

Inhalt

Teil I	Einleitung	1
1	Vorwort	3
2	Danksagung	5
3	Einleitung	7
3.1	Einführung	7
3.2	Zielsetzung des Buches	7
3.3	Aufbau des Buches	8
3.4	Leserschaft	10
3.5	Zusätzliche Unterlagen	10
Teil II	Python-Grundlagen unter funktionalen Aspekten	11
4	Variablen und Datentypen	13
4.1	Variablen	13
4.1.1	Gültige Variablennamen	15
4.1.2	Konventionen für Variablennamen	15
4.2	Übersicht Datenstrukturen	16
4.2.1	Unveränderliche (immutable) Datentypen	16
4.2.2	Veränderliche (mutable) Datentypen	16
4.3	Datenstrukturen im Detail	17
4.3.1	Integer	17
4.3.2	Floats	17

4.3.3	Zeichenketten oder Strings	18
4.3.4	Bytesequenz	20
4.3.5	Listen.....	21
4.3.6	Mengen	22
4.3.7	Dictionaries.....	23
5	Kontrollstrukturen	25
5.1	Sequenz	26
5.2	Bedingte Anweisungen	26
5.2.1	Vollständiges if	26
5.2.2	Ternäres if	28
5.3	Schleifen	29
5.3.1	while-Schleife	29
5.3.1.1	Allgemeine Arbeitsweise	29
5.3.1.2	while-Schleife mit else	30
5.3.2	while-Schleife als rekursive Funktion.....	31
5.3.3	Allgemeine while-Funktion	33
5.3.4	for-Schleife	34
5.3.5	for-Schleife in funktionaler Programmierung	36
6	Das Modul collections	37
6.1	Übersicht	37
6.2	namedtuple.....	38
6.3	Deque.....	41
6.4	ChainMap	42
6.5	Counter	44
Teil III	Funktionale Programmierung	47
7	Begriffsbestimmung	49
7.1	Einführung	49
7.2	Verschiedene Programmierparadigmen	50
7.3	Funktionale Programmierung	50
8	Funktionen	53
8.1	Einleitung	53
8.2	Funktionen	53
8.2.1	Einfache Funktionen	53

8.2.2	Typehints	55
8.2.2.1	Einführende Beispiele	55
8.2.2.2	Typehints mit mypy	56
8.2.3	Default-Parameter und Schlüsselwortparameter	57
8.2.4	Weitere Typehints	57
8.2.5	Lokale Funktionen	58
8.2.6	Globale und lokale Variablen in Funktionen	59
8.2.7	Gültigkeit von Variablen in verschachtelten Funktionen	60
8.3	Dataclasses und Pattern Matching	63
8.3.1	Dataclasses	63
8.3.1.1	Standardargumente und Fabrikfunktionen	64
8.3.1.2	Unterschiede zum namedtuple	66
8.3.2	Pattern Matching	69
8.3.2.1	Einführung	69
8.3.2.2	Binärbäume mit Pattern Matching durchlaufen	69
8.4	Aufgaben.....	70
8.5	lambda	71
8.5.1	lambda mit sorted.....	73
8.6	Aufgaben.....	74
8.7	Rekursive Funktionen	75
8.7.1	Einführung	75
8.7.2	Definition der Rekursion	76
8.7.3	Iterative Lösung im Vergleich zur rekursiven Lösung	77
8.7.4	Kategorien der Rekursion	77
8.7.4.1	Lineare Rekursion.....	78
8.7.4.2	Gegenseitige/Wechselseitige Rekursion	79
8.7.4.3	Baumartige Rekursion	80
8.7.4.4	Endrekursion.....	85
8.8	Aufgaben.....	85
8.9	Funktionen als Erste-Klasse-Objekte	87
8.9.1	Definition	87
8.9.2	Zuweisung an Variable	88
8.9.3	Funktionen in Datenstrukturen	89
8.9.4	Funktionen als Argumente	92
8.9.5	Funktionen innerhalb von Funktionen	93
8.9.5.1	Kapselung und Verbergen.....	93
8.9.5.2	Fabrikfunktionen	95
8.10	Aufgaben.....	98

9	Dekoratoren	99
9.1	Einführung Dekoratoren.....	99
9.2	Ein einfacher Dekorator	100
9.3	@-Syntax für Dekoratoren	101
9.4	Dekoratoren für beliebige Signaturen	102
9.5	Anwendungsfälle für Dekoratoren.....	103
9.5.1	Erweiterung von Funktionen	104
9.5.2	Logging-Dekorator	105
9.5.3	Authentifizierung und Autorisierung	106
9.5.4	Validierung: Überprüfung von Argumenten	107
9.5.5	Profiling: Funktionsaufrufe mit einem Dekorator zählen	108
9.6	Dekoratoren mit Parametern	109
9.6.1	Einführendes Beispiel	109
9.6.2	Weiteres Beispiel.....	110
9.7	Benutzung von Wraps aus functools	112
9.8	Mehrfache Dekoration einer Funktion	114
9.8.1	Veranschaulichung	114
9.8.2	Python-Beispiel	115
9.8.3	Praktisches Beispiel einer Mehrfachdekoration	116
9.9	Eine Klasse als Dekorator benutzen	119
9.10	Klassendekoration.....	120
9.11	Dekorator-Aufgaben	123
10	Memoisation	125
10.1	Bedeutung und Herkunft des Begriffs	125
10.2	Memoisation mit Dekoratorfunktionen	126
10.3	Memoisation mit einer Klasse	129
10.4	Memoisation mit functools.lru_cache	129
10.5	Aufgaben zur Memoisation	131
11	Closures	133
11.1	Einleitung	133
11.1.1	Definition	134
11.1.2	Praktische Anwendungen von Closures	136
11.2	Aufgaben.....	136

12	Komposition von Funktionen	137
12.1	Einführung	137
12.2	Funktionskomposition in Python	138
12.3	Komposition mit beliebiger Argumentenzahl	140
12.4	Komposition einer beliebigen Anzahl von Funktionen	141
12.5	Aufgaben.....	142
13	Currying in Python	143
13.1	Einführung	143
13.1.1	Zugrundeliegende Idee	143
13.1.2	Herkunft des Names	144
13.2	Currying von Funktionen	144
13.2.1	Definition und Beispiel	144
13.2.2	BMI als Beispiel	145
13.2.3	BMI als Beispiel	145
13.3	Benutzung von partial	146
13.4	Praktisches Beispiel für Currying: E-Mail-Vorlagen	146
13.5	Dekorator für Currying	148
13.6	Weitere Beispiele.....	150
13.6.1	Arithmetische Operatoren	150
13.6.2	Beispiel aus der Finanzwelt	151
13.6.3	Currying zur Listenfilterung	154
13.7	Currying-Funktion mit einer beliebigen Anzahl von Parametern	154
13.8	Aufgaben.....	157
14	Funktionale Emulation von OOP-Konzepten.....	159
14.1	Einführung	159
14.2	Imitation einer Klasse durch eine Funktion	160
14.2.1	Erstes Beispiel: Geradengenerierung	160
14.2.2	Private Attribute	161
14.2.3	Klasse mit Gettern und Settern als Funktion	162
14.2.4	Nachahmung von Vererbung	165
14.2.5	Überlagern von Funktionen	169
14.3	Aufgaben.....	171

15 Generatoren und Iteratoren	175
15.1 Definitionen	175
15.2 Einführung	175
15.2.1 Iterierbar	176
15.2.2 Iterator und Arbeitsweise der for-Schleife	176
15.3 Eigenen Iterator erzeugen	179
15.3.1 Beispiel einer Iteratorklasse.....	179
15.4 Generatoren und Generatorfunktionen	180
15.4.1 Unendliche Generatoren	184
15.5 Endlos-Generatoren zähmen mit firstn und islice	185
15.6 Beispiele aus der Kombinatorik	188
15.6.1 Permutationen	188
15.6.2 Variationen und Kombinationen	189
15.7 Generator-Ausdrücke.....	191
15.8 yield from	192
15.9 return-Anweisungen in Generatoren	193
15.10 send-Methode	196
15.10.1 Arbeitsweise	196
15.10.2 Umprogrammierung mittels send.....	198
15.10.3 Radio-Beispiel mit send	199
15.11 Die close-Methode	201
15.12 Die throw-Methode	202
15.13 Dekoration von Generatoren	206
15.14 Aufgaben.....	207
 16 Iteratoren der Standardbibliothek.....	 211
16.1 Einführung	211
16.2 Wichtige Iteratoren und iteratorähnliche Funktionen in der Python-Standardbibliothek.....	212
16.3 enumerate	212
16.4 map, filter und reduce	214
16.4.1 map.....	214
16.4.2 Filtern von sequentiellen Datentypen mittels „filter“	215
16.4.3 reduce	216
16.4.4 Zusammenspiel von map, filter und reduce	218
16.5 Listen-Abstraktion.....	218
16.5.1 Alternative zu map und filter	218
16.5.2 Syntax	219

16.5.3	Weitere Beispiele	220
16.5.4	Die zugrunde liegende Idee	220
16.5.5	Anspruchsvolleres Beispiel	221
16.5.6	Mengen-Abstraktion	222
16.5.7	Generatoren-Abstraktion	222
16.5.8	map und filter oder Listen-Abstraktion	223
16.5.9	Dict-Komprehension in Python	223
16.6	reversed	224
16.7	Die zip-Funktion.....	224
16.8	zip in Kombination mit map und filter	226
16.9	Aufgaben.....	227
17	Das Modul itertools	229
17.1	Übersicht	229
17.2	Unendliche Iteratoren	230
17.2.1	itertools.count	231
17.2.1.1	Beispiel: Teilbarkeit	232
17.2.1.2	Beispiel: Quadratpolynome	232
17.2.1.3	Beispiel: Bestellungsmanager	233
17.2.2	chain und chain.from_iterable.....	234
17.2.3	cycle	237
17.2.4	tee.....	237
17.2.4.1	Effiziente Implementierung von cycle	238
17.2.4.2	Beispiel: Aufgaben und Mitarbeiter.....	239
17.2.5	repeat	240
17.3	Iteratoren über endliche Sequenzen	240
17.3.1	slice und islice	241
17.3.1.1	slice.....	241
17.3.1.2	islice	241
17.3.1.3	slice und islice im Vergleich	243
17.3.2	accumulate	244
17.3.2.1	Praktisches Börsenbeispiel	244
17.3.3	compress	246
17.3.3.1	Einführung	246
17.3.3.2	Umfangreiches Beispiel	247
17.3.4	dropwhile	248
17.3.5	takewhile	249
17.3.6	filterfalse	251

17.3.7	groupby	252
17.3.8	pairwise	254
17.3.9	n_grams und map_n	256
17.3.10	zip_longest	258
17.3.11	starmap	258
17.4	Kombinatorische Generatoren	260
17.4.1	product	260
17.4.2	permutations.....	262
17.4.3	combinations	265
17.4.3.1	Arbeitsweise	265
17.4.3.2	Beispiel: Speisekartengenerierung	265
17.4.3.3	Beispiel: Benachbarte Kantengraphen.....	268
17.4.4	combinations_with_replacement	269
17.4.5	Umfangreiches praktisches Beispiel: csv-Datei lesen.....	270
17.5	Aufgaben.....	274
Teil IV Lösungen zu den Aufgaben		279
18	Lösungen zu den Aufgaben	281
18.1	Lösungen zu Abschnitt 8.9 (Funktionen als Erste-Klasse-Objekte)	281
18.2	Lösungen zu Abschnitt 8.3 (Dataclasses und Pattern Matching)	283
18.3	Lösungen zu Abschnitt 8.5 (lambda)	284
18.4	Lösungen zu Abschnitt 8.7 (Rekursive Funktionen)	285
18.5	Lösungen zu Abschnitt 8.9 (Funktionen als Erste-Klasse-Objekte)	289
18.6	Lösungen zu Kapitel 9 (Dekoratoren)	290
18.7	Lösungen zu Kapitel 10 (Memoisation)	298
18.8	Lösungen zu Kapitel 11 (Closures).....	303
18.9	Lösungen zu Kapitel 16 (Iteratoren der Standardbibliothek)	304
18.10	Lösungen zu Abschnitt 16.5 (Listen-Abstraktion)	306
18.11	Lösungen zu Kapitel 15 (Generatoren und Iteratoren)	308
18.12	Lösungen zu Kapitel 17 (Das Modul itertools)	316
18.13	Lösungen zu Kapitel 12 (Komposition von Funktionen)	331
18.14	Lösungen zu Kapitel 13 (Currying in Python)	334
18.15	Lösungen zu Kapitel 14 (Funktionale Emulation von OOP-Konzepten)	335
Stichwortverzeichnis		341