

Heiko Kalista

# Python 3

## Einsteigen und Durchstarten

HANSER

# Inhalt

<b>Vorwort</b> .....	<b>XV</b>
<b>1 Grundlagen</b> .....	<b>1</b>
1.1 Hältst Du das richtige Buch in den Händen? .....	1
1.2 Dieses Buch bricht mit einigen Konventionen! .....	2
1.3 Die Arbeit mit diesem Buch .....	3
1.4 Das Kapitel zu Minecraft .....	4
1.5 Das Begleitmaterial zum Buch .....	4
1.6 Anregungen? Kritik? Forum? .....	5
1.7 Die Geschichte von Python in 120 Wörtern .....	5
1.8 Was kann man mit Python machen (und was nicht)? .....	6
1.9 Interpreter vs. Compiler .....	7
1.10 Python 2.7 oder 3.7? .....	8
1.11 Die Entwicklungsumgebung PyCharm .....	9
1.11.1 Alternativen zu PyCharm .....	10
1.12 Python-Interpreter installieren .....	10
1.12.1 Python-Interpreter unter Windows installieren .....	11
1.12.2 Python-Interpreter unter macOS installieren .....	12
1.12.3 Python-Interpreter unter Linux installieren .....	12
1.12.4 PyCharm unter Windows installieren .....	13
1.12.5 PyCharm unter macOS installieren .....	13
1.12.6 PyCharm unter Linux installieren .....	14
1.12.7 PyCharm einrichten. ....	15
1.13 Genug geredet, los gehts! .....	17
1.13.1 Das erste Programm eingeben .....	18
1.13.2 Ausführen des ersten Beispiels .....	20
1.13.3 Laden der Beispiele .....	20
1.13.4 Der Quelltext im Detail .....	22
1.13.5 Kommentare im Detail. ....	23
1.14 PEP8 – um sie ewig zu binden .....	25

1.15	Python-Programme ohne Entwicklungsumgebung starten .....	27
1.15.1	Python-Programme unter Windows starten .....	27
1.15.2	Python-Programme unter Linux oder macOS starten .....	28
1.16	Python interaktiv .....	29
1.17	Aufgabenstellung .....	31
1.18	Kurz & knapp .....	31
<b>2</b>	<b>Variablen</b> .....	<b>33</b>
2.1	Was sind Variablen? .....	33
2.2	Statisch typisiert vs. dynamisch typisiert .....	34
2.3	Einige Details vorab – Variablen sind auch nur Namen .....	36
2.4	Richtlinien für Variablennamen .....	38
2.4.1	Von Kamelen, Schlangen und Pascal .....	38
2.4.2	Sinnvolle Variablennamen .....	39
2.5	Rechnen mit Variablen .....	40
2.5.1	Verkürzte Schreibweise bei Rechenoperationen .....	41
2.5.2	Gibt es noch weitere Datentypen? .....	42
2.5.3	Konvertierung von Datentypen .....	43
2.5.4	Besonderheiten bei der Konvertierung von Datentypen .....	44
2.6	Formatierte Ausgabe mit print() .....	45
2.7	Genauigkeit von Fließkommazahlen .....	47
2.7.1	Formatierte Ausgabe von Fließkommazahlen .....	48
2.8	Den ganzzahligen Rest einer Division bestimmen .....	49
2.9	Konstanten, Konventionen und Restriktionen .....	50
2.10	Fehlerquelltext .....	51
2.10.1	Was ist die Aufgabe des Programms? .....	52
2.10.2	Lösungsvorschlag .....	52
2.10.3	Die gemeinen Kleinigkeiten .....	54
2.11	Aufgabenstellung .....	55
2.11.1	Einige Tipps .....	55
2.11.2	Lösung zur Aufgabenstellung .....	56
2.12	Kurz & knapp .....	57
<b>3</b>	<b>Schleifen und Bedingungen</b> .....	<b>59</b>
3.1	Was sind Schleifen und Bedingungen? .....	59
3.2	Die if-Bedingung .....	60
3.2.1	Blöcke und Einrückungen .....	61
3.2.2	Arbeit sparen mit else .....	62
3.2.3	elif .....	64
3.3	Der Platzhalter pass .....	66
3.4	Blöcke im interaktiven Modus .....	67
3.5	Logische Operatoren .....	67

3.6	PyCharm - Korrekturvorschläge übernehmen .....	70
3.7	while-Schleifen .....	71
3.7.1	Eine Schleife mit break vorzeitig beenden .....	72
3.7.2	continue in while-Schleifen.....	74
3.7.3	while-else .....	75
3.8	for-Schleifen .....	76
3.8.1	Eine einfache Zählschleife.....	77
3.8.2	Ein Wort zu Schleifenvariablen .....	78
3.8.3	Die Funktion range() im Detail.....	79
3.8.4	break und continue in for-Schleifen.....	80
3.8.5	Sonderfall Unterstrich: die Wegwerfvariable .....	81
3.9	Ein kurzer Abstecher: Exceptions .....	82
3.10	Fehler Quelltext .....	83
3.10.1	Was ist die Aufgabe des Programms? .....	84
3.10.2	Lösung zum Fehler Quelltext.....	85
3.11	Aufgabenstellung .....	86
3.11.1	Einige Tipps .....	87
3.11.2	Lösungsvorschlag.....	88
3.12	Ein umfangreicheres Beispiel: Zahlenraten .....	89
3.12.1	Die Hauptschleife.....	92
3.12.2	Ein neues Spiel starten .....	92
3.12.3	Abfrage des Schwierigkeitsgrades und Beenden des Spiels.....	93
3.13	Kurz & knapp .....	93
<b>4</b>	<b>Funktionen .....</b>	<b>97</b>
4.1	Was sind Funktionen? .....	97
4.2	Eine einfache Funktion definieren .....	97
4.3	Parameter, Argumente und Rückgabewert .....	99
4.4	Weitere Möglichkeiten, Funktionen aufzurufen .....	101
4.5	Globale und lokale Variablen .....	103
4.5.1	Ein bisschen mehr Verwirrung, bitte! .....	105
4.5.2	Das Schlüsselwort global.....	106
4.5.3	Auch Parameter sind lokal .....	108
4.5.4	Ein Wort zu globalen Variablen .....	109
4.5.5	Von Schatten und Hauptprogrammen .....	109
4.6	Standardwerte für Parameter .....	113
4.7	Schlüsselwortparameter .....	114
4.8	Wann sollten Funktionen zum Einsatz kommen? .....	116
4.9	Aufgabenstellung .....	117
4.9.1	Lösung zur Aufgabenstellung.....	118
4.10	Vorgehensweise bei größeren Projekten .....	121
4.11	Rekursion .....	122

4.12	Fehlerquelltext .....	123
4.12.1	Was ist die Aufgabe des Programms? .....	124
4.12.2	Lösung zum Fehlerquelltext .....	125
4.13	Kurz & knapp .....	127
<b>5</b>	<b>Klassen</b> .....	<b>131</b>
5.1	Was ist Objektorientierung? .....	131
5.2	Eine einfache Klasse definieren .....	132
5.2.1	Aufbau einer Klasse .....	133
5.2.2	Erzeugen von Objekten .....	134
5.2.3	Verwenden der Objekte .....	135
5.3	Kontrollierter Zugriff auf Attribute: Properties .....	136
5.3.1	Getter .....	136
5.3.2	Setter .....	139
5.3.3	Wann sollten Attribute und wann Properties verwendet werden? .....	140
5.3.4	Nachträgliches Umstellen auf Properties .....	141
5.4	Dynamische Attribute .....	142
5.5	Klassenattribute .....	145
5.5.1	Stolpersteine bei Klassenattributen .....	147
5.6	Statische Methoden .....	149
5.7	Zwischenstand: Braucht man das wirklich? .....	151
5.8	Aufgabenstellung .....	152
5.8.1	Einige Tipps .....	153
5.8.2	Lösungsvorschlag .....	154
5.9	Das Gleiche ist nicht dasselbe .....	156
5.9.1	Für Fortgeschrittene: Parameterübergabe im Detail .....	159
5.10	None .....	161
5.11	Vererbung .....	163
5.11.1	Ein einfaches Beispiel .....	164
5.11.2	Überschreiben von Methoden .....	166
5.12	Wie das Smartphone erfunden wurde – oder: Mehrfachvererbung .....	171
5.13	Für Fortgeschrittene: Binden von Methoden .....	174
5.13.1	Ein Blick hinter die Kulissen .....	174
5.13.2	Klassen um eigene Funktionen erweitern .....	176
5.13.3	Statische Methoden und instanzbezogene Bindung .....	178
5.14	Überschreiben der Methode <code>__str__()</code> .....	182
5.15	Und wo ist der Fehlerquelltext? .....	183
5.16	Kurz & knapp .....	183
<b>6</b>	<b>Container</b> .....	<b>189</b>
6.1	Was sind Container? .....	189
6.1.1	Eine Anmerkung, bevor es losgeht .....	189

6.2	Listen .....	190
6.2.1	Listen erzeugen und auf Elemente zugreifen .....	190
6.2.2	Listen dynamisch erzeugen .....	191
6.2.3	Elemente löschen oder ersetzen .....	193
6.2.4	Aufgabenstellung .....	196
6.2.4.1	Einige Tipps .....	197
6.2.4.2	Lösung zur Aufgabenstellung .....	197
6.2.5	Sortieren von Listen .....	198
6.2.5.1	Eigene Sortierfunktionen .....	200
6.2.5.2	Eine weitere Möglichkeit der Sortierung: sorted .....	202
6.2.5.3	Für Fortgeschrittene: lambda – anonyme Funktionen ..	203
6.2.6	Listen verknüpfen .....	206
6.2.7	Nicht nur für Fortgeschrittene: tiefes und flaches Kopieren .....	207
6.3	Tupel .....	209
6.3.1	Unterschiede zu Listen .....	209
6.3.2	Vorsicht: Instanzen in Tupeln können geändert werden .....	211
6.3.3	Mehrere Rückgabewerte mit Tupeln .....	212
6.3.4	Für Fortgeschrittene: namedtuple .....	214
6.4	Strings .....	216
6.4.1	Für Fortgeschrittene: Darum sind Strings unveränderlich .....	217
6.4.2	Slicing .....	219
6.4.3	Arbeiten mit Strings .....	221
6.4.4	Strings verbinden .....	221
6.4.5	Zerlegen und wieder zusammensetzen: split und join .....	223
6.4.6	Strings „aufbereiten“ .....	225
6.4.7	Ändern der Groß- und Kleinschreibung .....	227
6.4.8	Strings durchsuchen .....	228
6.4.9	Aufgabenstellung .....	230
6.4.9.1	Einige Tipps .....	231
6.4.9.2	Lösungsvorschlag .....	231
6.4.10	Ersetzungen durchführen .....	233
6.5	Dictionaries .....	235
6.5.1	Grundlagen von Dictionaries .....	235
6.5.2	Vorsicht beim Zugriff! .....	236
6.5.3	Dictionaries durchlaufen und verändern .....	238
6.5.4	Elemente aus einem Dictionary entfernen .....	240
6.6	Fehlerquelltext .....	242
6.6.1	Was ist die Aufgabe des Programms? .....	242
6.6.2	Lösung zum Fehlerquelltext .....	243
6.7	Sets und Frozensets .....	245
6.7.1	Einfache Sets/Frozensets erzeugen .....	245
6.7.2	Sets: Elemente hinzufügen und entfernen .....	247
6.7.3	Mengenoperationen .....	248
6.8	Kurz & knapp .....	252

<b>7</b>	<b>Exceptions</b> .....	<b>261</b>
7.1	Was sind Exceptions? .....	261
7.2	Fehler? Die passieren mir doch nicht! .....	261
7.3	Die Mechanik von Exceptions .....	263
7.4	Abfangen unterschiedlicher Exceptions .....	265
7.5	Alle Exceptions fangen .....	267
7.6	Eigene Exceptions .....	269
7.7	else und finally .....	272
7.8	Einige allgemeine Tipps .....	275
7.9	Kurz & knapp .....	279
<b>8</b>	<b>Module und Pakete</b> .....	<b>283</b>
8.1	Module .....	283
8.1.1	Grundlagen .....	284
8.1.2	Dokumentation von Modulen .....	286
8.1.3	Umbenennen eines Namensraums .....	288
8.1.4	Selektives Importieren .....	289
8.1.5	Namensräume haben ihren Sinn .....	290
8.1.6	Batteries included! .....	290
8.1.7	Reihenfolge beim Importieren .....	291
8.1.8	Eigene Module in mehreren Projekten nutzen .....	292
8.1.9	Tipps für das Schreiben von Modulen .....	292
8.1.10	Automatische Ausführung beim Import verhindern .....	294
8.2	Pakete .....	295
8.2.1	Importieren von Modulen aus Paketen .....	296
8.2.2	Reguläre Pakete .....	297
8.2.3	Namespace Packages .....	298
8.2.4	Unterscheidung der Paketarten .....	299
8.3	Kurz & knapp .....	300
<b>9</b>	<b>Dateien und Dateisystem</b> .....	<b>303</b>
9.1	Lesen und Schreiben von Dateien .....	303
9.1.1	Eine einfache Textdatei auslesen .....	304
9.1.2	Fehler beim Öffnen von Dateien .....	305
9.1.3	Eine Datei schrittweise auslesen .....	306
9.1.4	Zeilenweises Auslesen von Textdateien .....	308
9.1.5	Textdateien schreiben .....	309
9.1.6	Übersicht möglicher Modi der Funktion open() .....	311
9.1.7	Aufgabenstellung .....	312
9.2	Dateien und Dateisystem .....	312
9.2.1	Verzeichnisse .....	313
9.2.2	Pfade und Prüfung auf deren Existenz .....	314
9.2.3	Pfade und Plattformunabhängigkeit .....	316

9.2.4	Verzeichnisse und Dateien erzeugen und löschen .....	317
9.2.5	Verzeichnisstrukturen erzeugen und löschen .....	320
9.2.6	Umbenennen von Verzeichnissen und Dateien .....	322
9.2.7	Kopieren und Verschieben .....	324
9.3	Kurz & knapp .....	326
<b>10</b>	<b>GUI-Programmierung mit tkinter .....</b>	<b>331</b>
10.1	Warum gerade tkinter? .....	331
10.2	Die Layout-Manager .....	332
10.2.1	Absolute Positionierung .....	332
10.2.2	Der pack-Manager .....	333
10.2.3	Der grid-Manager .....	338
10.2.4	Welcher Manager soll es sein? .....	340
10.3	Flexible Fenstergröße .....	341
10.4	Konfiguration von Widgets .....	344
10.5	Steuerelemente im Detail .....	346
10.5.1	Buttons .....	346
10.5.2	Kontrollvariablen .....	348
10.5.3	Eingabefelder .....	349
10.5.4	Textboxen .....	350
10.5.5	Checkboxen .....	354
10.5.6	Radiobuttons .....	356
10.5.7	Aufgabenstellung .....	357
10.5.8	Lösungsvorschlag .....	358
10.5.9	Listboxen .....	359
10.5.10	Listbox mit Scrollbalken .....	362
10.6	Fehlerquelltext .....	363
10.6.1	Was ist die Aufgabe des Programms? .....	363
10.6.2	Lösung zum Fehlerquelltext .....	364
10.7	Dialogfenster .....	366
10.8	Menüs .....	369
10.8.1	Einfache Menüs .....	369
10.8.2	Untermenüs .....	370
10.9	Kurz & knapp .....	372
<b>11</b>	<b>Debugging .....</b>	<b>377</b>
11.1	Was ist ein Debugger? .....	378
11.2	Eine einfache Fehlersuche .....	378
11.2.1	Haltepunkte setzen und entfernen .....	379
11.2.2	Das Programm durchlaufen und Werte betrachten .....	380
11.2.3	Geht es auch ohne Haltepunkte? .....	383
11.2.4	Interaktive Fehlersuche mit der Python-Konsole .....	384
11.3	Debuggen von Funktionen und Methoden .....	386

11.3.1	In Funktionen springen.....	387
11.3.2	Abschnitte überspringen und Funktionen verlassen.....	388
11.3.3	Clever springen.....	389
11.4	Watches.....	390
11.5	Haltepunkte im Detail.....	392
11.5.1	Ein Beispiel zum Experimentieren.....	392
11.5.2	Verwalten von Haltepunkten.....	393
11.5.3	Unterbrechen oder nicht?.....	397
11.5.4	Bedingte Haltepunkte.....	397
11.5.5	Protokollierung.....	398
11.5.6	Temporäre Haltepunkte.....	399
11.5.7	Verkettete Haltepunkte.....	400
11.5.8	Haltepunkte für Exceptions.....	400
11.5.9	Kombination der Optionen.....	401
11.6	Einsatz in der Praxis.....	402
<b>12</b>	<b>Versionsverwaltung mit Git.....</b>	<b>403</b>
12.1	Der vermeintlich leichte Weg.....	403
12.2	Wie funktionieren Versionskontrollsysteme?.....	404
12.2.1	Verallgemeinerte Arbeitsweise.....	404
12.2.2	Zentrale und verteilte Versionskontrolle.....	405
12.3	Was ist Git?.....	405
12.4	Interessiert mich nicht, ich arbeite alleine!.....	407
12.5	Vorbereitungen.....	407
12.5.1	Git unter Windows installieren.....	407
12.5.2	Git unter macOS installieren.....	408
12.5.3	Git unter Linux installieren.....	409
12.5.4	GitHub.....	409
12.6	Los geht's - Git lokal verwenden.....	410
12.6.1	Ein bestehendes Projekt unter Versionskontrolle stellen.....	410
12.6.2	Dateien hinzufügen.....	412
12.6.3	Commit/Einchecken.....	413
12.6.4	Änderungen vornehmen und überprüfen.....	414
12.6.5	Änderungen rückgängig machen.....	416
12.6.6	Betrachten der Historie.....	417
12.6.7	Zu älteren Versionen zurückkehren - Möglichkeit 1.....	419
12.6.8	Dateien ignorieren.....	420
12.7	Zusammenarbeit über ein Remote Repository.....	421
12.7.1	Projekt auf GitHub veröffentlichen.....	422
12.7.2	Ein Repository klonen.....	422
12.7.3	Änderungen pushen.....	424
12.7.4	Pull.....	424
12.7.5	Wer hat's erfunden?.....	425

12.7.6	Automatisches Merging	425
12.7.7	Mergen und Konflikte beheben	426
12.8	Branching	430
12.8.1	Um was geht es?	430
12.8.2	Einen Branch erzeugen und damit arbeiten	431
12.8.3	Zwischen Branches wechseln	433
12.8.4	Branches mergen	433
12.8.5	Branches pushen	435
12.8.6	Fetch	435
12.8.7	Zu älteren Versionen zurückkehren – Möglichkeit 2	436
12.9	Weitere nützliche Features	436
12.9.1	Stashing - Änderungen temporär speichern	437
12.9.2	Commits korrigieren	439
12.9.3	Tagging	439
12.10	Noch ein paar Tipps	440
12.10.1	Zwei einfache Branching-Modelle	440
12.10.2	Atomare Commits	441
12.10.3	Test-Repository griffbereit halten	442
12.10.4	Andere Git-Clients	442
12.11	Kurz & knapp	443
<b>13</b>	<b>Minecraft auf dem Raspberry Pi</b>	<b>447</b>
13.1	Um was geht es in diesem Kapitel?	447
13.1.1	Der Raspberry Pi – ein kleines Kraftpaket	447
13.1.2	Minecraft Pi	448
13.1.3	Was wird benötigt?	449
13.2	Der Sprung ins kalte Wasser	450
13.3	Die Entwicklungsumgebungen	450
13.4	Den Raspberry Pi einrichten	451
13.4.1	Erstellen der SD-Karte	451
13.4.2	Einstellen des Tastaturlayouts	452
13.4.3	Minecraft starten	452
13.4.4	Die Steuerung	452
13.5	Der erste Testlauf	454
13.5.1	Arbeiten mit IDLE	454
13.5.2	Arbeiten mit Thonny	456
13.6	Das Begleitmaterial	457
13.7	Die Minecraft Python API	457
13.7.1	Quellen und weitere Informationen:	458
13.7.2	Eine Übersicht der Minecraft API	458
13.7.3	Die Klasse Minecraft	458
13.7.4	Die Klasse CmdCamera	461
13.7.5	Die Klasse CmdPlayer	462

13.7.6	Die Klasse CmdEntity .....	462
13.7.7	Die Klasse Block .....	463
13.7.7.1	Eine Übersicht der wichtigsten Blöcke .....	463
13.7.8	Noch einmal alles zusammen .....	464
13.8	Beispiele zu Schleifen und Bedingungen .....	466
13.8.1	Blocktypen ausprobieren .....	466
13.8.2	Spielfigur automatisch durch die Welt bewegen .....	468
13.8.3	Eine Treppe bauen .....	469
13.8.4	Eine Pyramide bauen .....	470
13.9	Beispiele zu Funktionen .....	472
13.9.1	Swimmingpools bauen .....	472
13.9.2	Moderne Kunst? .....	474
13.10	Beispiele zu Klassen .....	477
13.10.1	Blöcke regnen lassen .....	477
13.10.2	Blinklichter .....	480
13.11	Beispiele zu Containern .....	483
13.11.1	Lichterkette .....	483
13.11.2	Mengenoperationen .....	485
13.12	Ein Beispiel zu Modulen und Paketen .....	487
13.13	exit() – wie geht es weiter? .....	490
<b>Index</b>	.....	<b>491</b>