch fürs Quietscheentchen-Debugging.

Ohne unseren Fachkorrektor Thomas Smits wäre das Buch wesentlich schlechter und wir ein wenig dümmer. Danke.

# Einleitung

Python ist die beste Programmiersprache. Das behaupten wir nicht einfach so, sondern wir (sowohl Horst als auch Johannes) mussten erst viele gute und schlechte Erfahrungen mit anderen Sprachen sammeln, um zu so einer pauschalen Aussage zu kommen. In der Schule gab es Turbopascal und PHP. In der Uni gab's obendrein C, C++ und Java, im Job kam C# hinzu. Später nahmen wir JavaScript ins Portfolio auf.

Wie viel Spaß Programmieren machen kann, wissen wir aber erst, seit wir Python benutzen. Python ist unbürokratisch, dynamisch, schnell, intelligent und elegant. Python hat etwas Magisches an sich. Mit diesem Buch wollen wir Ihnen etwas von dieser Magie abgeben. Vielleicht können wir Sie ein bisschen verzaubern!

Wenn Sie das Buch gut finden, sagen Sie es anderen. Wenn Sie es blöd finden, sagen Sie es uns – dann verbessern wir es!

Viel Spaß beim Lesen!

## *Törichte Annahmen über den Leser*

Dieses Buch richtet sich an Programmierer, die schon Erfahrungen in anderen Sprachen wie C, Java oder JavaScript sammeln konnten, sowie an ambitionierte Neulinge.

Wir werfen immer mal wieder Infos ein, die auf grundlegendem, technischem Hintergrundwissen aufbauen – das könnte blutige Programmieranfänger

überfordern. Aber vielleicht trauen Sie sich trotzdem. Wir versuchen natürlich, alles zu erklären.

Die folgenden Dinge sollten Sie mitbringen:

- ✔ Einen Computer mit Windows, Linux oder macOS
- ✔ Deutsch- und Englischkenntnisse
- ✔ Idealerweise Erfahrung im Umgang mit der Kommandozeile

# Wie Sie dieses Buch nutzen können

Dies ist ein Buch zum Mitmachen. Sie finden viele Code-Schnipsel, die Sie gerne abtippen können – dabei lernt man sehr viel, vor allem am Anfang. Wenn Sie keine Lust haben, alles von Hand abzutippen, finden Sie alle Code-Beispiele zum Herunterladen auf der Webseite des Buches unter www.wiley-vch.de/ISBN9783527714148.

Sie dürfen auch gerne ins Buch reinkritzeln und Eselsohren hineinmachen, denn so ein Buch ist ein Gebrauchsgegenstand.

# Was Sie nicht lesen müssen

Lesen Sie nicht einfach von vorne nach hinten, sondern der Nase nach. Lesen Sie, was Sie gerade interessiert und schmökern Sie hin und her. Außerdem sollten Sie etwas Humor mitbringen. Wir hatten viel Spaß beim Schreiben und hoffen, dass Sie ebenso viel Spaß beim Lesen haben.

# Wie dieses Buch aufgebaut ist

Das Buch ist in folgende Teile unterteilt:

## Teil I: Langweilige Einmallektüre

Irgendwo muss man anfangen. In diesem Teil helfen wir Ihnen bei der Installation und erklären noch ein bisschen, wie Sie die Code-Beispiele im Buch zum Laufen bringen. Diese Vorbereitungen sind notwendig, damit Sie den Beispielen im Buch folgen können.

## Teil II: Python sprechen lernen

Weil wir davon ausgehen, dass Sie schon ein bisschen programmieren können, zeigen wir direkt einen wilden Ritt durch die wichtigsten Sprachfeatures, sodass Sie diese mit Ihren eigenen Erfahrungen abgleichen können. Danach lernen Sie noch einige grundlegende Details sowie coole Syntax-Features, die Sie so sicher noch nicht kennen. Am Ende dieses Teils können Sie Programme erstellen, die Dateien lesen und schreiben.

## Teil III: Mit Python Probleme lösen

Eine Programmiersprache allein macht noch kein Programm – man braucht noch eine Problemstellung. Viele Probleme sind in Python bereits gelöst, denn Python bringt Myriaden an kleinen Funktionen, praktischen Modulen, interessanten Paketen und sogar eigenständigen Programmen mit, die Sie direkt nach der Installation einsetzen können. Bevor Sie also anfangen, kleinschrittig viel Code zu schreiben, sollten Sie sich ansehen, was es in Python schon so alles gibt. So ersparen Sie sich viel Kleinarbeit wie die Neuerfindung des Rades und des plattgefahrenen Reifens.

### Teil IV: Python als Handwerk

Die kleinen Probleme sind in Python schon gelöst, daher können Sie sich ganz auf Ihre eigene Problemstellung konzentrieren. Um einem Computer beizubringen, wie man sich die Welt denn nun vorstellt, muss man seine Worte aber vorsichtig wählen. Wie Sie sich gut ausdrücken, zeigen wir in diesem Buchteil. Hier geht es um Funktionen, Objekte und Ausnahmen, die Ihnen sicher aus anderen Sprachen geläufig sind. Danach haben Sie das Wichtigste gesehen und sollten keine Überraschungen mehr erleben, wenn Sie anderer Leute Python-Code lesen.

### Teil V: Brötchen (oder Lorbeeren) mit Python verdienen

Programmieren bereitet Freude, weil es so effektvoll ist. Man tippt wirre Zeichen und der Computer rattert und gibt irgendetwas aus – das macht einen Höllenspaß. Leider genießen nur wenige den Luxus, aus Spaß an der Freude vor sich hin zu programmieren. Daher zeigt dieser Buchteil einige Problemstellungen, die Sie so oder so ähnlich auch in der Praxis erleben könnten.

### Teil VI: Der Top-Ten-Teil

Im Top-Ten-Teil wird eine Auswahl interessanter Bibliotheken präsentiert. Zusätzlich finden Sie Themen, die im Buch nicht besprochen werden konnten, aber Ihnen einen Einstieg bieten, falls Sie sich in Zukunft noch tiefer mit Python befassen wollen.

## Symbole, die in diesem Buch verwendet werden

Im Buch verteilt finden Sie verschiedene Symbole:

 Mit der Glühbirne werden praktische Tipps für den Alltag hervorgehoben.

 Der Techniker liefert Hintergrund-Infos, Fakten oder interessante Details und taucht immer dann auf, wenn Erbsen gezählt werden müssen.

 Der Finger mit dem Faden zeigt an, dass Sie sich etwas gut merken sollten oder dass etwas Wichtiges wiederholt wird.

 Dieses Symbol zeigt, dass hier etwas schiefgehen könnte. Meistens heißt das, dass wir Autoren diesen Fehler schon mindestens 300-mal gemacht haben. Daher sollten Sie sich aus Solidarität die Zeit nehmen, ihn auch zu machen.

 Hier plaudern wir aus dem persönlichen Nähkästchen und erzählen einen Schwank aus dem Entwickleralltag.

# Konventionen in diesem Buch

Wir benutzen in diesem Buch die folgenden typografischen Konventionen.

Python-Code ist so formatiert:

```
# Kommentar
print("Python für Dummies.")
```

Wir wollen uns nicht auf ein Betriebssystem festlegen, daher machen wir viel mit der Kommandozeile. Egal ob Linux, macOS und Windows – eine Kommandozeile gibt es immer frei Haus. Wir ermutigen Sie, sich mit ihr anzufreunden, denn sie ermöglicht einen direkten Draht zu Ihrem Betriebssystem.

Immer wenn einem Befehl ein Dollar (`$`) voransteht, meinen wir ganz allgemein, dass Sie diesen Befehl auf einer Kommandozeile eingeben sollen.

```
$ command
```

Wir unterscheiden dabei nicht zwischen den einzelnen Betriebssystemen und gehen davon aus, dass alle Befehle auf den gängigen Kommandozeilen funktionieren – auch wenn Eingabeaufforderung und Bildschirmausgabe subtil abweichen können.

Neue *Konzepte* heben wir hervor.

Wenn wir Code erklären, dann formatieren wir diesen auch im Fließtext so, dass Sie die Variablen wiedererkennen. Weiter oben im Code war zum Beispiel von `print` die Rede.

- ✔ Variablen, die Sie im Fließtext finden, werden durch eine nichtproportionale Schriftart hervorgehoben, etwa so: `variable`
- ✔ Diese Schriftart verwenden wir auch für Internetadressen (URLs): `https://www.python.org/doc/humor/`
- ✔ Anklickbare Menüpunkte und Beschriftungen werden in KAPITäLCHEN gesetzt
- ✔ Namen von Tools sehen so aus: *Werkzeug*
- ✔ Auch Platzhalter, die Sie geistig mit eigenen Werten füllen müssen, werden kursiv hervorgehoben:

*<passwort>*

# Teil I

# Langweilige Einmallektüre