

Einstieg in Python

Grundlagen der Python-Programmierung
leicht und verständlich erklärt

DAS INHALTS- VERZEICHNIS

» Hier geht's
direkt
zum Buch

Inhaltsverzeichnis

Vorwort	xvii
----------------------	-------------

I Einstieg	1
1 Einführung	3
1.1 Python im Überblick	3
1.2 Los geht's – Installation	6
1.2.1 Python-Download	7
1.2.2 Installation von Python	7
1.2.3 Nacharbeiten nach der Python-Installation	8
1.2.4 Python-Installation prüfen	9
1.2.5 Python-Programm als Skript ausführen	10
1.3 Arbeiten mit der Kommandozeile (Kurzeinführung)	10
1.3.1 Für macOS	11
1.3.2 Für Windows	13
1.3.3 IDLE als Alternative	15
1.4 Entwicklungsumgebung PyCharm (Optional)	17
1.4.1 Installation von PyCharm	19
1.4.2 PyCharm starten	20
1.4.3 Erstes Projekt in PyCharm	22
1.4.4 Erstes Modul in PyCharm	22
1.5 Ausprobieren der Beispiele	26
1.5.1 Python-Kommandozeileninterpreter	26
1.5.2 Texteditor und direkte Ausführung	27
1.5.3 IDE	28
1.5.4 Abschließender Hinweis	28
2 Schnelleinstieg	29
2.1 Hallo Welt (Hello World)	29
2.2 Variablen und Datentypen	30
2.2.1 Definition von Variablen	30
2.2.2 Variablen und Typen	31

2.2.3	Ausgaben mit <code>print()</code>	33
2.2.4	Bezeichner (Variablenamen)	35
2.3	Operatoren im Überblick	36
2.3.1	Arithmetische Operatoren	37
2.3.2	Zuweisungsoperatoren	39
2.3.3	Vergleichsoperatoren	41
2.3.4	Logische Operatoren	42
2.4	Fallunterscheidungen	43
2.5	Funktionen	46
2.5.1	Eigene Funktionen definieren	47
2.5.2	Nützliche Beispiele aus Python	50
2.6	Fehlerbehandlung und Exceptions	50
2.7	Kommentare	51
2.8	Module	52
2.8.1	Imports – Einbinden anderer Funktionalitäten	52
2.8.2	Zusammenfassung und Ergänzendes	54
2.9	Built-in-Datentypen	55
2.9.1	Listen (<code>list</code>)	55
2.9.2	Tupel (<code>tuple</code>)	56
2.9.3	Mengen (<code>set</code>)	57
2.9.4	Dictionaries (<code>dict</code>)	58
2.10	Schleifen	59
2.10.1	Besonderheit: <code>Ranges</code>	59
2.10.2	Indexbasierte <code>for-in</code> -Schleife	59
2.10.3	Wertebasierte <code>for-in</code> -Schleife	64
2.10.4	Die <code>for-in-enumerate</code> -Schleife mit Index und Wert	65
2.10.5	Die <code>while</code> -Schleife	67
2.11	Weiterführende Informationen	68
2.12	Aufgaben und Lösungen	69
2.12.1	Aufgabe 1: Mathematische Berechnungen	69
2.12.2	Aufgabe 2: Bedingung vereinfachen	69
2.12.3	Aufgabe 3: Funktion und <code>if</code>	70
2.12.4	Aufgabe 4: Selbstabholerrabatt	71
2.12.5	Aufgabe 5: Schleifen mit Berechnungen	72
2.12.6	Aufgabe 6: Schleifen und fixe Schrittweite	73
2.12.7	Aufgabe 7: Schleifen mit variabler Schrittweite	73
2.12.8	Aufgabe 8: Verschachtelte Schleifen – Variante 1	74
2.12.9	Aufgabe 9: Verschachtelte Schleifen – Variante 2	76
2.12.10	Aufgabe 10: Verschachtelte Schleifen – Variante 3	77

3	Strings	79
3.1	Schnelleinstieg	79
3.1.1	Gebräuchliche Stringaktionen	79
3.1.2	Suchen, Enthaltensein und Ersetzen	88
3.1.3	Informationen extrahieren und formatieren	90
3.1.4	Praxisrelevante Funktionen im Kurzüberblick	92
3.2	Nächste Schritte	93
3.2.1	Zeichenverarbeitung	93
3.2.2	Strings und Listen	93
3.2.3	Mehrzeilige Strings	95
3.3	Aufgaben und Lösungen	97
3.3.1	Aufgabe 1: Länge, Zeichen und Enthaltensein	97
3.3.2	Aufgabe 2: Zeichen wiederholen	97
3.3.3	Aufgabe 3: Vokale raten	98
3.3.4	Aufgabe 4: String Merge	100
4	Klassen und Objektorientierung	101
4.1	Schnelleinstieg	101
4.1.1	Grundlagen zu Klassen und Objekten	102
4.1.2	Eigenschaften (Attribute)	105
4.1.3	Verhalten (Methoden)	107
4.1.4	Typprüfung mit <code>isinstance()</code>	110
4.1.5	Objekte vergleichen – die Rolle von <code>__eq__()</code>	111
4.2	Nächste Schritte	114
4.2.1	Klassen ausführbar machen	114
4.2.2	Packages	117
4.2.3	Übergang zum Einsatz einer IDE	118
4.2.4	Verstecken von Informationen	121
4.2.5	Packages: Auswirkungen auf unsere Applikation	125
4.3	Vererbung	129
4.4	Aufgaben und Lösungen	132
4.4.1	Aufgabe 1: Superheld	132
4.4.2	Aufgabe 2: Zähler	133
4.4.3	Aufgabe 3: Objekte mit Dictionary selbst gebaut	135
5	Collections	137
5.1	Schnelleinstieg	137
5.1.1	Die Klasse <code>list</code>	137
5.1.2	Die Klasse <code>set</code>	144
5.1.3	Die Klasse <code>dict</code>	147

5.2	Nächste Schritte	151
5.2.1	Comprehensions	151
5.2.2	Slicing – Zugriff auf Teilbereiche von Listen	154
5.2.3	Sortierung – <code>sort()</code> / <code>sorted()</code>	156
5.2.4	Tauschen von Elementen – <code>swap()</code>	159
5.2.5	Reihenfolge umkehren – <code>reverse()</code> und <code>reversed()</code> ...	161
5.2.6	Mehrdimensionale Listen	162
5.3	Aufgaben und Lösungen	166
5.3.1	Aufgabe 1: Tennisverein-Mitgliederliste	166
5.3.2	Aufgabe 2: Liste mit Farbnamen füllen und filtern	167
5.3.3	Aufgabe 3: Duplikate entfernen – Variante 1	167
5.3.4	Aufgabe 4: Duplikate entfernen – Variante 2	168
5.3.5	Aufgabe 5: Hauptstädte	169
5.3.6	Aufgabe 6: Häufigkeiten von Namen	169
5.3.7	Aufgabe 7: Rotation um eine oder mehrere Positionen	170
5.3.8	Aufgabe 8: Dreieckige Liste: Upside Down	172
6	Ergänzendes Wissen	175
6.1	Benutzereingaben <code>input()</code>	175
6.2	Zufallswerte und das Modul <code>random</code>	176
6.3	Besonderheiten von Parametern	178
6.3.1	Parameter mit Position bzw. Name	178
6.3.2	Parameter mit Defaultwert	179
6.3.3	Var Args – variable Anzahl an Argumenten	179
6.4	Ternary-Operator	183
6.5	Aufzählungen mit <code>Enum</code>	184
6.6	Fallunterscheidungen mit <code>match</code>	186
6.7	<code>break</code> , <code>continue</code> und <code>else</code> in Schleifen	189
6.7.1	Funktionsweise von <code>break</code> und <code>continue</code>	189
6.7.2	Wie macht man es besser?	192
6.7.3	Besonderheit: <code>else</code> in Schleifen	195
6.8	Ausdrücke mit <code>eval()</code> auswerten	196
6.9	Rekursion	197
6.9.1	Einführendes Beispiel: Fakultät	197
6.9.2	Weiterführendes Beispiel: Fibonacci-Zahlen	198
6.9.3	Weiterführendes Beispiel: Lineal	199
6.10	Praxisbeispiel: Flächen füllen	200
6.11	Aufgaben und Lösungen	202
6.11.1	Aufgabe 1: Würfelspiel	202
6.11.2	Aufgabe 2: Temperaturumrechnung	203
6.11.3	Aufgabe 3: Palindrom-Prüfung mit Rekursion	204
6.11.4	Aufgabe 4: Einarmiger Bandit	205
6.11.5	Aufgabe 5: BMI-Rechner	206

7	Verarbeitung von Dateien	207
7.1	Schnelleinstieg	207
7.1.1	Anlegen von Dateien und Verzeichnissen	207
7.1.2	Informationen zu Verzeichnissen und Dateien	209
7.1.3	Informationen in Dateien schreiben und daraus lesen	210
7.1.4	Diverse Informationen ermitteln	216
7.1.5	Kopieren und Umbenennen	217
7.1.6	Löschen	218
7.2	Praxisbeispiel: Directory-Baum darstellen	219
7.2.1	Basisvariante	220
7.2.2	Variante mit schönerer Darstellung	221
7.2.3	Finale Variante mit ausgeklügelter Darstellung	222
7.3	JSON-Verarbeitung	223
7.3.1	JSON in eine Datei schreiben	223
7.3.2	Lesen von JSON aus einer Datei	224
7.3.3	Pretty Printing	225
7.4	Aufgaben und Lösungen	226
7.4.1	Aufgabe 1: Texte in Datei schreiben und wieder lesen	226
7.4.2	Aufgabe 2: Dateigrößen	226
7.4.3	Aufgabe 3: Existenzprüfung	227
7.4.4	Aufgabe 4: Rechteprüfung	228
7.4.5	Aufgabe 5: Verzeichnisinhalt auflisten	228

II Aufstieg 231

8	Collections Advanced	233
8.1	Sequenzielle Datentypen	233
8.2	Spaß mit Listen	235
8.2.1	Additionen und Multiplikationen mit Listen	235
8.2.2	Trickserei mit Slicing	236
8.3	Iteratoren	238
8.4	Generatoren	241
8.5	Datencontainer mit <code>namedtuple</code>	244
8.6	Einstieg in Lambdas	248
8.6.1	Syntax von Lambdas	248
8.6.2	Lambdas im Einsatz mit <code>filter()</code> und <code>map()</code>	249
8.6.3	Lambdas im Einsatz mit <code>sort()</code>	251

8.7	Aufgaben und Lösungen	254
8.7.1	Aufgabe 1: Obstkorb	254
8.7.2	Aufgabe 2: Erwachsene aus Personenliste extrahieren	254
8.7.3	Aufgabe 3: Eigene Implementierung von <code>rindex()</code>	255
8.7.4	Aufgabe 4: Every-N-th-Iterator	256
8.7.5	Aufgabe 5: Greeting-Generator	257
8.7.6	Aufgabe 6: Fibonacci-Generator	258
8.7.7	Aufgabe 7: Initialisierung mit List Comprehension und Multiplikation	260
9	Fehlerbehandlung mit Exceptions	261
9.1	Schnelleinstieg	261
9.1.1	Fehlerbehandlung	262
9.1.2	Exceptions selbst auslösen – <code>raise</code>	267
9.1.3	Eigene Exception-Typen definieren	268
9.1.4	Exceptions propagieren – <code>throws</code>	269
9.2	Fehlerbehandlung in der Praxis	271
9.2.1	Exceptions für Bereichsprüfungen	271
9.2.2	Elegante Prüfungen mit <code>assert</code>	273
9.3	Automatic Resource Management (<code>with</code>)	275
9.4	Aufgaben und Lösungen	276
9.4.1	Aufgabe 1: Abgesicherter Indexzugriff – Kür mit Fallback ...	276
9.4.2	Aufgabe 2: Resource Handling	277
10	Datumsverarbeitung	279
10.1	Schnelleinstieg	279
10.1.1	Zeitpunkte und die Klasse <code>datetime</code>	279
10.1.2	Datumswerte und die Klasse <code>date</code>	281
10.1.3	Zeit und die Klasse <code>time</code>	284
10.1.4	Zeitdifferenzen und die Klasse <code>timedelta</code>	285
10.1.5	Berechnungen mit Datumswerten	286
10.1.6	Formatierung und Parsing	288
10.2	Praxisbeispiel: Kalenderausgabe	289
10.3	Aufgaben und Lösungen	293
10.3.1	Aufgabe 1: Wochentage	293
10.3.2	Aufgabe 2: Freitag, der 13.	294
10.3.3	Aufgabe 3: Mehrmals Freitag, der 13.	295
10.3.4	Aufgabe 4: Schaltjahre	296

11	Bildverarbeitung und Kontakt zur Außenwelt	299
11.1	Die Bibliothek <i>Pillow</i> / <i>PIL</i>	299
11.1.1	Installation und Import	299
11.1.2	Bildverarbeitung	300
11.1.3	Praxisbeispiel	305
11.1.4	Weiterführende Informationen	306
11.2	Das Modul <code>requests</code>	307
11.2.1	Installation und Import	307
11.2.2	Request absetzen – auf Webseite zugreifen	307
11.2.3	Wesentliche HTTP-Kommandos (GET, POST, PUT, DELETE, HEAD)	308
11.2.4	Response als Datei sichern	309
11.2.5	Weiterführende Informationen	310
11.3	Das Modul <code>webbrowser</code>	310
11.3.1	Datei im Webbrowser öffnen	310
11.3.2	Google-Suche programmatisch ausführen	311
12	LLMs mit Python im Kurzüberblick	315
12.1	Einführung	315
12.1.1	Was sind LLMs? Geschichte und Einsatzgebiete	315
12.2	Einführende Beispiele mit <i>transformers</i>	317
12.2.1	Textgenerierung mit <i>transformers</i>	318
12.2.2	Analyse von Texten	319
12.2.3	Was haben wir bisher gelernt?	320
12.3	Experimente mit <i>OpenAI</i>	321
12.3.1	Textgenerierung	323
12.3.2	Sourcecode erklären lassen	324
12.3.3	Sourcecode generieren lassen	326
12.3.4	Textnachricht in MP3 wandeln	328
12.3.5	Bildgenerierung	329
12.3.6	Weiterführende Infos	332

III Praxisbeispiele **333**

13 Praxisbeispiel: Tic Tac Toe	335
13.1 Spielfeld initialisieren und darstellen	335
13.2 Setzen der Steine	336
13.3 Prüfen auf Sieg	337
13.4 Bausteine im Einsatz	339
14 Praxisbeispiel: CSV-Highscore-Liste einlesen	341
14.1 Verarbeitung von Spielständen (Highscores)	341
14.2 Extraktion der Daten	342
14.3 Besonderheiten der Implementierung	344
15 Praxisbeispiel: Worträtsel	347
15.1 Applikationsdesign – Vorüberlegungen zur Strukturierung	348
15.2 Einlesen der verfügbaren Wörter	348
15.3 Hilfsdatenstrukturen	350
15.4 Datenmodell	351
15.4.1 Datenspeicherung und Initialisierung	351
15.4.2 Zufällige Wahl von Richtung, Position, Wort und Buchstabe	352
15.4.3 Algorithmus zum Verstecken von Wörtern	352
15.4.4 Wort prüfen und platzieren	353
15.5 HTML-Erzeugung	354
15.6 Ausgabe als HTML und Darstellung im Browser	356
15.7 Hauptapplikation	356
15.8 Fazit	358

IV Schlussgedanken **359**

16 Gute Angewohnheiten	361
16.1 Grundregeln eines guten Programmierstils	361
16.1.1 Keep It Human-Readable	361
16.1.2 Keep it Understandable	362
16.2 Coding Conventions	364
16.2.1 PEP 8 – Coding Standard	364
16.2.2 Zen of Python	366
16.2.3 Namensgebung	366
16.2.4 Dokumentation	369
16.2.5 Programmdesign	369
16.2.6 Parameterlisten	370
16.2.7 Logik und Kontrollfluss	371

16.3 Auch ans Testen denken	371
16.3.1 Das Pytest-Framework	371
16.3.2 Schreiben und Ausführen von Tests	372
17 Schlusswort	377

V Anhang	379
-----------------	------------

A Schlüsselwörter im Überblick	381
A.1 Schlüsselwörter im Überblick	381
B Schnelleinstieg Python-REPL	385
B.1 Python-REPL	385
C Neuerungen in Python 3.13	389
C.1 Verbesserungen bei Fehlermeldungen	389
C.1.1 Namenskonflikte bei Modulen	390
C.1.2 Tippfehler bei benannten Parametern	391
C.2 Verbesserungen in der Python-REPL	392
C.3 Verschiedenes	393
Literaturverzeichnis	395
Index	397