

---

# Inhalt

<b>Der Autor</b> .....	<b>xvi</b>
<b>Der Fachgutachter</b> .....	<b>xvi</b>
<b>Danksagung</b> .....	<b>xvii</b>
<b>Einleitung</b> .....	<b>1</b>
Wer dieses Buch lesen sollte und warum .....	2
Über dieses Buch .....	3
Ihr Weg zur Programmierung .....	5
<b>1 Fehlermeldungen und Recherche</b> .....	<b>7</b>
Python-Fehlermeldungen verstehen .....	8
Tracebacks untersuchen .....	8
Fehlermeldungen recherchieren .....	11
Fehler vermeiden mit Lintern .....	13
Um Hilfe bitten .....	14
Geben Sie gleich ausreichend Informationen, um Rückfragen zu vermeiden .....	15
Formulieren Sie Ihre Fragen als Fragen .....	15
Stellen Sie Ihre Fragen auf einer geeigneten Website .....	16
Geben Sie das Problem in der Überschrift an .....	16
Erklären Sie, was der Code tun soll .....	16
Geben Sie die vollständige Fehlermeldung an .....	17
Teilen Sie Ihren Code vollständig mit .....	17
Gestalten Sie Ihren Code durch saubere Formatierung lesbar .....	18
Beschreiben Sie, was Sie bereits versucht haben .....	19
Beschreiben Sie Ihre Ausstattung .....	19
Ein Beispiel für eine gut gestellte Frage .....	20
Zusammenfassung .....	21

<b>2 Die Einrichtung der Umgebung und die Befehlszeile</b> .....	<b>23</b>
Das Dateisystem .....	24
Pfade in Python .....	25
Das Benutzerverzeichnis .....	25
Das aktuelle Arbeitsverzeichnis .....	26
Absolute und relative Pfade .....	27
Programme und Prozesse .....	28
Die Befehlszeile .....	29
Ein Terminalfenster öffnen .....	30
Programme an der Befehlszeile ausführen .....	31
Befehlszeilenargumente .....	32
Python-Code mit <code>-c</code> an der Befehlszeile ausführen .....	33
Python-Programme an der Befehlszeile ausführen .....	34
Py.exe .....	34
Befehle aus einem Python-Programm heraus ausführen .....	35
Tipparbeit durch Tabulatorvervollständigung sparen .....	35
Der Befehlsverlauf .....	36
Gebräuchliche Befehle .....	36
PATH und andere Umgebungsvariablen .....	44
Umgebungsvariablen anzeigen .....	44
Die Umgebungsvariable PATH .....	45
Die Umgebungsvariable PATH in der Befehlszeile ändern .....	46
Ordner in Windows dauerhaft zu PATH hinzufügen .....	47
Ordner in macOS und Linux dauerhaft zu PATH hinzufügen .....	49
Python-Programme außerhalb der Befehlszeile ausführen .....	49
Python-Programme in Windows ausführen .....	49
Python-Programme in macOS ausführen .....	51
Python-Programme in Ubuntu Linux ausführen .....	51
Zusammenfassung .....	52
<b>3 Codeformatierung mit Black</b> .....	<b>53</b>
Wie man Freunde und Kollegen vergrault .....	54
PEP 8 und andere Stilrichtlinien .....	54
Horizontale Abstände .....	55
Leerzeichen zur Einrückung verwenden .....	55
Abstände innerhalb einer Zeile .....	57
Vertikale Abstände .....	60
Beispiel für vertikale Abstände .....	60
Empfohlene Vorgehensweisen für vertikale Abstände .....	62

---

Black: Der kompromisslose Codeformatierer	63
Black installieren	63
Black an der Befehlszeile ausführen	64
Black für einzelne Abschnitte Ihres Codes ausschalten	67
Zusammenfassung	68
<b>4 Aussagekräftige Namen</b>	<b>69</b>
Groß- und Kleinschreibung	70
Namenskonventionen in PEP 8	71
Namen geeigneter Länge	72
Zu kurze Namen	72
Zu lange Namen	73
Leicht zu findende Namen	75
Scherze, Wortspiele und Anspielungen vermeiden	75
Integrierte Namen nicht überschreiben	76
Die allerschlechtesten Variablennamen	78
Zusammenfassung	78
<b>5 Codegerüche</b>	<b>79</b>
Duplizierter Code	80
Magische Zahlen	82
Auskommentierter und toter Code	84
Print-Debugging	86
Variablen mit numerischen Suffixen	87
Klassen statt Funktionen oder Module	88
Verschachtelte Listennotation	89
Leere except-Blöcke und nichtssagende Fehlermeldungen	91
Legenden über Code Smells	92
Legende: Funktionen sollten nur eine return-Anweisung am Ende aufweisen	93
Legende: Funktionen mit einer try-Anweisung dürfen keine anderen Anweisungen enthalten	93
Legende: Flag-Argumente sind schlecht	94
Legende: Globale Variablen sind schlecht	95
Legende: Kommentare sind unnötig	96
Zusammenfassung	97

<b>6 Pythonischer Code</b> .....	<b>99</b>
Python-Zen .....	100
Aussagekräftige Einrückungen .....	104
Leistung mit dem Modul <code>timeit</code> messen .....	105
Häufig falsch angewendete Syntax .....	108
Verwenden Sie <code>enumerate()</code> statt <code>range()</code> .....	108
Verwenden Sie <code>with</code> statt <code>open()</code> und <code>close()</code> .....	109
Verwenden Sie <code>is</code> statt <code>==</code> zum Vergleich mit <code>None</code> .....	110
Stringformatierung .....	111
Verwenden Sie Rohstrings bei einer großen Anzahl von Backslashes ...	111
F-Strings .....	112
Flache Kopien von Listen .....	113
Pythonischer Umgang mit Dictionarys .....	114
Verwenden Sie <code>get()</code> und <code>setdefault()</code> für Dictionarys .....	114
Verwenden Sie <code>collections.defaultdict</code> für Standardwerte .....	116
Verwenden Sie Dictionarys statt <code>switch</code> -Konstruktionen .....	117
Bedingte Ausdrücke: Pythons »hässlicher« ternärer Operator .....	118
Variablenwerte .....	120
Zuweisungs- und Vergleichsoperatoren verketteten .....	120
Eine Variable auf Gleichheit mit mehreren Werten prüfen .....	121
Zusammenfassung .....	121
<b>7 Programmierjargon</b> .....	<b>123</b>
Definitionen .....	124
Die Sprache Python und der Interpreter Python .....	124
Garbage Collection .....	125
Literele .....	126
Schlüsselwörter .....	126
Objekte, Werte und Identitäten .....	127
Elemente .....	131
Veränderbare und unveränderbare Objekte .....	131
Indizes, Schlüssel und Hashes .....	134
Container-, Folgen-, Maps- und Set-Datentypen .....	137
Dunder- oder magische Methoden .....	138
Module und Pakete .....	139
Aufrufbare Objekte und Objekte erster Klasse .....	139

---

Häufig verwechselte Begriffe .....	141
Anweisungen und Ausdrücke .....	141
Blöcke, Klauseln und Rümpfe .....	142
Variablen und Attribute .....	143
Funktionen und Methoden .....	143
Iterierbare Objekte und Iteratoren .....	144
Syntax-, Laufzeit- und semantische Fehler .....	146
Parameter und Argumente .....	148
Implizite und explizite Typumwandlung .....	148
Eigenschaften und Attribute .....	149
Bytecode und Maschinencode .....	149
Skripte und Programme, Skriptsprachen und Programmiersprachen ...	150
Bibliotheken, Frameworks, SDKs, Engines und APIs .....	150
Zusammenfassung .....	151
Literatur .....	152
<b>8 Häufige Fallstricke in Python .....</b>	<b>153</b>
Elemente zu Listen hinzufügen und entfernen .....	154
Veränderbare Werte kopieren .....	161
Standardargumente .....	164
Strings zusammenbauen .....	166
Sortierung mit sort() .....	168
Genauigkeit von Fließkommazahlen .....	169
Verkettung von Ungleichheitsoperatoren .....	172
Das Komma in einelementigen Tupeln .....	172
Zusammenfassung .....	173
<b>9 Exotische Eigenarten von Python .....</b>	<b>175</b>
256 ist 256, aber 257 ist nicht 257 .....	175
String-Interning .....	177
Die Bedeutung von ++ und -- in Python .....	178
Alles von nichts .....	179
Boolesche Werte als Integer .....	180
Verkettung unterschiedlicher Operatoren .....	182
Schwereelosigkeit in Python .....	182
Zusammenfassung .....	183

<b>10 Zweckmäßige Funktionen</b> .....	<b>185</b>
Funktionsnamen .....	186
Der Umfang von Funktionen .....	186
Funktionsparameter und -argumente .....	189
Standardargumente .....	190
Argumente mit * und ** an Funktionen übergeben .....	190
Variadische Funktionen mit * erstellen .....	192
Variadische Funktionen mit ** erstellen .....	195
Wrapper-Funktionen mit * und ** erstellen .....	197
Funktionale Programmierung .....	198
Nebenwirkungen .....	198
Funktionen höherer Ordnung .....	200
Lambda-Funktionen .....	201
Zuordnung und Filterung mit Listennotation .....	202
Der Datentyp von Rückgabewert .....	203
Ausnahmen auslösen oder Fehlercodes zurückgeben .....	206
Zusammenfassung .....	207
<b>11 Kommentare, Docstrings und Typhinweise</b> .....	<b>209</b>
Kommentare .....	210
Formatierung von Kommentaren .....	211
Inline-Kommentare .....	212
Erklärende Kommentare .....	213
Kommentare zur Gliederung .....	213
»Lessons-Learned-Kommentare« .....	214
Rechtliche Hinweise .....	215
Professionelle Formulierung .....	215
Codetags und TODO-Kommentare .....	216
Magische Kommentare und Quelldateicodierung .....	217
Docstrings .....	217
Typhinweise .....	220
Tools zur statischen Analyse .....	222
Typhinweise für mehrere Typen .....	225
Typhinweise für Listen, Dictionarys u. Ä. ....	226
Rückportierung von Typhinweisen mithilfe von Kommentaren .....	227
Zusammenfassung .....	228

<b>12 Versionssteuerung mit Git</b> .....	<b>231</b>
Commits und Repositories in Git .....	232
Neue Python-Projekte mit Cookiecutter erstellen .....	232
Git installieren .....	235
Git-Benutzername und E-Mail-Adresse angeben .....	236
GUI-Werkzeuge für Git installieren .....	236
Der Arbeitsablauf in Git .....	237
Der Dateistatus in Git .....	237
Wozu gibt es bereitgestellte Dateien? .....	239
Ein Git-Repository erstellen .....	240
Zu verfolgende Dateien hinzufügen .....	241
Einzelne Dateien ignorieren .....	243
Änderungen mit Commit bestätigen .....	244
Änderungen mit git diff vor dem Commit einsehen .....	245
Änderungen mit git difftool in einer GUI-Anwendung einsehen .....	247
Häufigkeit von Commits .....	248
Dateien löschen .....	249
Dateien umbenennen und verschieben .....	250
Das Commitprotokoll einsehen .....	252
Frühere Versionen wiederherstellen .....	253
Unbestätigte lokale Änderungen rückgängig machen .....	253
Bereitstellung einer Datei aufheben .....	254
Die letzten Commits zurücknehmen .....	254
Zurücksetzen einer einzelnen Datei zu einem bestimmten Commit .....	255
Den Commitverlauf ändern .....	256
GitHub und git push .....	257
Ein bestehendes Repository auf GitHub übertragen .....	258
Ein GitHub-Repository klonen .....	259
Zusammenfassung .....	260
<b>13 Leistungsmessung und Algorithmusanalyse</b> .....	<b>261</b>
Das Modul timeit .....	262
Der Profiler cProfile .....	265
Komplexitätsanalyse .....	267
Ordnungen .....	268
Ein Bücherregal als Metapher für Ordnungen .....	269
Worst Case und Best Case .....	273

Die Ordnung Ihres Codes bestimmen	275
Warum Terme niedriger Ordnungen und Koeffizienten keine Rolle spielen	276
Beispiele für die Komplexitätsanalyse	278
Die Ordnung gängiger Funktionsaufrufe	281
Komplexitätsanalyse im Überblick	282
Die Ordnung spielt bei kleinem $n$ keine Rolle – und $n$ ist gewöhnlich klein	284
Zusammenfassung	284
<b>14 Praxisprojekte</b>	<b>287</b>
Turm von Hanoi	288
Die Ausgabe	289
Der Quellcode	290
Den Code schreiben	292
Vier gewinnt	301
Die Ausgabe	301
Der Quellcode	302
Den Code schreiben	306
Zusammenfassung	315
<b>15 Klassen</b>	<b>317</b>
Formulare als Veranschaulichung	318
Objekte aus Klassen erstellen	320
Eine einfache Klasse erstellen: WizCoin	321
Methoden, <code>__init__()</code> und der Parameter <code>self</code>	324
Attribute	325
Private Attribute und private Methoden	326
Die Funktion <code>type()</code> und das Attribut <code>__qualname__</code>	328
OOP- und Nicht-OOP-Code im Vergleich	329
Klassen für reale Objekte	335
Zusammenfassung	336



<b>16 Vererbung</b> .....	<b>339</b>
Wie Vererbung funktioniert .....	340
Methoden überschreiben .....	342
Die Funktion super() .....	344
Komposition statt Vererbung .....	346
Nachteile der Vererbung .....	348
Die Funktionen isinstance() und subclass() .....	350
Klassenmethoden .....	351
Klassenattribute .....	354
Statische Methoden .....	354
Wann brauchen Sie Klassenmerkmale und statische Methoden? .....	355
Schlagwörter der objektorientierten Programmierung .....	355
Kapselung .....	356
Polymorphismus .....	356
Wann Sie die Vererbung nicht nutzen sollten .....	357
Mehrfachvererbung .....	358
Die Reihenfolge der Methodenauflösung .....	360
Zusammenfassung .....	362
<b>17 Pythonische OOP: Eigenschaften und Dunder-Methoden</b> .....	<b>363</b>
Eigenschaften .....	364
Attribute in Eigenschaften umwandeln .....	364
Set-Methoden zur Datenvalidierung .....	367
Schreibgeschützte Eigenschaften .....	369
Wann Sie Eigenschaften verwenden sollten .....	371
Dunder-Methoden .....	371
Dunder-Methoden zur Stringdarstellung .....	372
Numerische Dunder-Methoden .....	375
Reflektierte numerische Dunder-Methoden .....	379
Direkte Dunder-Methoden für erweiterte Zuweisungsoperatoren .....	381
Dunder-Methoden für Vergleiche .....	383
Zusammenfassung .....	388
<b>Stichwortverzeichnis</b> .....	<b>391</b>