Leseprobe

Hans-Bernhard Woyand

Python für Ingenieure und Naturwissenschaftler

Einführung in die Programmierung, mathematische Anwendungen und Visualisierungen

ISBN (Buch): 978-3-446-45198-8
ISBN (E-Book): 978-3-446-45300-5

Weitere Informationen oder Bestellungen unter

http://www.hanser-fachbuch.de/978-3-446-45198-8

sowie im Buchhandel.

© Carl Hanser Verlag, München
Vorwort


Im Studiengang Maschinenbau wie auch in vielen anderen Studiengängen spielt das wissenschaftliche Rechnen und Visualisieren eine große Rolle. In der Literatur zu Python ist das jedoch nicht so: entweder werden die wissenschaftlichen Anwendungen gar nicht bzw. nur am Rande behandelt, oder die Bücher sind so umfangreich, dass sie als vorlesungsbegleitender Text ungeeignet sind. So entstand die Idee, das Skript zur Vorlesung zu diesem Buch auszuarbeiten.

Zielgruppe
Das Buch richtet sich an Programmieranfänger, die Python als erste Programmiersprache lernen möchten. Es wird also zuerst in Python eingeführt und dann werden die Grundlagen erläutert. Es folgen Vertiefungen und ein eigenes Kapitel über die objektorientierte Programmierung. Dann beginnt das wissenschaftliche Rechnen und Visualisieren. Hierzu wird in die Nutzung der wichtigsten Programmbibliotheken (Packages) eingeführt. Hauptzielgruppe sind also Studierende wissenschaftlicher Studiengänge, die Python erlernen wollen und bei denen die mathematischen und graphische Anwendungen eine wichtige Rolle spielen. Vorausgesetzt wird nur mathematisch-naturwissenschaftliches Wissen, wie es an höheren Schulen vermittelt wird.

Aufgabenorientierte Lehre
Mehr als 90 Aufgabenstellungen mit fast immer kommentierten Lösungen werden im Buch behandelt. Nach meiner Erfahrung lernen Studierende am meisten durch das selbstständige Lösen von Aufgaben.

Software
Die Software, die benötigt wird, um mit diesem Buch zu arbeiten, ist kostenfrei erhältlich. Es handelt sich dabei um die neuere Variante 3.2 bzw. 3.5 der Programmiersprache Python, sowie deren Vorgängerversionen 2.6 und 2.7. Die wissenschaftlichen Anwendungspakete sind nämlich noch nicht alle mit der neuen Python-Version kompatibel. Es ist zu erwarten, dass es auch noch einige Zeit dauern wird, bis die wissenschaftlichen Pakete unter der neuesten Version laufen. Für Programmieranfänger ist der Unterschied zwischen diesen Versionen sowieso nicht sehr bedeutsam.

Internet-Seite zum Buch
Zu dem vorliegenden Buch existiert eine Internet-Seite, auf der in loser Folge Ergänzungen, Fehlerberichtigungen usw. bereitgestellt werden. Weiterhin können alle Beispiele sowie die Lösungen der Aufgaben dort abgerufen werden. Für Nutzer des Betriebssystems MS-Windows sind auch Hinweise zur Installation der Software dort verfügbar. Die Web-Adresse ist
http://woyand.eu/python_buch

Falls sich diese Adresse einmal ändern sollte, so kann der Leser den neuen Standort über die Internet-Seiten des Verlages

http://www.hanser-fachbuch.de/buch/
Python+fuer+Ingenieure+und+Naturwissenschaftler/9783446451988

erfahren.

Hinweise, Fehlermeldungen und Anregungen werden über die eMail-Adresse info@woyand.eu
gern entgegen genommen.

Haftungsausschluss
Das Buch wurde mit größtmöglicher Sorgfalt erstellt. Trotzdem können Fehler nicht ganz ausgeschlossen werden. Aus diesem Grund sind die in diesem Buch dargestellten Verfahren mit keinerlei Garantie verbunden. Ich weise darauf hin, dass weder Autor noch Verlag eine Haftung für direkt oder indirekt entstandene Schäden übernehmen, die sich aus der Benutzung dieses Buches ergeben könnten.

Danksagung
Ich danke meiner Frau Annette Woyand für die sorgfältige Durchsicht des Manuskriptes. Frau Mirja Werner und Frau Franziska Kaufmann vom Carl Hanser Verlag danke ich für Ihr Engagement beim Zustandekommen dieses Buches.

Ich wünsche allen Lesern viel Erfolg und Spaß beim Einstieg in Python.

Wuppertal, im März 2017

Hans-Bernhard Woyand
# Inhaltsverzeichnis

## 1 Einführung
- 1.1 Die Programmiersprache Python .................................................. 1  
- 1.2 Hinweise zur Installation ............................................................... 2  
- 1.3 Erste Schritte - Der Interpretermodus von Python ......................... 3
  - 1.3.1 Addition und Subtraktion ......................................................... 4  
  - 1.3.2 Multiplikation und Division ..................................................... 4  
  - 1.3.3 Vergleichsausdrücke ............................................................... 6  
  - 1.3.4 Logische Ausdrücke ................................................................. 7  
  - 1.3.5 Mathematische Funktionen ...................................................... 7  
  - 1.3.6 Grundlegendes über Variablen und Zuweisungen ....................... 8  
  - 1.3.7 Zeichenketten ........................................................................... 10  
  - 1.3.8 Turtle-Grafik ........................................................................... 10  
- 1.4 Python-Programme mit IDLE erstellen ............................................. 12  
- Aufgaben .................................................................................................. 18  
- Lösungen .................................................................................................. 22

## 2 Grundlagen
- 2.1 Einfache Objekttypen ........................................................................ 31
  - 2.1.1 Ganze Zahlen - Integer ................................................................ 31  
  - 2.1.2 Gleitpunktzahlen - Float .......................................................... 32  
  - 2.1.3 Komplexe Zahlen - Complex .................................................... 34  
  - 2.1.4 Zeichenketten - Strings .............................................................. 35  
  - Aufgaben ............................................................................................... 41  
  - Lösungen ............................................................................................... 42
- 2.2 Operatoren und mathematische Standardfunktionen ......................... 45
  - 2.2.1 Operatoren zur arithmetischen Berechnung .................................. 45  
  - 2.2.2 Mathematische Standardfunktionen ........................................... 46  
  - Aufgaben ............................................................................................... 49  
  - Lösungen ............................................................................................... 49
- 2.3 Variablen und Zuweisungen ............................................................... 50
- 2.4 Funktionen ........................................................................................... 55
  - 2.4.1 Funktionen mit Rückgabewert .................................................... 55  
  - 2.4.2 Funktionen ohne Rückgabewert ................................................ 59  
  - Aufgaben ............................................................................................... 60  
  - Lösungen ............................................................................................... 62
- 2.5 Eingabe und Ausgabe ......................................................................... 64
- 2.6 Programmverzweigungen ................................................................... 67
  - 2.6.1 Einfache if-Anweisungen .......................................................... 67  
  - 2.6.2 Erweiterte if-Anweisung ............................................................ 68  
  - Aufgaben ............................................................................................... 70  
  - Lösungen ............................................................................................... 71
- 2.7 Bedingungen ....................................................................................... 72
Inhaltsverzeichnis

2.8 Programmschleifen ................................................................. 73
  2.8.1 for-Schleifen ................................................................. 73
  2.8.2 while-Schleifen .............................................................. 78
Aufgaben ..................................................................................... 81
Lösungen ..................................................................................... 81

3 Vertiefung ................................................................................... 85
  3.1 Listen ..................................................................................... 85
    Aufgaben .................................................................................. 90
    Lösungen ................................................................................ 91
  3.2 Tuples ................................................................................... 95
  3.3 Sets ....................................................................................... 97
  3.4 Dictionaries .......................................................................... 98
    Aufgaben .................................................................................. 102
    Lösungen ................................................................................ 103
  3.5 Slicing .................................................................................. 105
  3.6 List Comprehensions ............................................................... 108
  3.7 Iteratoren und die ZIP-Funktion .............................................. 109
  3.8 Funktionen, Module und Rekursion ........................................ 111
    3.8.1 Schlüsselwort-Parameter ................................................ 111
    3.8.2 Module .......................................................................... 112
    3.8.3 Rekursion ....................................................................... 114
    3.8.4 Globale und lokale Variablen ......................................... 116
  3.9 Turtle-Grafik - verbessert ...................................................... 117
  3.10 Dateien lesen und schreiben .................................................. 120
    Aufgaben .................................................................................. 124
    Lösungen ................................................................................ 129

4 Objektorientiertes Programmieren ............................................ 141
  4.1 Klassen und Objekte ............................................................. 141
  4.2 Konstruktoren und Destruktoren ............................................. 149
  4.3 Überladen von Operatoren ...................................................... 152
  4.4 Vererbung ............................................................................. 156
    Aufgaben .................................................................................. 159
    Lösungen ................................................................................ 161

5 Numerische Berechnungen mit Numpy ..................................... 173
  5.1 Hinweise zur Installation ....................................................... 173
  5.2 Arrays ................................................................................... 173
  5.3 Darstellung von Matrizen ...................................................... 175
  5.4 Spezielle Funktionen ............................................................. 175
  5.5 Operationen .......................................................................... 176
  5.6 Lineare Algebra .................................................................... 178
  5.7 Zufallswerte .......................................................................... 179
    Aufgaben .................................................................................. 180
    Lösungen ................................................................................ 181

6 Graphische Darstellungen mit Matplotlib .................................. 183
  6.1 Hinweise zur Installation ....................................................... 183
  6.2 XY-Diagramme ..................................................................... 183
Inhaltsverzeichnis

6.3 Balken-Diagramme ................................................................. 187
6.4 Torten-Diagramme ................................................................. 189
6.5 Polar-Diagramme ................................................................. 190
6.6 Histogramme ........................................................................ 191
6.7 Subplots ............................................................................... 192
6.8 Axes .................................................................................... 194
6.9 Anmerkungen und Legenden ................................................ 195
Aufgaben ............................................................................. 197
Lösungen ........................................................................... 197

7 Computeralgebra mit Sympy ................................................... 201
7.1 Hinweise zur Installation ....................................................... 201
7.2 Differentiation ...................................................................... 202
7.3 Integration ........................................................................... 203
7.4 Potenzreihen ....................................................................... 205
7.5 Matrizenrechnung - lineare Algebra ..................................... 206
7.6 Die Datentypen Rational und Float ....................................... 208
7.7 Nützliche Ergänzungen ......................................................... 209
Aufgaben ............................................................................. 211
Lösungen ........................................................................... 212

8 3D-Grafik mit VPython ............................................................ 215
8.1 Hinweise zur Installation ........................................................ 215
8.2 Szenen ............................................................................... 216
8.3 Grundkörper ....................................................................... 220
8.4 Faces ............................................................................... 228
8.5 Controls ............................................................................. 231
8.6 Steuerung mit Tastatur und Maus ........................................... 236
Aufgaben ............................................................................. 240
Lösungen ........................................................................... 242

9 Python-Versionen, Programmbibliotheken und Distributionen.... 249
9.1 Python 2 ............................................................................. 250
9.2 Die Python-Distribution Anaconda ....................................... 252
Aufgaben ............................................................................. 253
Lösungen ........................................................................... 255

Literaturverzeichnis ................................................................. 259
Sachwortverzeichnis ................................................................. 261
2 Grundlagen

In diesem Kapitel werden die wichtigsten Datentypen, sowie die zugehörigen Operatoren und Standardfunktionen vorgestellt. Als wichtige Grundelemente von Programmen werden weiterhin die Sequenz, die Programmverzweigung und die Programmschleife eingeführt. Schließlich wird gezeigt, wie mithilfe von Funktionen Programme strukturiert werden können.

2.1 Einfache Objekttypen


2.1.1 Ganze Zahlen – Integer


Mit der eingebauten Funktion type() kann der Typ einer Zahl ermittelt werden. Hier zwei Beispiele:

```python
>>> type(3)
<class 'int'>
```

Die interne Bezeichnung für Ganzzahlen in Python ist int.


```python
>>> int("235")
235
>>> int(235.57)
235
>>> int("255 Grad")
```

32

In der Datenverarbeitung spielen auch andere Zahlensysteme eine bedeutendere Rolle: das Dualsystem, das Oktalsystem sowie das Hexadezimalsystem. Dualzahlen bestehen aus nur aus den zwei Ziffern 0 und 1. Die Zahl 2 bildet die Basis dieses Systems. In Python können wir Dualzahlen direkt eingeben und in das Dezimalsystem umrechnen lassen. Hierzu wird die Zeichenfolge \texttt{0b} vor einer Dualzahl codiert. Mit der Funktion \texttt{bin()} können wir eine Dezimalzahl in eine Dualzahl umrechnen lassen. Dies ist in folgendem Shell-Dialog dargestellt:

```python
>>> x = 0b110011
>>> print(x)
51
>>> y = bin(51)
>>> print(y)
0b110011
```

Das Oktalsystem weist die Basis 8 auf. Oktalzahlen werden durch die Ziffern 0 bis 7 aufgebaut. Python erkennt Oktalzahlen an den beiden führenden Ziffern \texttt{0o} (oder auch die Ziffern \texttt{0O}). Mit der Funktion \texttt{oct()} können Dezimalzahlen ins Oktalsystem umgerechnet werden. Das Hexadezimalsystem weist die Basis 16 auf und kommt deshalb nicht mit den Ziffern 0 bis 9 aus. Es werden die zusätzlichen Zahlzeichen A, B, C, D, E und F verwendet. Die Funktion \texttt{hex()} berechnet die hexadezimale Darstellung aus Dezimalzahlen. Mit den führenden Ziffern \texttt{0x} wird die Darstellung von Hexadezimalzahlen eingeleitet. Dies ist in dem folgenden Shell-Dialog gezeigt:

```python
>>> x = oct(51); y = hex(51)
>>> print(x,y)
0o63 0x33
>>> u = 0o63; v = 0x33
>>> print(u,v)
51 51
```

### 2.1.2 Gleitpunktzahlen – Float

Bei wissenschaftlichen Berechnungen wird oft mit reellen Zahlen gearbeitet. Dies ist jene Zahlenmenge, die sich aus den rationalen Zahlen (also jenen Zahlen, die durch Brüche dargestellt werden können) und den irrationalen Zahlen (Beispiele hierfür sind die Zahl \( \pi \) und die Eulersche Zahl \( e \)) aufbaut. Irrationale Zahlen können potentiell unendlich viele Nachkommastellen haben und sind damit natürlich nicht exakt in einem Digitalrechner nachbildbar. Man begnügt sich dann mit einer angenäherten Darstellung durch eine rationale Zahl.

Da der Ausdruck \(10^{-5}\) in Programmiersprachen nicht direkt codiert werden kann, umschreibt man ihn durch die Notation e-5. Der Buchstabe e darf in diesem Zusammenhang nicht mit der Eulerzahl verwechselt werden! Er steht hier stellvertretend für die Zahl 10 (Basis des Exponenten). Statt eines Kleinbuchstaben kann auch ein Großbuchstabe verwendet werden: 22.3E-5.


```python
# Mehrere Wege zur Darstellung der Zahl 3000,0
x1 = 3000.0
x2 = 3e+3
x3 = 3.0e3
x4 = 3E3
x5 = 3.e+3
print(x1,x2,x3,x4,x5)
```


```python
# Erkennen Sie die Fehler? Korrigieren Sie diese!
x1 = 3000,0
x2 = 3 e+3
x3 = 3.0 e 3
x4 = 3E3.0
print(x1,x2,x3,x4)
```

Da ist zunächst das etwas merkwürdige Verhalten von Python im Falle der Zuweisung \(x1 = 3000,0\). Python erkennt dies nicht als Fehler, sondern weist der Variablen \(x1\) ein so genanntes Tupel (siehe Abschnitt 3.2) zu! Ein Tupel ist ein Verbund aus zwei Zahlen. Als Ergebnis enthält die Variable \(x1\) die beiden Zahlen: 3000 und 0.
Auch bei Gleitpunktzahlen können wir den Objekttyp mittels der eingebauten Funktion type() ermitteln. Er heißt intern float.

```python
>>> type(3000.0)
<class 'float'>
>>> type(3e3)
<class 'float'>
```

Mit der Funktion float(), deren Anwendung nachfolgend gezeigt wird, können andere Objekttypen (z.B. Ganzzahlen oder Strings) in Gleitpunktzahlen umgewandelt werden. Wichtig ist bei der Anwendung dieser Funktion, dass das übergebene Argument auch wirklich in eine Ganzzahl umgewandelt werden kann!

```python
>>> float(235)
235.0
>>> float("235")
235.0
>>> float("235 Grad")
```

Traceback (most recent call last):
  File "<pyshell#28>", line 1, in <module>
    float("235 Grad")
ValueError: could not convert string to float: '235 Grad'

```python
>>> float("0.754e-3")
0.000754
```

### 2.1.3 Komplexe Zahlen – Complex

Eine komplexe Zahl z kann als Zusammenfassung zweier reeller Zahlen a und b aufgefasst werden: \( z = a + bj \). Dabei ist a der so genannte Realteil und b der so genannte Imaginärteil. Das Symbol j wird als imaginäre Einheit bezeichnet. Dabei gilt: \( j = \sqrt{-1} \). Komplexe Zahlen sind in Python fest eingebaut. Das folgende Programmfragment zeigt, wie man komplexe Zahlen zuweist und anschließend ausgibt. Statt \( z1 = 3.0 - 5.0j \) kann auch mittels der Funktion complex() geschrieben werden: \( z1 = \text{complex}(3.0, -5.0) \). Als Ergebnis erhält man die gleiche komplexe Zahl.

```python
# Komplexe Zahlen
z1 = 3.0 - 5.0j
z2 = complex(3.0,-5.0)
z3 = 3 - 5j
print(z1, z2, z3)
```

Wie bei Gleitpunktzahlen können wir die Null nach dem Dezimalpunkt (und auch den Dezimalpunkt) weglassen.

Die imaginäre Einheit j (oder J) muss unmittelbar nach der zweiten Zahl folgen. Es ist also unzulässig, ein Leerzeichen einzufügen. Auch die intuitive Schreibweise \( z = 2 + 3j \) führt zu einem Fehler, weil Python dann die Zahl 3 mit einer (vordefinierten) Vari-
ablen j multiplizieren will. Für den Fall, dass diese Variable vorher verwendet wurde, wird diese Multiplikation dann auch ausgeführt! In dem folgenden Programmfennt ein Fehler enthalten. Finden und korrigieren Sie diesen! Was geschieht beim Aufruf von \texttt{complex(3.0,-5.0j)}?

\begin{verbatim}
# Komplexe Zahlen - Die Fehler suchen und finden!
j = 3.0 - 5.0j                  #1
k = complex(3.0,-5.0j)    #2
m = -2 - 4*j                    #3
print(j, k, m)
\end{verbatim}

Die erste Zuweisung (Zeile #1) ist ungewöhnlich, aber nicht falsch. Dort wird einer Variablen j eine komplexe Zahl 3.0-5.0j zugewiesen. Der Buchstabe j tritt also dop-
pelt auf: in Form eines Variablennamens und in Form der imaginären Einheit. Die zweite Zeile (#2) ist ebenfalls nicht falsch. Dort wird eine komplexe Zahl mittels der Funktion \texttt{complex()} erzeugt. Als Realteil wird die Zahl 3.0 und als Imaginärteil die Zahl -5.0j, die jedoch ebenfalls eine komplexe Zahl ist. Python wertet diese Funktion folgendermaßen aus: \( k = 3.0 + (-5.0j) \) \( j = 3.0 + 5.0 = 8.0 \). Bedenken Sie, dass \( j^*j = -1 \) ist! Der Fehler findet sich in Form eines Leerzeichens in Zeile #3. Richtig muss es dort heißen: \( m = -2 - 4j \).

Schließlich rufen wir wieder die Funktion \texttt{type()} auf:

\begin{verbatim}
# Komplexe Zahlen - Funktion type()
j = 3.0 - 5.0j                  #1
k = complex(3.0,-5.0j)    #2
m = -2 - 4j                    #3
print(type(j), type(k), type(m)
\end{verbatim}

Die Ausgabe dieses Programmfenntes lautet:

\begin{verbatim}
<class 'complex'> <class 'complex'> <class 'complex'>
\end{verbatim}

\subsection*{2.1.4 Zeichenketten – Strings}

Zeichenketten hatten wir schon im ersten Kapitel kennen gelernt. Eine Zeichenkette wird entweder durch zwei obere Anführungszeichen ("\) oder durch zwei Apostrophe ('\) eingeschlossen. Zulässige Zuweisungen mit Strings sind also \texttt{x = "Wuppertal"} oder \texttt{x = 'Wuppertal'}. Ein häufiges Problem besteht darin, dass unsere Programme Texte ausgeben sollen, die ebenfalls Apostrophe oder Anführungszeichen enthalten kön-
nen. Das kann dann dadurch realisiert werden, dass wir die beiden möglichen Be-
grenzungszeichen abwechseln lassen. In dem folgenden Beispiel wird auch gezeigt, wie man mit der Funktion \texttt{type()} den Typ eines Strings abfragen kann.

\begin{verbatim}
>>> x = "Der Programmierer sagte: 'Python ist ein tolles Werkzeug' " 
>>> print(x)  
Der Programmierer sagte: 'Python ist ein tolles Werkzeug'
>>> type(x)
<class 'str'>
\end{verbatim}
Oder als Alternative: x = 'Der Programmierer sagte: “Python ist ein tolles Werkzeug“'.

Im ersten Kapitel war schon kurz gezeigt worden, dass Strings addiert (aneinander gehängt, engl. concatenation) werden können:

```python
>>> x = "Python"
>>> y = "Programmierung"
>>> leer = " 
>>> z = x+y
>>> u = x+leer+y
>>> print(z)
PythonProgrammierung
>>> print(u)
Python Programmierung
```

Strings können also auch Leerzeichen enthalten, oder – wie oben gezeigt – auch nur aus Leerzeichen bestehen. Weiterhin können Strings auch mit ganzen Zahlen multipliziert werden. Bei der Multiplikation mit n wird der String dann n-mal wiederholt. Hier ein Beispiel:

```python
>>> x = "Python"
>>> y = 5*x
>>> print(y)
PythonPythonPythonPythonPython
```

Hierfür gibt es sinnvolle Anwendungen. Viele Programmierer möchten Ausgaben auf dem Computerbildschirm in der folgenden Form erstellen:

```
-------------------------------
Hauptprogramm
-------------------------------

-------------------------------
Programm-Ende
-------------------------------

>>>
```

Die Überschriften sollen also durch Über- und Unterstriche aus Minuszeichen eingerahmt werden. Natürlich können diese Zeilen durch print-Anweisungen der Form

```
print("-------------------------------")
```

erzeugt werden. Dies ist aber umständlich. Viel einfacher ist folgende Lösung:

```python
anzahl_zeichen = 30
strich = anzahl_zeichen * "-"
print(strich)
print(" Hauptprogramm")
print(strich)
```
print(); print()
print(strich)
print(" Programm-Ende")
print(strich)

Falls die Anzahl der Minuszeichen pro Zeile im gesamten Programm verändert werden sollen, so kann dies in dieser Programmvariante ganz einfach durch die Veränderung der Variablen anzahl_zeichen geschehen.

Strings sind so genannte sequentielle Objekttypen. Das bedeutet, dass die Zeichen, die einen String bilden, gewissermaßen durchnummeriert sind. Die folgende Abbildung soll dies erläutern. Wir betrachten die Zeichenkette "Python".

<table>
<thead>
<tr>
<th>P</th>
<th>y</th>
<th>t</th>
<th>h</th>
<th>o</th>
<th>n</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
</tbody>
</table>

Abbildung 2.1 Sequentielle Anordnung von Zeichen

Von links beginnend erhält jedes Zeichen einen Index. Das erste Zeichen hat den Index Null; das letzte Zeichen den Index 5. Der letzte Index ist immer um die Zahl 1 kleiner als die Länge der Zeichenkette. Diese interne Organisation der Zeichenketten wird nachfolgend eine wichtige Rolle spielen.

In dem folgenden Abschnitt sollen die wichtigsten Funktionen zum Arbeiten mit Zeichenketten vorgestellt werden.

len(s)
Diese Funktion ermittelt die Länge einer Zeichenkette s.

```python
>>> x = "Python Programmierung"
>>> len(x)
21
```

str(o)
Die Funktion str() wandelt beliebige Objekte o in Zeichenketten um. Wir zeigen dies am Beispiel der Umwandlung einer Gleitpunktzahl.

```python
>>> y = 120.45
>>> str(y)
'120.45'
```

Neben den genannten Funktionen gibt es auch Methoden zur Behandlung von Zeichenketten. Methoden sind Funktionen, die im Zusammenhang mit einem bestimmten Objekt aufgerufen werden. Näheres hierzu im Abschnitt „Objektorientierte Programmierung“. Hier sind die wichtigsten Methoden für Strings:

s.count(z)
Die Methode count() stellt fest, wie oft die Zeichenkette z in der Zeichenkette s enthalten ist.
>>> x = "Hallo Hallo HalloHallo"
>>> x.count("Hallo")
4

s.center(w)
Die Methode center() wandelt die Zeichenkette in einen String mit der Länge w um. Die Zeichenkette ist dabei in diesem String zentriert. Dies eignet sich z.B. zur Erzeugung von Überschriften. Der Parameter w muss dabei größer als die Länge des Strings s sein; also w>len(s).

>>> x = "Python"
>>> print(x)
Python
>>> y = x.center(14)
>>> print(y)
Python

s.ljust(w), s.rjust(w)
Diese Methoden richten eine Zeichenkette linksbündig (ljust) bzw. rechtsbündig (rjust) in einem String der Länge w aus. Diese Länge w muss größer als die Länge des vorgegebenen Strings s sein; also w>len(s).

>>> x = "Python"
>>> y = x.ljust(20)
>>> z = x.rjust(20)
>>> print(y)
Python
>>> print(z)
Python

s.endswith(z), s.startswith(z)

>>> x = "Python\n"
>>> x.endswith("on")
False
>>> x.endswith("\n")
True
>>> x.startswith("Py")
True

s.find(z)
Die Methode find() sucht den String z in der Zeichenkette s. Wird er gefunden, so gibt diese Methode den Index des ersten Zeichens von z in s zurück. Diese Methode verwendet also die oben dargestellte, sequentielle Organisation der Zeichenketten.
## Sachwortverzeichnis

### A
- **acos()-Funktion** 47
- **and** 73
- **Anmerkungen (Matplotlib)** 195
- **annotate()-Methode** 195
- **append()-Methode** 87
- **arange()-Funktion (Numpy)** 176
- **Arrays** 173
- **arrow() (VPython)** 222
- **asin()-Funktion** 47
- **atan()-Funktion** 47
- **atan2()-Funktion** 47
- **Attribute** 142
- **Axes** 194
- **axes()-Methode** 194

### B
- **Balken-Diagramme** 187
- **bar()-Methode** 187
- **Bedingungen** 72
- **Beschriftung von Graphen** 187
- **Bestimmte Integrale (Sympy)** 204
- **box() (VPython)** 221
- **Buttons (VPython)** 231

### C
- **close()-Funktion** 121
- **complex()-Funktion** 34
- **Computeralgebra** 201
- **cone() (VPython)** 223
- **Controls (VPython)** 231
- **cos()-Funktion** 47
- **cosh()-Funktion** 48
- **count()-Methode** 38
- **count()-Methode für Listen** 87
- **curve() (VPython)** 223
- **cylinder() (VPython)** 222

### D
- **Dateien** 120
- **Dateiobjekt** 121
- **Datenkapselung** 145
- **Datentypen** 31
- **def** 56, 57
- **del** 100
- **Destructeur** 149, 150
- **Determinante (Sympy)** 206
- **Dictionaries** 98
- **Differentiation (Sympy)** 202
- **Differenz von Mengen** 97
- **dtype (Numpy Attribut)** 174

### E
- **Einheitsmatrix (Einsmatrix)** 179
- **Einmaleins (Beispiel)** 77
- **Einrückung von Code** 57
- **ellipsoid() (VPython)** 224
- **else-Klausel** 68
- **Encapsulation** 145
- **encode()-Methode** 38
- **endswith()-Methode** 38
- **exp()-Funktion** 48
- **expand()-Methode (Sympy)** 209
- **Exponent** 33
- **Exponentialfunktion** 48

### F
- **faces() (VPython)** 228
- **Fakultät** 77, 114
- **Fibonacci-Zahlen** 115
- **find()-Methode** 38
- **Float** 32
- **float()-Funktion** 34
- **float-Datentyp (Sympy)** 208
- **Formatetyp** 65
- **for-Schleife** 73, 74, 75
- **Funktion ohne Rückgabewert** 59
- **Funktionen** 55, 111
- **Funktionen mit Rückgabewert** 55
- **Funktionsaufruf** 60
Ganzzahlen 31
Geheimnisprinzip 145
Geometrie-Elemente (Sympy) 210
Gleitpunktzahlen 32
Globale Variable 116, 117
Graphische Darstellungen 183
3D-Grafik 215
grid()-Methode (Matplotlib) 184
Grundkörper (VPython) 217, 220

H
has_key()-Methode 100
helix() (VPython) 225
help()-Funktion 47
Hilfefunktion 47
hist()-Methode 191
Histogramme 191

I
IDLE 2, 3, 12
if-Anweisung (einfach) 67
if-Anweisung (erweitert) 68
Indentation 57
index()-Methode 86
index()-Methode für Listen 87
Information Hiding 145
Inheritance (Vererbung) 156
inner()-Funktion (Numpy) 178
Inneres Produkt (Sympy) 207
Inneres Produkt von Vektoren 154
in-Operator 86
input()-Funktion 64
insert()-Methode 87
Instanzen 142
int()-Funktion 32
Integer 31
Integration (Sympy) 203
Interpreter 3
Inverse Matrix 179
Inverse Matrix (Sympy) 206
isalnum()-Methode 39
isalpha()-Methode 39
isdigit()-Methode 39
islower()-Methode 40
isupper()-Methode 40
items()-Methode 100
Iteratoren 109

J
JPEG-Format 185

K
keys()-Methode 100
Klassen 141
Koeffizientenmatrix 179
Komplexe Zahlen 34
Konstruktor 149, 150
Konstruktor für Vektoren 150
Koordinatensystem (VPython) 220
Kreuzprodukt (Sympy) 207

L
legend()-Methode 196
Legenden (Matplotlib) 195
len()-Funktion 37
len()-Funktion für Listen 87
Line Feed 121
Lineare Algebra 178
Lineare Algebra (Sympy) 206
List Comprehensions 108
Listen 85
log()-Funktion 48
log10()-Funktion 48
Logischer Ausdruck 7
Logischer Operator 72
Lokale Variable 116
Long Integer 31
lower()-Methode 40

M
Mantisse 33
Masken 178
Materialien (VPython) 218, 219
Math. Standardfunktionen 46
Mathematische Funktion 7, 8
Matplotlib 183
Matrizen 175
Matrizenprodukt (Numpy) 177
Matrizenrechnung (Sympy) 206
Maus-Interaktion (VPython) 238
Mengen (Sets) 97
Menues (VPython) 232
Methoden 145
Module 111, 112
Modulo-Operator % 45, 46
| N | Namensraum | 117 |
|   | Nutrulcher Logarithmus | 48 |
|   | not | 73 |
|   | Numpy | 173 |
| O | Oberklasse (Superklasse) | 157 |
|   | object | 157 |
|   | Objekte | 141 |
|   | ones()-Funktion (Numpy) | 175 |
|   | O-Notation | 206 |
|   | OOP | 141 |
|   | open()-Funktion | 120 |
|   | Operationen (Numpy-Arrays) | 176 |
|   | Operatoren | 4, 5, 45 |
|   | Operator-Overloading | 152 |
|   | or | 73 |
| P | Palindrom | 125 |
|   | Parameter | 111 |
|   | PDF-Format | 185 |
|   | pie()-Methode | 189 |
|   | Plain Integer | 31 |
|   | plot()-Methode (Matplotlib) | 184 |
|   | points() (VPython) | 227 |
|   | polar()-Methode | 190 |
|   | Polardiagramme | 190 |
|   | Potenzierung | 48 |
|   | Potenzreihen (Sympy) | 205 |
|   | pow()-Funktion | 48 |
|   | Programmschleifen | 73 |
|   | Programmverzweigung | 67 |
|   | Punktfolge (Graphen) | 186 |
|   | Punkt-Operator | 145 |
|   | pyramid() (VPython) | 225 |
| S | Schalter (VPython) | 234 |
|   | Schieberegler (VPython) | 235 |
|   | Schlüssel (Dictionary) | 98 |
|   | Schlüssel-Wert-Paar | 98 |
|   | Schlüsselwort-Parameter | 111 |
|   | Schnittmenge | 97 |
|   | Schnittstellen | 145 |
|   | Sequentielle Objekttypen | 37 |
|   | Sets (Mengen) | 97 |
|   | shape (Numpy Attribut) | 175 |
|   | Shell | 3 |
|   | show()-Methode (Matplotlib) | 184 |
|   | sin()-Funktion | 47 |
|   | sinh()-Funktion | 48 |
|   | Slicing | 105 |
|   | Slider (VPython) | 235 |
|   | sort()-Methode | 87 |
|   | split()-Methode | 40 |
|   | sqrt()-Funktion | 48 |
|   | Stack (Stapel) | 90, 91 |
|   | Stapel (stack) | 90, 91 |
|   | startswith()-Methode | 38 |
|   | Stereographische Darstellung (VPython) | 242 |
|   | Steuerung (VPython) | 236 |
|   | STL-Datei | 125 |
|   | STL-Dateien lesen (VPython) | 229 |
|   | str()-Funktion | 37 |
|   | strip()-Funktion | 122 |
|   | Subklasse (Unterkasse) | 157 |
|   | Subplots | 192 |
|   | subs()-Methode (Sympy) | 209 |
|   | Summation | 76, 79 |
|   | Superklasse (Oberklasse) | 157 |
|   | SVG-Format | 185 |
|   | sympify()-Funktion (Sympy) | 209 |
|   | Szenen (VPython) | 216 |

<p>| R | RAD (Rapid Application Development) | 1 |
|   | radians()-Funktion | 48 |
|   | random.randn() | 179 |</p>
<table>
<thead>
<tr>
<th>T</th>
<th>W</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tabulatorzeichen</td>
<td>Warteschlange (Queue)</td>
</tr>
<tr>
<td>tan()-Funktion</td>
<td>160, 167</td>
</tr>
<tr>
<td>tanh()-Funktion</td>
<td>Wert (Dictionary)</td>
</tr>
<tr>
<td>Tastaturlange (VPython)</td>
<td>47</td>
</tr>
<tr>
<td>Textdatei lesen</td>
<td>while-Schleife</td>
</tr>
<tr>
<td>Textdatei schreiben</td>
<td>48</td>
</tr>
<tr>
<td>Textur (VPython)</td>
<td>Wiederholungsanweisungen</td>
</tr>
<tr>
<td>TIFF-Format</td>
<td>237</td>
</tr>
<tr>
<td>title()-Methode (Matplotlib)</td>
<td>184</td>
</tr>
<tr>
<td>together()-Funktion (Sympy)</td>
<td>209</td>
</tr>
<tr>
<td>Toggles (VPython)</td>
<td>234</td>
</tr>
<tr>
<td>Tortendiagramme</td>
<td>189</td>
</tr>
<tr>
<td>Transponierte Matrix</td>
<td></td>
</tr>
<tr>
<td>(Sympy)</td>
<td>206</td>
</tr>
<tr>
<td>Triangulierung</td>
<td></td>
</tr>
<tr>
<td>Tupel</td>
<td>126</td>
</tr>
<tr>
<td>Tuples</td>
<td>33</td>
</tr>
<tr>
<td>Turtle-Grafik</td>
<td>95</td>
</tr>
<tr>
<td>type()-Funktion</td>
<td>10, 11, 12, 117</td>
</tr>
<tr>
<td>U</td>
<td></td>
</tr>
<tr>
<td>Überladen von Operatoren</td>
<td>152</td>
</tr>
<tr>
<td>Unbestimmte Integrale</td>
<td></td>
</tr>
<tr>
<td>(Sympy)</td>
<td>203</td>
</tr>
<tr>
<td>Uneigentliche Integrale</td>
<td></td>
</tr>
<tr>
<td>(Sympy)</td>
<td>205</td>
</tr>
<tr>
<td>unicode()-Funktion</td>
<td>37</td>
</tr>
<tr>
<td>Unterklasse (Subklasse)</td>
<td>157</td>
</tr>
<tr>
<td>Unterliste (sub-list)</td>
<td>86</td>
</tr>
<tr>
<td>upper()-Methode</td>
<td>40</td>
</tr>
<tr>
<td>V</td>
<td></td>
</tr>
<tr>
<td>values()-Methode</td>
<td>100</td>
</tr>
<tr>
<td>Variablen</td>
<td>8, 9, 50</td>
</tr>
<tr>
<td>Variablenennamen</td>
<td>9</td>
</tr>
<tr>
<td>Varianz</td>
<td>180</td>
</tr>
<tr>
<td>Vektoren (VPython)</td>
<td>219</td>
</tr>
<tr>
<td>Verbund</td>
<td>142</td>
</tr>
<tr>
<td>Vereinigung von Mengen</td>
<td>97</td>
</tr>
<tr>
<td>Vererbung (Inheritance)</td>
<td>156</td>
</tr>
<tr>
<td>Vergleichsausdruck</td>
<td>6, 7</td>
</tr>
<tr>
<td>Vergleichsaußdrücke</td>
<td></td>
</tr>
<tr>
<td>(Numpy)</td>
<td>178</td>
</tr>
<tr>
<td>Vergleichsoperator</td>
<td>72</td>
</tr>
<tr>
<td>VIDLE</td>
<td>215</td>
</tr>
<tr>
<td>Visual</td>
<td>215</td>
</tr>
<tr>
<td>Vollständige Alternative</td>
<td>68</td>
</tr>
<tr>
<td>VPython</td>
<td>215</td>
</tr>
<tr>
<td>VR-Darstellung</td>
<td>242</td>
</tr>
</tbody>
</table>