

Der Inhalt (im Überblick)

1	Die Grundlagen: <i>Volle Kraft voraus!</i>	1
2	Listendaten: <i>Mit geordneten Daten arbeiten</i>	47
3	Strukturierte Daten: <i>Mit strukturierten Daten arbeiten</i>	95
4	Code wiederverwenden: <i>Funktionen und Module</i>	145
5	Eine Webapplikation erstellen: <i>Auf ins wahre Leben!</i>	195
6	Daten speichern und bearbeiten: <i>Wo kommen die Daten hin?</i>	243
7	Datenbanken benutzen: <i>Die DB-API von Python verwenden</i>	281
8	Ein bisschen Klasse: <i>Verhalten und Zustand abstrahieren</i>	309
9	Das Kontextmanagement-Protokoll: <i>Sich in Pythons with-Anweisung einklinken</i>	335
10	Funktionsdekoratoren: <i>Funktionen verpacken</i>	363
11	Mit Ausnahmen umgehen: <i>Was zu tun ist, wenn mal etwas schiefgeht</i>	413
11 ^{3/4}	Ein bisschen Threading: <i>Taten statt Warten</i>	461
12	Fortgeschrittene Iteration: <i>Schleifen wie verrückt</i>	477
A	Installation: <i>Python installieren</i>	521
B	PythonAnywhere: <i>Ihre Webapplikation bereitstellen</i>	529
C	Die 10 wichtigsten Dinge, die wir nicht behandelt haben: <i>Es gibt immer noch etwas zu lernen</i>	539
D	Die 10 wichtigsten Projekte, die wir nicht behandelt haben: <i>Noch mehr Werkzeuge, Bibliotheken und Module</i>	551
E	Mitmachen: <i>Die Python-Gemeinschaft</i>	563
	Index	573

Der Inhalt (jetzt ausführlich)

Einführung

Für wen ist dieses Buch?	xxiv
Wir wissen, was Sie gerade denken	xxv
Und wir wissen, was Ihr Gehirn gerade denkt	xxv
Metakognition: Nachdenken übers Denken	xxvii
Das haben WIR getan	xxxii
Lies mich	xxx

v

1

Die Grundlagen

Volle Kraft voraus!

Die IDLE-Fenster verstehen	4
Code ausführen, eine Anweisung nach der anderen	8
Funktionen + Module = Standardbibliothek	9
Datenstrukturen sind schon eingebaut	13
Methodenaufrufe haben Ergebnisse	14
Entscheiden, wann Codeblöcke ausgeführt werden	15
Welche »else« will schon mit »if«?	17
Suiten können selbst Suiten enthalten	18
Zurück zur Python-Shell	22
Experimente auf der Shell	23
Über eine Folge von Objekten iterieren	24
Eine bestimmte Anzahl von Wiederholungen ausführen	25
Das Ergebnis von Aufgabe 1 auf unseren Code anwenden	26
Die Ausführung unterbrechen	28
Zufallszahlen mit Python erzeugen	30
Eine ernsthafte Businessapplikation programmieren	38
Machen die Einrückungen Sie verrückt?	40
Mit dem help-Befehl des Interpreters den Hilfetext zu einer Funktion anzeigen	41
Mit Wertebereichen experimentieren	42
Der Code aus Kapitel 1	46

Listendaten

2 Mit geordneten Daten arbeiten

Zahlen, Strings ... und Objekte	48
Die vier eingebauten Datentypen	50
Eine ungeordnete Datenstruktur: Dictionary	52
Eine Datenstruktur ohne Duplikate: Set	53
Literale Erzeugung von Listen	55
Wenn Sie mit mehr als ein paar Codezeilen arbeiten, sollten Sie den Editor benutzen	57
Eine Liste zur Laufzeit »wachsen lassen«	58
Zugehörigkeit mit »in« überprüfen	59
Objekte aus einer Liste entfernen	62
Eine Liste mit Objekten erweitern	64
Objekte in eine Liste einfügen	65
Datenstrukturen richtig kopieren	73
Listen erweitern die Schreibweise der eckigen Klammern	75
Listen verstehen start, stop und step	76
start- und stop-Werte für Listen	78
Listen mithilfe von Slices bearbeiten	80
Pythons »for«-Schleife versteht Listen	86
Marvins Slices im Detail	88
Wann man Listen nicht benutzen sollte	91
Der Code aus Kapitel 2, 1 von 2	92

Strukturierte Daten

3

Mit strukturierten Daten arbeiten

Ein Dictionary speichert Schlüssel/Wert-Paare	96
Dictionaries im Code erkennen	98
Reihenfolge des Einfügens wird NICHT beibehalten	99
Werte mithilfe eckiger Klammern nachschlagen	100
Zur Laufzeit mit Dictionaries arbeiten	101
Einen Frequenzzähler aktualisieren	105
Über ein Dictionary iterieren	107
Über Schlüssel und Werte iterieren	108
Mithilfe von »items« über ein Dictionary iterieren	110
Wie dynamisch sind Dictionaries wirklich?	114
KeyError-Laufzeitfehler vermeiden	116
Vorhandensein mit »in« überprüfen	117
Initialisierung vor Gebrauch sicherstellen	118
»in« durch »not in« ersetzen	119
Die »setdefault«-Methode verwenden	120
Sets effektiv erzeugen	124
Set-Methoden sinnvoll nutzen	125
Ein Plädoyer für Tupel	132
Eingebaute Datenstrukturen kombinieren	135
Auf Daten einer komplexen Datenstruktur zugreifen	141
Der Code aus Kapitel 3, Seite 1 von 2	143

4

Code wiederverwenden Funktionen und Module

Code mithilfe von Funktionen wiederverwenden	146
Einführung in Funktionen	147
Rufen Sie Ihre Funktionen auf	150
Funktionen können Argumente übernehmen	154
Einen Wert zurückgeben	158
Mehr als einen Wert zurückgeben	159
Eingebaute Datenstrukturen: Wiederholung	161
Eine allgemein nützliche Funktion erstellen	165
Eine neue Funktion erstellen, 1 von 3	166
Standardwerte für Argumente definieren	170
Positionelle und Schlüsselwortzuweisung im Vergleich	171
Aktualisierung unseres Wissens über Funktionen	172
Python auf der Kommandozeile ausführen	175
Die erforderlichen Setup-Dateien erstellen	179
Eine Distributionsdatei erstellen	180
Pakete mit »pip« installieren	182
Demonstration von Werteparametern	185
Demonstration von Referenzparametern	186
Die Entwicklerwerkzeuge zum Testen installieren	190
Wie PEP 8-konform ist unser Code?	191
Die Fehlermeldungen verstehen	192
Der Code aus Kapitel 4	194

5

Eine Webapplikation erstellen Auf ins wahre Leben!

Python: Was Sie bereits wissen	196
Was soll unsere Webapplikation können?	200
Flask installieren	202
Wie funktioniert Flask?	203
Die Flask-Web-App zum ersten Mal ausführen	204
Ein Flask-Webapplikationsobjekt erzeugen	206
Eine Funktion mit einer URL dekorieren	207
Das Verhalten Ihrer Webapplikation testen	208
Funktionalität im Web bereitstellen	209
Das HTML-Formular erstellen	213
Templates beziehen sich auf Webseiten	216
Templates mit Flask rendern	217
Das Formular der Webapplikation anzeigen	218
Vorbereitungen zum Ausführen des Template-Codes	219
HTTP-Statuscodes verstehen	222
Mit POST-Daten umgehen	223
Den Zyklus aus Bearbeiten, Anhalten, Starten und Testen optimieren	224
Mit Flask auf Formulardaten zugreifen	226
Die Formulardaten in der Webapplikation verwenden	227
Die Ergebnisse als HTML ausgeben	229
Die Webapplikation für die Cloud vorbereiten	238
Der Code aus Kapitel 5	241

6

Daten speichern und bearbeiten

Wo kommen die Daten hin?

Etwas mit den Daten Ihrer Webapplikation anstellen	244
Python unterstützt die Öffnen-Bearbeiten-Schließen-Technik	245
Daten aus einer bestehenden Datei lesen	246
Eine bessere Version von Öffnen-Bearbeiten-Schließen: »with«	248
Das Protokoll von der Webapplikation anzeigen lassen	254
Die Rohdaten per »Quelltext anzeigen« untersuchen	256
Es ist Zeit, Ihre Daten zu escapen	257
Die gesamte Log-Datei in der Webapplikation betrachten	258
Bestimmte Attribute des Web-Requests protokollieren	261
Eine Zeile voneinander getrennter Datenfelder protokollieren	262
Von Rohdaten zu lesbaren Ausgaben	265
Lesbare Ausgaben mit HTML erzeugen	274
Darstellungslogik in das Template integrieren	275
Mit Jinja2 lesbare Ausgaben erzeugen	276
Der aktuelle Status Ihres Webapplikationscodes	278
Die Daten befragen	279
Der Code aus Kapitel 6	280

Datenbanken benutzen

7 Die DB-API von Python verwenden

Die Webapplikation für die Benutzung von Datenbanken vorbereiten	282
Aufgabe 1: Den MySQL-Server installieren	283
Einführung in die DB-API von Python	284
Aufgabe 2: Einen MySQL-Datenbanktreiber für Python installieren	285
MySQL-Connector/Python installieren	286
Aufgabe 3: Die Datenbank und die nötigen Tabellen für die Webapplikation erstellen	287
Eine Struktur für Ihre Log-Daten definieren	288
Bestätigen Sie, dass die Tabelle für die Daten bereit ist	289
Aufgabe 4: Den Code für die Datenbank und die Tabellen unserer Webapplikation schreiben	296
Daten speichern ist die halbe Miete	300
Wie kann der Datenbankcode am besten wiederverwendet werden?	301
Überlegen Sie, was Sie hier wiederverwenden wollen	302
Und was ist mit import?	303
Sie kennen dieses Muster bereits	305
So schlecht sind die schlechten Nachrichten gar nicht	306
Der Code aus Kapitel 7	307

8

Ein bisschen Klasse

Verhalten und Zustand abstrahieren

Sich in die »with«-Anweisung einklinken	310
Kurze Einführung in Objektorientierung	311
Objekte aus Klassen erzeugen	312
Objekte übernehmen das Verhalten, aber nicht den Zustand	313
Mehr mit CountFromBy anfangen	314
Methodenaufrufe: Die Details verstehen	316
Methoden einer Klasse hinzufügen	318
Die Bedeutung von »self«	320
Die Gültigkeit von Geltungsbereichen	321
Stellen Sie Ihren Attributnamen »self« voran	322
(Attribut-)Werte vor Gebrauch initialisieren	323
__init__ initialisiert Attribute	324
Attribute mit »__init__« initialisieren	325
Die Darstellung von CountFromBy verstehen	328
Die Darstellung von CountFromBy selbst definieren	329
Sinnvolle Standardwerte für CountFromBy	330
Klassen: Was wir bereits wissen	332
Der Code aus Kapitel 8	333

Das Kontextmanagement-Protokoll

9

Sich in Pythons with-Anweisung einklinken

Wie können wir den Code unserer Webapplikation am besten mit anderen teilen?	336
Kontext anhand von Methoden verwalten	338
Sie kennen den Kontextmanager bereits	339
Eine neue Klasse für den Kontextmanager erstellen	340
Die Klasse mit der Datenbankkonfiguration initialisieren	341
Setup mit »__enter__«	343
Teardown mit »__exit__«	345
Den Code der Webapplikation überdenken, Teil 1 von 2	348
Die »log_request«-Funktion auf dem Prüfstand	350
Die »log_request«-Funktion anpassen	351
Die »view_the_log«-Funktion auf dem Prüfstand	352
Nicht nur der Code ändert sich	353
Die »view_the_log«-Funktion anpassen	354
Die Datenfragen beantworten	359
Der Code aus Kapitel 9, 1 von 2	360

10 Funktionsdekoren

Funktionen verpacken

Der Webserver (nicht Ihr Computer) führt den Code aus	366
Zustandsverwaltung mit Flasks Sessions	368
Den Zustand im Dictionary nachschlagen	369
Anmeldevorgänge mit Sessions verwalten	374
Log-out und Status überprüfen	377
Eine Funktion an eine Funktion übergeben	386
Eine übergebene Funktion aufrufen	387
Eine Liste mit Argumenten übernehmen	390
Eine Liste mit Argumenten verarbeiten	391
Ein Dictionary mit Argumenten übernehmen	392
Ein Dictionary mit Argumenten verarbeiten	393
Funktionsargumente von beliebiger Zahl und beliebigem Typ übernehmen	394
Einen Funktionsdekorator erstellen	397
Der letzte Schritt: Mit Argumenten umgehen	401
Der Dekorator im praktischen Einsatz	404
Zurück zur Zugangsbeschränkung für /viewlog	408
Der Code aus Kapitel 10, Teil 1 von 2	410

11

Mit Ausnahmen umgehen

Was zu tun ist, wenn mal etwas schiefgeht

Datenbanken sind nicht immer verfügbar	418
Angriffe aus dem Web können richtig nerven	419
Ein- und Ausgaben sind (manchmal) langsam	420
Funktionsaufrufe können fehlschlagen	421
Versuchen Sie immer, möglicherweise fehlerhaften Code auszuführen	423
Ein try, viele excepts	426
Ein Handler, sie zu knechten ...	428
Mit »sys« mehr über Ausnahmen erfahren	430
Noch mal: der »catch all«-Ausnahme-Handler	431
Zurück zum Code unserer Webapplikation	433
Ausnahmen leise handhaben	434
Mit anderen Datenbankfehlern umgehen	440
Vermeiden Sie eng verbundenen Code	442
Wiedersehen mit dem DBcm-Modul	443
Eigene Ausnahmen erstellen	444
Was kann mit »DBcm« noch schiefgehen?	448
Die Behandlung von <code>SQLError</code> funktioniert anders	451
Einen <code>SQLError</code> auslösen	453
Ein schneller Rückblick: Robustheit hinzufügen	455
Wie mit Wartezeiten umgehen? Kommt drauf an ...	456
Der Code aus Kapitel 11, 1 von 3	457

11 ³/₄

Ein bisschen Threading

Taten statt Warten

Warten: Was ist zu tun?	462
Wie fragen Sie Ihre Datenbank ab?	463
Datenbank-INSERTs und -SELECTs sind verschieden	464
Mehrere Dinge gleichzeitig tun	465
Keine Sorge. Benutzen Sie Threads	466
Das Wichtigste zuerst: keine Panik	470
Keine Sorge: Flask kann helfen	471
Ist Ihre Webapplikation jetzt robust?	474
Der Code aus Kapitel 11 ³ / ₄ , 1 von 2	475

12

Fortgeschrittene Iteration

Schleifen wie verrückt

CSV-Daten als Listen einlesen	479
CSV-Daten als Dictionaries einlesen	480
Rohdaten säubern und trennen	482
Vorsicht beim Verketteten von Methodenaufrufen	483
Daten in das benötigte Format umwandeln	484
Die Daten in ein Dictionary mit Listen umwandeln	485
Das Programmiermuster bei Listen erkennen	490
Programmiermuster in Comprehensions umwandeln	491
Ein genauerer Blick auf Comprehensions	492
Eine Dictionary-Comprehension definieren	494
Comprehensions mit Filtern erweitern	495
Pythons Weg für den Umgang mit Komplexität	499
Set-Comprehensions in Aktion	505
Und was ist mit »Tupel-Comprehensions«?	507
Runde Klammern um Code == Generator	508
URLs mit einer Listen-Comprehension verarbeiten	509
URLs mit einem Generator verarbeiten	510
Definieren Sie, was Ihre Funktion tun soll	512
Die Macht der Generatorfunktionen	513
Die Generatorfunktion aufspüren, Teil 1 von 2	514
Eine letzte Frage	518
Der Code aus Kapitel 12	519
Wir sind dann mal weg ...	520

Installation

A

Python installieren

Python 3 unter Windows installieren	522
Python 3 unter Windows testen	523
Python 3 unter Windows ergänzen	524
Python 3 unter Mac OS X (macOS) installieren	525
Python 3 unter macOS konfigurieren und testen	526
Python 3 unter Linux installieren	527

PythonAnywhere

B

Ihre Webapplikation bereitstellen

Schritt 0: Etwas Vorbereitung	530
Schritt 1: Bei PythonAnywhere registrieren	531
Schritt 2: Die Dateien in die Cloud hochladen	532
Schritt 3: Den Code extrahieren und installieren	533
Schritt 4: Eine Starter-Webapplikation erstellen, 1 von 2	534
Schritt 5: Die Webapplikation konfigurieren	536
Schritt 6: Drehen Sie eine Runde mit Ihrer cloudbasierten Webapplikation!	537



Die 10 wichtigsten Dinge, die wir nicht behandelt haben

Es gibt immer noch etwas zu lernen

1. Was ist mit Python 2?	540
2. Virtuelle Programmierumgebungen	541
3. Mehr zur Objektorientierung	542
4. Formate für Strings und Ähnliches	543
5. Dinge sortieren	544
6. Mehr zur Standardbibliothek	545
7. Code gleichzeitig ausführen	546
8. GUIs mit Tkinter (und Spaß mit turtle)	547
9. Ohne Test ist es nicht fertig	548
10. Debuggen, Debuggen, Debuggen	549



Die 10 wichtigsten Projekte, die wir nicht behandelt haben

Noch mehr Werkzeuge, Bibliotheken und Module

1. Alternativen zu >>>	552
2. Alternativen zu IDLE	553
3. Jupyter Notebook: die webbasierte IDE	554
4. Data Science betreiben	555
5. Technologien für die Webentwicklung	556
6. Mit Webdaten arbeiten	557
7. Noch mehr Datenquellen	558
8. Programmierwerkzeuge	559
9. Kivy: Unsere Wahl für das »coolste Projekt überhaupt«	560
10. Alternative Implementierungen	561

E Mitmachen **Die Python-Gemeinschaft**

Wohltuender Diktator auf Lebenszeit	564
Eine tolerante Gemeinschaft: Respekt für Vielfalt	565
Python-Podcasts	566
Das Python-Zen	567
Welches Buch sollte ich jetzt lesen?	569
Unsere englischen Lieblings-Python-Bücher	570
Unsere deutschen Lieblings-Python-Bücher	571