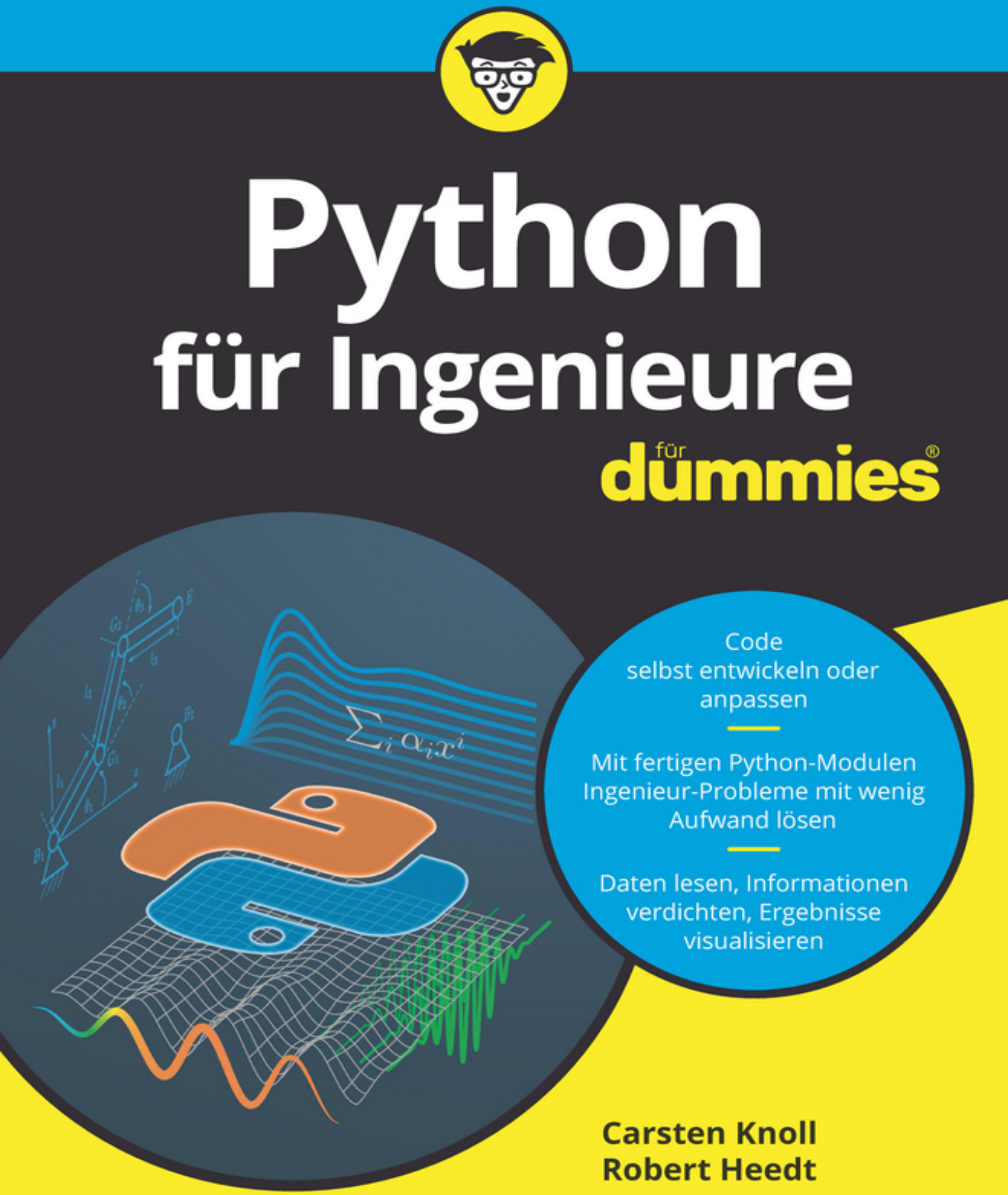
LERNEN EINFACH GEMACHT



Python für Ingenieure

für dummies®

Code selbst entwickeln oder anpassen

Mit fertigen Python-Modulen Ingenieur-Probleme mit wenig Aufwand lösen

Daten lesen, Informationen verdichten, Ergebnisse visualisieren

Carsten Knoll
Robert Heedt

Python für Ingenieure für Dummies

Schummelseite

DATENTYPEN

`360`	`int`
`104.5`	`float`
`1+2j`	`complex`
`True, False`	`bool`
`None`	`NoneType`
`[1, 2, 3]`	`list`
`(1, 2, 3)`	`tuple`
`{'a': 1, 'b': 2}`	`dict`
`'abc', "abc"`	`str`
`f"Var: {var}"`	f-String
`b"abc"`	`bytes`

KONTROLLSTRUKTUREN

`if cond:, elif cond:, else:`	If-Bedingung
`while cond:`	While-Schleife
`for element in list:`	Iteriere über Elemente
`for i in range(a, b):`	Iteriere über Zahlen
`for i, element in enumerate(list):`	Iteriere über Index und Inhalt

LISTENOPERATIONEN

`list[0]`	erstes Element
`list[-1]`	letztes Element
`list[0:10]`	Teilliste mit 10 Elementen
`list[0:2:10]`	Teilliste mit Schrittweite 2, 5 Elemente
`list[::-1]`	Liste umdrehen
`a.append(e)`	Element am Ende anfügen
`del list[0]`	erstes Element löschen
`len(a)`	Länge der Liste
`e in a`	Prüfen, ob Element in Liste enthalten
`b = list(a)`	Liste kopieren
`a + b`	Listen verknüpfen
`[e for e in list if cond]`	List-Comprehension

NUMPY-ARRAYS ANLEGEN

`np.array([10, 20, 30, 40])`	1D-Array aus einer Liste
`np.array([[10, 20], [30, 40]])`	2D-Array aus Liste von Listen
`np.arange(start, stop, step)`	Sequenz von Zahlen (Schrittweite vorgegeben)
`np.linspace(start, stop, N)`	Sequenz von `N` Zahlen (linear skaliert)
`np.logspace(start, stop, N)`	Sequenz von `N` Zahlen (logarithmisch skaliert)
`np.zeros(shape)`	Array voller Nullen mit Shape `shape`
`np.ones(shape)`	Array voller Einsen mit Shape `shape`
`np.random.random(shape)`	zufälliges Array mit Shape `shape`
`np.diag(x)`	quadratisches Array mit Diagonale `x`

NUMPY-ARRAYS INDIZIEREN

`x[0]`	»erstes« (vorderstes) Element von `x`
`x[-1]`	letztes Element von `x`
`x[:3]`	Elemente von `x` mit den Indizes 0, 1, 2
`x[3:]`	Elemente von `x` ab dem Index 3
`x[:-2]`	alle Elemente von `x` außer den beiden letzten
`x[-5:]`	die fünf letzten Elemente von `x`
`x[4:10:2]`	Elemente von `x` mit den Indizes 4, 6, 8
`x[:]`	alle Elemente von `x`
`x[::-1]`	alle Elemente von `x` in umgekehrter Reihenfolge
`y[0, :]`	erste Zeile (Index 0) von `y` als 1D-Array
`y[2:3, :]`	dritte Zeile (Index 2) von `y` als 2D-Array
`y[:, -4:]`	vier letzte Spalten von `y` als 2D-Array
`y[1:, [0, -1]]`	erste und letzte Spalte von `y`, aber ohne erste Zeile
`y[1:, [0, -1]]`	erste und letzte Spalte von `y`, aber ohne erste Zeile
`x[x>0.7]`	alle Werte von `x`, die größer als `0.7` sind

NUMPY-ARRAYS VERBINDEN

`np.concatenate((x1, x2))`	zwei Arrays hintereinanderhängen
`np.row_stack((x1, x2))`	zwei Arrays zeilenweise stapeln
`np.col_stack((x1, x2, x3))`	drei Arrays spaltenweise stapeln

NUMPY-ARRAYS UMFORMEN

`x.T`	`x` transponieren
`x.flatten()`	`x` auf 1D-Array reduzieren
`x.squeeze()`	überflüssige Dimensionen (mit Länge 1) von `x` entfernen

`x.reshape(3, 7)`	`x`-Elemente in 3 Zeilen und 7 Spalten anordnen
`x.reshape(3, -1)`	`x`-Elemente in 3 Zeilen mit passender Spaltenzahl anordnen

MIT *NUMPY*-ARRAYS RECHNEN

`3*x`	Array mit elementweisem Produkt aus 3 und jeweiligem `x`-Element
`4+x`	Array mit elementweiser Summe aus 4 und jeweiligem `x`-Element
`x1+x2`	Array mit elementweiser Summe von `x1` und `x2`
`x1*x2`	Array mit elementweisem Produkt von `x1` und `x2`
`x1@x2`	Skalarprodukt zwischen zwei 1D-Arrays gleicher Länge
`x@A`	interpretiert als: Zeilenvektor (1D-Array) mal Matrix (2D-Array)
`A@x`	interpretiert als: Matrix (2D-Array) mal Spaltenvektor (1D-Array)
`x**3`	Array mit 3. Potenz jedes `x`-Elements
`np.exp(x)`	Array mit Exponentialfunktion, angewendet auf jedes `x`-Element

Passende Dimensionen werden vorausgesetzt, sonst: `BroadcastingError`.

MINIBEISPIEL SIMULATION

```
from scipy.integrate\
        import solve_ivp

def ho_rhs(t, xx):
    # Harmonischer Oszillator
    x1, x2 = xx
    xdot1 = x2
    xdot2 = -x1
    return xdot1, xdot2

x0 = [1, 0]
res = solve_ivp(ho_rhs,
                (0, 30), x0)
plt.plot(res.t, res.y[0, :])
```

MINIBEISPIEL OPTIMIERUNG

```
from scipy.optimize\
        import minimize

def targetfunc(xx):
    # quadratische Zielfunktion
    x1, x2 = xx
    return (x1-3)**2+(x2+4)**2

x0 = [0, 0]
res = minimize(targetfunc, x0)
assert res.success
print(res.x)
```

MINIBEISPIEL FUNKTIONSAPPROXIMATION

```
from scipy.optimize\
        import curve_fit

xx = np.linspace(0, 10)
yy = -4*np.sin(1.5*xx + 2)

# Amplitude, Kreisfrequenz
# und Phase schätzen
def func(xx, a, omega, phi):
    return a*np.sin(omega*xx \
                    + phi)

res = curve_fit(func, xx, yy)
assert np.allclose(res[1], 0)
```

MINIMALBEISPIEL ANNAHMENPRÜFUNG

```
x = externer_funktionsaufruf()
assert x > 0
# oder:
if not x > 0:
    msg = "erwarte x > 0!"
    raise ValueError(msg)
```

MINIMALBEISPIEL UNITTEST

```
import unittest as ut

class Test1(ut.TestCase):
    def testcore(self):
        self.\
        assertEqual(1, 1)

ut.main()
```

KOMBINATORISCHE FUNKTIONEN

```
import itertools as it
# Reihenfolge: ja, Wiederholung: ja
it.product([0, 1], repeat=2)
# -> [(0, 0), (0, 1), (1, 0), (1, 1)]
# Reihenfolge: ja, Wiederholung: nein
it.permutations([2, 3])
# -> [(2, 3), (3, 2)]
# Reihenfolge: nein, Wiederholung: nein
it.combinations([4, 5, 6], 2)
# -> [(4, 5), (4, 6), (5, 6)]
# Reihenfolge: nein, Wiederholung: ja
it.combinations_with_replacement([7, 8], 2)
# -> [(7, 7), (7, 8), (8, 8)]
```

FOURIER-TRANSFORMATION

`np.fft.fft(xx)`	diskrete Fourier-Transformation
`np.fft.ifft(XX)`	inverse DFT
`np.fft.fftfreq(N, d=dt)`	passende Stützfrequenzen
`np.fft.fftshift(ff)`	Nullfrequenz verschieben
`scipy.signal.periodogram(xx, fs=fs)`	Leistungsdichtespektrum berechnen

`scipy.signal.welch(xx, fs=fs)`	Welch-Methode

PLOTTEN: GRUNDLAGEN

`plt.figure()`	neue Abbildung
`plt.subplot(234)`	Subplot unten links für 2 Zeilen und 3 Spalten
`plt.plot(xx, yy)`	Linienplot zeichnen
`plt.tight_layout()`	Elemente platzsparend ausrichten
`plt.savefig("my_plot.pdf")`	Diagramm abspeichern
`plt.show()`	Diagramm anzeigen

PLOTTEN: DEKORATIONEN

`plt.xlim(-1, 1)`	Definitionsbereich einstellen
`plt.title("Überschrift")`	Diagrammüberschrift
`plt.xlabel("$t$ [s]")`	Achsenbeschriftung
`plt.text(1, 3, "$x_0$")`	Text hinzufügen
`plt.grid(True)`	Gitter aktivieren
`plt.legend()`	Legende basierend auf `label`-Parametern

PLOTTEN: STYLING

`figsize=(4, 3)`	Diagrammgröße 4 × 3 Zoll (1 Zoll = 22.4 mm)
`color="red"`	Linienfarbe
`linestyle="--"`	gestrichelte Linie
`linewidth=3`	Linienbreite 3 pt
`marker="x"`	Marker für Stützpunkte

`markersize=4`	Markergröße
`label="$x_1$"`	Linienbeschriftung für Legende

Carsten Knoll
Robert Heedt

Python für Ingenieure

für dummies®

Fachkorrektur von Thomas Smits

WILEY
WILEY-VCH GmbH

Python für Ingenieure für Dummies

Bibliografische Informationder Deutschen Nationalbibliothek

Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie;detaillierte bibliografische Daten sind im Internet über http://dnb.d-nb.de abrufbar.

Print ISBN: 978-3-527-71767-5
ePub ISBN: 978-3-527-82846-3

Coverillustration: Carsten Knoll – https://python-fuer-ingenieure.de
Korrektur: Matthias Delbrück, Dossenheim/Bergstraße

Über die Autoren

Carsten Knoll studierte an der TU Dresden Mechatronik und ist dort inzwischen als Postdoktorand am Institut für Regelungs- und Steuerungstheorie tätig. Seine ersten Erfahrungen mit Python machte er bereits im ersten Semester in einem inspirierenden Workshop und vertiefte sie beständig durch den Einsatz in diversen Lehrveranstaltungen sowie Tätigkeiten in verschiedenen Instituten und Laboren. Um seine Begeisterung zu teilen und dieses Wissen weiterzugeben, initiierte er 2010 zusammen mit anderen Doktoranden eine fächerübergreifende Lehrveranstaltung und entwickelte sie seitdem inhaltlich und methodisch zum aktuellen »Pythonkurs für Ingenieur:innen« weiter.

Neben Regelungstechnik und Python-Programmierung befasst er sich unter anderem mit Aspekten des Maschinellen Lernens und der digitalen Wissensrepräsentation – sowie mit der Frage, wie diese Ansätze zur Lösung der großen Herausforderungen des 21. Jahrhunderts beitragen können.

Robert Heedt ist ebenfalls Absolvent der TU Dresden und Ingenieur der Elektrotechnik. Sein Arbeitsgebiet ist die Regelungstechnik, aus dem die Rechentechnik schon lange nicht mehr wegzudenken ist. Nach jahrelanger Odyssee durch die Welt der Programmiersprachen ist er jetzt im Python-Hafen gelandet.

Carstens Widmung

Für Maria und meine Eltern. Ohne Euch hätte ich das nicht geschafft.

Roberts Widmung

Für meine Eltern. Danke für Eure bedingungslose Unterstützung.

Danksagung

Die Autoren bedanken sich aufs Herzlichste bei Frau Andrea Baulig vom Verlag Wiley-VCH für den Anstoß zu diesem Projekt, die sehr konstruktive Kombination aus Motivation und Ermahnung sowie die Übernahme organisatorischer Details. Herrn Prof. Smits danken wir ebenfalls sehr herzlich für die sehr professionelle fachliche Korrektur des Manuskripts und die wertvollen Hinweise (alle verbliebenen Schwächen des Buches gehen natürlich auf unsere Kappe).

Wir danken auch unseren Kolleginnen und Kollegen sowie den Studierenden am Institut für Regelungs- und Steuerungstheorie, die sowohl mit Tipps als auch mit Fragen und Diskussionen eine beständige Inspirationsquelle für dieses Buch waren.

Nicht zuletzt gilt unser Dank den vielen Menschen, die durch Fähigkeiten, Fleiß und Enthusiasmus die Python-Welt zu dem gemacht haben, was sie ist.

Inhaltsverzeichnis

- Cover
- Titelblatt
- Impressum
- Über die Autoren
- Einleitung
  - Über dieses Buch
  - Törichte Annahmen über den Leser (m/w/d)
  - Ein Wort zur Sprache
  - Was Sie nicht lesen müssen
  - Wie dieses Buch aufgebaut ist
  - Symbole, die in diesem Buch verwendet werden
  - Wie es weiter geht
- Teil I: Los geht's
  - Kapitel 1: Erstkontakt
    - Was ist Python und wozu ist es gut?
    - Jupyter-Notebook im Web
    - Besser als Ihr »Casio«
    - Erstes Date mit NumPy-Arrays
    - Bunte Bilder mit Matplotlib
    - Zusammenfassung
  - Kapitel 2: Installation und Inbetriebnahme
    - Python installieren
    - Alternativen zum Texteditor
    - Zusammenfassung
  - Kapitel 3: Grundlagen
    - Elementare Datentypen und Operationen
    - Listen, Strings und andere Sequenzen

Verzweigungen und Schleifen

List Comprehensions – Listen kompakt bauen und filtern

Funktionen

Pakete, Module und Namensräume

Fehlermeldungen verstehen und lieben lernen

Zusammenfassung

Teil II: Rechnen und Plotten

Kapitel 4: Effiziente Numerik mit NumPy

Arrays – Felder und ihre Bewirtschaftung

Array-Typen, elementweise Vergleiche und Indizierung von Arrays mit Arrays

Felder mit dem Mähdrescher abernten: Effiziente NumPy-Funktionen

Arrays, Matrizen – Wo ist denn da der Unterschied?

Zusammenfassung

Kapitel 5: Ein bisschen Mathe – So viel Spaß muss sein

Komplexe Zahlen – Nur manchmal reell, aber immer fantastisch

Lineare Gleichungssysteme – Von trivial bis unlösbar

Eigenwerte und Singulärwerte – Matrizen auf das Wesentliche reduziert

Zusammenfassung

Kapitel 6: Brunftzeit für Termhirsche

Aller Anfang ist sympel

Gleichungen lösen

Agent Smith in meiner Formel? Matrix und Vektor

Differenzial- und Integralrechnung

Symbolische Ergebnisse weiterverwenden

Was sonst noch geht

Zusammenfassung

Kapitel 7: Visualisierung

Diagramme zeichnen für Anfänger: Grundlagen

Gestochen scharf bis zum Schluss: Sauberer Export
Eine Schicht tiefer: Objektorientierung
Im Bilderzoo: Mehr Diagrammtypen
Zeit für Bewegung: Animationen
Zusammenfassung

Teil III: Fortgeschrittene Ingenieursmethoden

Kapitel 8: So tun, als ob: Modellbildung und Simulation

Beschreibung dynamischer Systeme: Ich krieg' Zustände
Dynamische Systeme lösen
Was beim Simulieren schiefgehen kann
Ruhelagen, (In-)Stabilität, exponentielles Wachstum und Eigenwerte
Wunschverhalten erzeugen: Etwas Regelungstechnik mit Tempomat
Zusammenfassung

Kapitel 9: Optimierung – Besser geht's nicht

Numerische Optimierung: Wie es grundlegend funktioniert
Nicht-lineare Gleichungen lösen durch Minimierung des Fehlers
Lokale Minima und n Dimensionen
Optimale Steuerung
Optimierung mit Begrenzungen …
… und Nebenbedingungen
Fit for Fun: Funktionsapproximation
Zusammenfassung

Kapitel 10: Mechanik – Ganz ohne schmutzige Hände

Statik und Festigkeitslehre – Bis sich die Balken biegen
Kinematik – Zwangsläufig gut
Kinetik: Volle Kraft voraus
Zusammenfassung

Kapitel 11: Fourier-Analyse – Python in der Disco

- Die Fourier-Transformation der kontinuierlichen Welt
- Diskrete Fourier-Transformation auf dem Papier
- Die DFT in Python
- Fourier-Analyse
- Zusammenfassung

Kapitel 12: Kombinatorik, Zufall und Statistik

- Kombinatorik – So viele Möglichkeiten
- Pseudozufall – Abzählreime 2.0
- Histogramme und Verteilungen
- Der Mittelwert und seine Freunde
- Statistische Unsicherheit visuell darstellen
- Zusammenfassung

Kapitel 13: Python im Labor – Steuern, Messen, Filtern, Darstellen

- Einordnung: In welchen Situationen ist Python sinnvoll?
- Schnittstellen und Busse
- Netzwerksockets
- Schnittstellen über kompilierte Treiber ansprechen
- Allgemeine Hinweise zur Laborautomatisierung
- Rauschen und wie man damit umgeht
- Zusammenfassung

Teil IV: Schöner, schneller, robuster

Kapitel 14: Echt Klasse: Fortgeschrittene Programmierung

- Objektorientierung für Normalsterbliche
- Funktionen, die Funktionen bauen
- Exceptions für sich gewinnen
- Zusammenfassung

Kapitel 15: Profiling und Performanz-Optimierung

- Wärmeleitung mit Python simulieren
- Wie schlimm ist es wirklich? Performanz messen
- NumPy seine Arbeit machen lassen
- Fremdsprachen willkommen – Das »Foreign Function Interface«
- Gerade noch rechtzeitig – Just-in-Time-Kompilierung
- Funktionen im Profil betrachtet
- Zusammenfassung

Kapitel 16: Von den Profis lernen – Das Wichtigste zur Softwaretechnik

- Seismografen im Kartenhaus: Assertions und Unittests
- Ghostbusters: Interaktives Fehlersuchen
- Quellcode mit Stil: PEP8
- Ordnung und Sicherheit: Versionskontrolle mit Git
- Zusammenfassung

Teil V: Der Top-Ten-Teil

Kapitel 17: Die 10 nützlichsten Module, die bisher nicht erwähnt wurden

- Kommandozeilenargumente auswerten
- Attraktive Benutzeroberflächen
- Objekte inspizieren
- Mit Daten und Uhrzeiten rechnen
- Parallelisierung in mehreren Prozessen
- Python-Objekte als Dateien abspeichern
- Reguläre Ausdrücke
- Typenannotationen
- Datenauswertung mit pandas
- Maschinelles Lernen mit scikit-learn

Kapitel 18: 10 fiese Fallstricke und 3 Mal schwarze Python-Magie

- Code an der falschen Stelle bearbeitet oder installiert

Syntaxfehler durch falsche Einrückung
Python 2.x: Semikompatibler Gespenster-Code
Verwirrung mit globalen und lokalen Variablen
Wiederholter import bleibt wirkungslos
lambdify und Arrays
Konversion von SymPy-Matrizen zu NumPy-Arrays
Der Rückgabetyp von sympy.solve
Versehentliches Broadcasting
Backslashes in Strings
Tricksereien mit Indizes
Fallunterscheidungen in Zuweisung und als Index
Für Furchtlose: »Spezielle« Module

Kapitel 19: 10 nützliche Links

docs.python.org und pypi.org
pythontips.com
github.com/search
pythex.org
computers-are-fast.github.io
stackoverflow.com
sphinx-doc.org
hackerrank.com/domains/python
cscircles.cemc.uwaterloo.ca
planet.scipy.org

Stichwortverzeichnis

End User License Agreement

Tabellenverzeichnis

Kapitel 12

Tabelle 12.1: Übersicht über kombinatorische Funktionen.

Illustrationsverzeichnis

Kapitel 1

Abbildung 1.1: Oberfläche eines *Jupyter*-Notebooks....

Abbildung 1.2: Zeitverlauf von Strom und Spannung.

Abbildung 1.3: Messdaten und Näherungsgerade im U-von-I-Diagramm.

Kapitel 2

Abbildung 2.1: Bedienoberfläche der IDE Spyder.

Abbildung 2.2: Konfigurationsfenster für Python-Interpreter in Spyder.

Abbildung 2.3: Bedienoberfläche der IDE PyCharm.

Abbildung 2.4: Konfigurieren des Interpreters beim Anlegen eines neuen Projekts...

Kapitel 5

Abbildung 5.1: Reihenschaltung aus Ohmschem Widerstand R und ...

Abbildung 5.2: Ortskurve für das RL-Glied.

Abbildung 5.3: Das Bode-Diagramm für das RL-Glied.

Abbildung 5.4: Reihenschaltung aus zwei Widerständen $R_1(T)$...

Abbildung 5.5: Fließschema einer (imaginären) verfahrenstechnischen Anlage mit ...

Kapitel 6

Abbildung 6.1: Verschiedene *SymPy*-Symbole und wie sie in *Jupyter*-Notebooks...

Kapitel 7

Abbildung 7.1: *Matplotlib*-Diagramme in unterschiedlichen Umgebungen.

Abbildung 7.2: Verschiedene Darstellungen einer Sinusfunktion.

Abbildung 7.3: Verschiedene Einstellungen in einem Diagramm kombiniert.

Abbildung 7.4: Ergebnisse der beiden Beispiellayouts.

Abbildung 7.5: Bode-Diagramm eines harmonischen Oszillators mit Markierung der Durchtrittsfrequenz...

Abbildung 7.6: Vergleich korrekter und inkorrekter Größeneinstellung.

Abbildung 7.7: Die Begriffswelt der Elemente einer Abbildung.

Abbildung 7.8: Beispiele für Balkendiagramme.

Abbildung 7.9: Beispiele für Darstellungen von 2D-Arrays.

Abbildung 7.10: Temperaturmessung, mit Wärmebildkamera aufgenommen.

Abbildung 7.11: Beispiele für Konturdiagramme.

Abbildung 7.12: Kurven in 3D.

Abbildung 7.13: Dreidimensionale Flächen.

Abbildung 7.14: Drahtgitterdarstellung eines Torus.

Kapitel 8

Abbildung 8.1: Ein einfaches Pendel mit masselosem Stab als Beispiel für ein...

Abbildung 8.2: Simulationsergebnisse für das Pendel mit dem Euler-Solver.

Abbildung 8.3: Simulationsergebnisse für das gedämpfte Pendel entsprechend...

Abbildung 8.4: Simulationsergebnisse für das »angeregte« Pendel mit...

Abbildung 8.5: Ein Standardregelkreis mit den folgenden zeitabhängigen...

Abbildung 8.6: Tempomat mit Proportionalregler...

Abbildung 8.7: Tempomat mit Proportionalregler...

Abbildung 8.8: Tempomat mit PI-Regler ...

Abbildung 8.9: Geschwindigkeitsverlauf für den Tempomat mit PI-Regler bei einem...

Kapitel 9

Abbildung 9.1: Photovoltaikmodul mit angeschlossenem elektrischen Verbraucher.

Abbildung 9.2: Links: Strom-Spannungs-Kennlinie $I(U)$ eines...

Abbildung 9.3: Testfunktion nach Gleichung (9.3) mit vielen lokalen Minima...

Abbildung 9.4: 2D-Zielfunktion 9.4. Links: 3D-Darstellung (Funktionswert als...

Abbildung 9.5: Anwendung von `minimize` auf Zielfunktion...

Abbildung 9.6: Wagen, auf den eine beschleunigende Kraft wirkt.

Abbildung 9.7: Ergebnis der Optimalsteuerungsaufgabe. Links: Eingangsverlauf....

Abbildung 9.8: Vergleich der optimalen Eingangsfolge $a_1, \ldots, a_N$ ...

Abbildung 9.9: Vergleich der Optimierung für die Fälle »ohne Nebenbedingungen«,...

Abbildung 9.10: Strommessung durch einen Ohmschen Widerstand. Links: Schaltplan. Rechts: Messwerte.

Abbildung 9.11: Lineare Approximation der fehlerbehafteten Strommesswerte.

Abbildung 9.12: Quadratische Approximation der Leistung (auf Basis der fehlerbehafteten Strommesswerte).

Abbildung 9.13: »Überanpassung (*overfitting*)« eines parametrischen Ansatzes...

Abbildung 9.14: Spline-Interpolation (für $s = 0$) beziehungsweise Regression ($s > 0$)...

Kapitel 10

Abbildung 10.1: Ein an zwei Punkten gelagerter Balken mit einer schräg angreifenden...

Abbildung 10.2: Ein eingespannter Balken mit einer Streckenlast $q(x)$.

Abbildung 10.3: Biegelinie des eingespannten Balkens unter konstanter Streckenlast.

Abbildung 10.4: Offene kinematische Kette des zu modellierenden Viergelenkgetriebes (schematische Skizze).

Abbildung 10.5: Geschlossene kinematische Kette für...

Abbildung 10.6: Interaktives Bedienelement (Slider-Widget) und »Zwiebelschalengrafik der...

Abbildung 10.7: Koppelkurve des Punktes K (starr mit dem Koppelglied verbunden).

Abbildung 10.8: Schematische Darstellung eines Doppelpendels unter Einfluss der Gravitation.

Abbildung 10.9: Simulationsergebnis für das Doppelpendel. Verlauf der Winkel für den Anfangszustand...

Abbildung 10.10: Verlauf der Gesamtenergie für das Doppelpendel. Die waagerechte Linie zeigt, dass...

Kapitel 11

Abbildung 11.1: Überlagerung eines konstanten Signals mit zwei Schwingungen bei 2 Hz und 3 Hz.

Abbildung 11.2: Frequenzanteile des Beispielsignals.

Abbildung 11.3: Zusammenhang zwischen einem kontinuierlichen Signal und seinen zeitdiskretisierten Abtastpunkten.

Abbildung 11.4: Amplituden- und Phasenspektrum des Testsignals, angenähert durch eine DFT.

Abbildung 11.5: Signalpegel der Beispiel-Audioaufnahme in Abhängigkeit von der Zeit. Die y-Achse...

Abbildung 11.6: Leistungsdichtespektrum des Audiosignals in linearer und logarithmischer Darstellung.

Abbildung 11.7: Periodisch fortgesetzter Ausschnitt eines Sinussignals, ohne und mit Fenster.

Abbildung 11.8: Gegenüberstellung verschiedener Fensterfunktionen. Links: Form des Fensters...

Abbildung 11.9: Anordnung der überlappten Fenster für Welch-Methode

Abbildung 11.10: Leistungsdichtespektrum nach der Methode von Welch mit 2048 Abtastwerten pro Segment.

Abbildung 11.11: Leistungsdichtespektrum nach der Methode von Welch, aber mit nur 512 Abtastwerten pro Segment....

Abbildung 11.12: Spektrogramm des Audiobeispiels

Kapitel 12

Abbildung 12.1: Histogramm von 3000 (per Zufallsgenerator simulierten) Würfelwürfen...

Abbildung 12.2: Histogramm der Häufigkeit k einer »Sechs« bei...

Abbildung 12.3: Histogramm von 2000 normalverteilten Temperaturmesswerten. Außerdem: Wahrscheinlichkeitsdichte...

Abbildung 12.4: Exponentialverteilung mit $\lambda = 5\ \mathrm{h}^{-1}$ ....

Abbildung 12.5: Histogramm von 10 000 Werten der Exponentialverteilung mit $\lambda = 5\ \mathrm{h}^{-1}$ ....

Abbildung 12.6: Messwerte der Windgeschwindigkeiten in Abhängigkeit von der Tageszeit...

Abbildung 12.7: Kastendiagramm (Boxplot) der Windgeschwindigkeitsmesswerte.

Abbildung 12.8: Violinendiagramm der Windgeschwindigkeitsmesswerte. Aus Platzgründen...

Kapitel 13

Abbildung 13.1: ~~Veralteter~~ Klassischer 9-poliger D-Sub-Stecker einer...

Abbildung 13.2: GPIB-Stecker (Draufsicht).

Abbildung 13.3: Netzwerkkabel mit einem RJ45-Stecker.

Abbildung 13.4: Um zwei Achsen schwenkbarer USB-Raketenwerfer (Bildquelle: getDigital.de).

Abbildung 13.5: Links: Verrauschtes Spannungssignal (zeitabhängig). Rechts: Zugehöriges...

Abbildung 13.6: Links: Verrauschtes Spannungssignal im Zeitbereich (originale und interpolierte Daten)...

Abbildung 13.7: Zirp-Signal – ungestört und mit Rauschsignal überlagert.

Abbildung 13.8: Anwendung zweier verschiedener Filter erster Ordnung auf das Testsignal...

Abbildung 13.9: Butterworth-Filter 5. Ordnung mit $f_{\mathrm{Grenz}} = 50$ ...

Kapitel 15

Abbildung 15.1: Wirkung der Wärmeleitungsgleichung in einer Dimension.

Abbildung 15.2: Ergebnis der Simulation der Wärmeleitungsgleichung für die im...

Einleitung

»Dem Ingenieur ist nichts zu schwör!« – Diese Herangehensweise hat über viele Jahre mehr oder weniger das Bild von Ingenieurinnen und Ingenieuren geprägt. Seit der Entstehungszeit dieses Credos in der Ära der Reißbretter und Tabellenbücher haben sich die Ingenieurwissenschaften aber erheblich weiterentwickelt. Digitale Werkzeuge bestimmen mehr denn je den Ingenieuralltag. Sie ermöglichen es, in kurzer Zeit Probleme zu lösen, die noch vor einer Generation unlösbar schienen. Mit diesen gestiegenen (und permanent weiter steigenden) Möglichkeiten wachsen aber auch die Erwartungshaltung und die Anforderung an Ingenieur:innen.

Der souveräne Umgang mit digitalen Werkzeugen ist kein Selbstläufer, sondern muss mit Mühe und Aufwand erlernt werden. Um so ärgerlicher ist es, wenn die hart erarbeiteten Fähigkeiten aus der Mode kommen oder sich aufgrund von Lizenzbeschränkungen (zum Beispiel beim Wechsel aus der Hochschule in ein Startup) nicht anwenden lassen. Die Programmiersprache Python und das darauf basierende Ökosystem aus Zusatzpaketen bieten vor diesem Hintergrund eine exzellente Wahl. Einerseits steigt die Popularität von Python seit den 1990ern moderat, aber beständig – das genaue Gegenteil von sonst häufigen Hype-Zyklen. Zum anderen sind alle relevanten Komponenten freie Software und plattformunabhängig – mehr Freiheit geht nicht.

Leider ist die Ingenieur-Ausbildung teilweise erstaunlich ~~rückständig~~ konservativ, weswegen das Lösen von Ingenieur-Problemen mithilfe digitaler Werkzeuge bisher kaum als Lehrveranstaltung angeboten wird – oder wenn, dann oft als kommerziell motivierte externe

»Produktschulung« mit den oben angedeuteten Lizenz- und Abhängigkeitsproblemen. Das vorliegende Buch soll dem Abhilfe schaffen.

Über dieses Buch

Dieses Buch soll Sie befähigen, vielfältige Aufgabenstellungen mit Python zu lösen. Es vermittelt dabei sowohl das grundsätzliche Herangehen als auch ganz konkrete Tipps und Tricks. Um dieses Wissen möglichst interessant, nachvollziehbar und anwendungsfreundlich darzustellen, werden Sie in diesem Buch auf eine Vielzahl von Beispielen treffen. Manche nur für einen dreizeiligen Codeschnipsel, andere für ein komplexeres Szenario, das sich einer praxisrelevanten Fragestellung widmet. Dabei finden Sie einen thematischen Streifzug durch viele Ingenieurdisziplinen: Von der Biegelinie eines Balkens über optimale Mischverhältnisse in der Verfahrenstechnik und die $j\omega$-Wechselstromrechnung bis zur Fourier-Analyse und Simulation eines Regelkreises im Zustandsraum ist (fast) alles dabei, was das Herz begehrt.

Zum Ingenieuralltag gehört auch der Umgang mit Zielkonflikten: Ein Fahrradrahmen kann zum Beispiel entweder extrem leicht oder extrem stabil sein. Am sinnvollsten ist aber ein auf typische Anwendungsfälle zugeschnittener Kompromiss. So ähnlich ist es auch bei diesem Buch: Die behandelten Code-Beispiele könnten entweder sehr einfach gestrickt sein, wären aber dafür weit weg von jedem Anwendungsbezug. Das andere Extrem wäre, an jedes Beispiel einen harten Realismusanspruch zu stellen und dabei die Verständlichkeit außer Acht zu lassen.

Ziel des Buches ist es, für eine möglichst große Zielgruppe möglichst oft den optimalen Bereich zwischen langweiliger Trivialität und demotivierender Komplexität zu treffen.

Törichte Annahmen über den Leser (m/w/d)

Diese Buch richtet sich an Menschen, die sich für Technik interessieren, wie sie funktioniert und wie man sie verbessern kann. Wir vermuten, dass sich dieses Interesse in einer gewissen Vorbildung niedergeschlagen hat und wollen unsere Zielgruppe weder langweilen noch überfordern. Idealerweise haben Sie

- ✔ vier Semester oder mehr eines Ingenieurstudiengangs auf dem Buckel,
- ✔ keine Angst vor gelegentlichen Gleichungen,
- ✔ schon ein bisschen Programmiererfahrung (Variablen, Typen, Verzweigungen, Schleifen, ...),
- ✔ einen Computer, dem üblicherweise Sie Anweisungen geben (und nicht umgekehrt),
- ✔ »Mut zur Lücke« – also die Bereitschaft, auch mal etwas grob zu überfliegen, was für Sie aktuell gerade nicht so relevant ist, und
- ✔ Sinn für Humor sowie die Bereitschaft, Fehler zu machen, Fehler zu suchen und daraus zu lernen.

Ein Wort zur Sprache

Die Entstehung des Buches fällt in eine Zeit, in der mitunter sehr aufgeregt emotional über das Für und Wider verschiedener Formulierungen gestritten wird.

Einige potenzielle Leser:innen werden das Buch vielleicht hochmütig ignoriert haben, weil der grammatikalisch maskuline Titel »für Ingenieure« ihrem Anspruch an geschlechtsneutrale Sprache zuwiderläuft. Andere werden beim ersten Auftreten von Syntaxinnovationen wie »Ingenieur:innen« einen mittleren Empörungsanfall bekommen und eine rituelle Verbrennung des Buches in Erwägung ziehen. Wir vertrauen jedoch darauf, dass die große Mehrzahl der Leserinnen und Leser ihren gesunden Menschenverstand einsetzt und weder der einen noch der anderen Überreaktion zum Opfer fällt. Selbstverständlich sind mit dem Titel und anderen maskulinen oder femininen Formulierungen grundsätzlich alle angesprochen, unabhängig davon, welchem Geschlecht sie sich zuordnen. Es sollte aber nicht überraschen, dass uns als Autoren eines Buches voller formaler Logik und wichtiger (programmier-)sprachlicher Detailnuancen das Konzept des »generischen Maskulinums«, wonach »der Ingenieur« ein neutraler Oberbegriff von »der Ingenieur« und »die Ingenieurin« sein soll, nicht so 100 % überzeugt. Andererseits haben wir auch (noch) keine ultimative Lösung und behelfen uns daher mit einem Mix aus klassischem und innovativem Sprachgebrauch. Ein ähnlicher Pragmatismus gilt in diesem Buch übrigens auch für den Umgang mit Anglizismen.

Was Sie nicht lesen müssen

Nicht alles in diesem Buch ist für alle Leser:innen gleich wichtig. Grundsätzlich gilt, dass der Inhalt weiter vorn von eher allgemeinem Interesse ist und nach hinten immer spezieller wird. Andererseits kann es natürlich

sein, dass Sie schon ein paar Erfahrungen mit Python haben und die ersten Kapitel deshalb nicht benötigen. Grundsätzlich gilt: Lesen Sie nur das, was Sie interessiert! Wenn Sie in einem der hinteren Kapitel ein Sprachkonstrukt oder einen Funktionsaufruf nicht verstehen, lohnt sich aber ein Blick in die vorderen (und ins Stichwortverzeichnis). Die Chance ist groß, dort Aufklärung zu finden. Wenn Sie inhaltlich mal bei einem Beispiel aussteigen: Machen Sie sich nichts draus. Die Ingenieurwissenschaften sind ein weites Feld und niemand kann alles.

Eine erfolgreiche Herangehensweise aus der Informatik ist es, in Schnittstellen zu denken. Sie müssen nicht wissen, wie ein Magnetron funktioniert (oder was das ist), um einen Mikrowellenherd zu benutzen. Es reicht, die Schnittstellen zu verstehen (Timer, Türknopf, »Pling«). Genauso ist es mit Code. Sie müssen nicht immer im Detail verstehen, wie eine Funktion oder ein Code-Abschnitt genau arbeitet. Oft reicht es zu verstehen, was die Eingangsdaten sind und was das Ergebnis ist oder bewirkt. Konzentrieren Sie Ihre Aufmerksamkeit auf die Bereiche, die aktuell relevant für Sie sind.

Wie dieses Buch aufgebaut ist

Das Buch ist in folgende Teile untergliedert:

Teil I: Los geht's

Hier setzen wir den Zug aufs Gleis und rollen behutsam los. Den Anfang macht das (möglicherweise) allererste Ausführen eines Python-Programms – schön bequem online im Browser. Es folgen Installation und