



Hans-Bernhard  
WOYAND

# PYTHON

## FÜR INGENIEURE UND NATURWISSENSCHAFTLER

Einführung in die Programmierung,  
mathematische Anwendungen und  
Visualisierungen

5., aktualisierte und erweiterte Auflage



Im Internet: Beispiele und  
Lösungen zu den Aufgaben

HANSER





**Ihr Plus – digitale Zusatzinhalte!**

Auf unserem Download-Portal finden Sie zu diesem Titel kostenloses Zusatzmaterial. Geben Sie dazu einfach diesen Code ein:

plus-r5mzv-gawa5

**[plus.hanser-fachbuch.de](http://plus.hanser-fachbuch.de)**



**Bleiben Sie auf dem Laufenden!**

Hanser Newsletter informieren Sie regelmäßig über neue Bücher und Termine aus den verschiedenen Bereichen der Technik. Profitieren Sie auch von Gewinnspielen und exklusiven Leseproben. Gleich anmelden unter

**[www.hanser-fachbuch.de/newsletter](http://www.hanser-fachbuch.de/newsletter)**



Hans-Bernhard Woyand

# **Python für Ingenieure und Naturwissenschaftler**

Einführung in die Programmierung, mathematische Anwendungen und Visualisierungen

5., aktualisierte und erweiterte Auflage

**HANSER**

Über den Autor:

Prof. Dr.-Ing. Hans-Bernhard Woyand, Bergische Universität Wuppertal, Fakultät für Maschinenbau und Sicherheitstechnik

Aus Gründen der besseren Lesbarkeit wird auf die gleichzeitige Verwendung der Sprachformen männlich, weiblich und divers (m/w/d) verzichtet. Sämtliche Personenbezeichnungen gelten gleichermaßen für alle Geschlechter.



Print-ISBN: 978-3-446-48236-4

E-Book-ISBN: 978-3-446-48237-1

E-Pub-ISBN: 978-3-446-48336-1

Alle in diesem Werk enthaltenen Informationen, Verfahren und Darstellungen wurden zum Zeitpunkt der Veröffentlichung nach bestem Wissen zusammengestellt. Dennoch sind Fehler nicht ganz auszuschließen. Aus diesem Grund sind die im vorliegenden Werk enthaltenen Informationen für Autor:innen, Herausgeber:innen und Verlag mit keiner Verpflichtung oder Garantie irgendeiner Art verbunden.

Autor:innen, Herausgeber:innen und Verlag übernehmen infolgedessen keine Verantwortung und werden keine daraus folgende oder sonstige Haftung übernehmen, die auf irgendeine Weise aus der Benutzung dieser Informationen – oder Teilen davon – entsteht. Ebenso wenig übernehmen Autor:innen, Herausgeber:innen und Verlag die Gewähr dafür, dass die beschriebenen Verfahren usw. frei von Schutzrechten Dritter sind. Die Wiedergabe von Gebrauchsnamen, Handelsnamen, Warenbezeichnungen usw. in diesem Werk berechtigt also auch ohne besondere Kennzeichnung nicht zu der Annahme, dass solche Namen im Sinne der Warenzeichen- und Markenschutz-Gesetzgebung als frei zu betrachten wären und daher von jedermann benützt werden dürften.

Die endgültige Entscheidung über die Eignung der Informationen für die vorgesehene Verwendung in einer bestimmten Anwendung liegt in der alleinigen Verantwortung des Nutzers.

Bibliografische Information der Deutschen Nationalbibliothek:

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

Dieses Werk ist urheberrechtlich geschützt.

Alle Rechte, auch die der Übersetzung, des Nachdruckes und der Vervielfältigung des Werkes, oder Teilen daraus, vorbehalten. Kein Teil des Werkes darf ohne schriftliche Einwilligung des Verlages in irgendeiner Form (Fotokopie, Mikrofilm oder einem anderen Verfahren), auch nicht für Zwecke der Unterrichtsgestaltung – mit Ausnahme der in den §§ 53, 54 UrhG genannten Sonderfälle –, reproduziert oder unter Verwendung elektronischer Systeme verarbeitet, vervielfältigt oder verbreitet werden.

Wir behalten uns auch eine Nutzung des Werks für Zwecke des Text- und Data Mining nach § 44b UrhG ausdrücklich vor.

© 2025 Carl Hanser Verlag GmbH & Co. KG, München

[www.hanserfachbuch.de](http://www.hanserfachbuch.de)

Lektorat: Dipl.-Ing. Natalia Silakova-Herzberg

Herstellung: Frauke Schafft, Malin Göthel

Coverkonzept: Marc Müller-Bremer, [www.rebranding.de](http://www.rebranding.de), München

Covergestaltung: Tom West

Satz: Eberl & Koesel Studio, Kempten

Druck: CPI Books GmbH, Leck

Printed in Germany

# Vorwort

Seit vielen Jahren halte ich an der Bergischen Universität Wuppertal die Lehrveranstaltung Informatik im Grundstudium Maschinenbau. Mehr als 10 Jahre haben wir in diesen Kursen die Programmiersprache C/C++ eingesetzt. Seit mehreren Jahren verwenden wir stattdessen nun *Python* als erste Programmiersprache und das mit großem Erfolg. Es zeigte sich, dass gerade bei Ingenieuren, die oftmals Schwierigkeiten mit dem algorithmischen Denken haben, Python einen leichteren Zugang ermöglicht.

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 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 grafischen 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.

## Webseite zum Buch

Zu dem vorliegenden Buch existiert eine Webseite, 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](http://woyand.eu/python_buch)*

Außerdem sind die Daten zum Buch unter folgender Webadresse aufrufbar:

*<http://www.plus.hanser-fachbuch.de>*.

Die Zugangsdaten finden Sie auf Seite 1 des Buches.

Hinweise, Fehlermeldungen und Anregungen werden über die E-Mail-Adresse

*[info@woyand.eu](mailto:info@woyand.eu)*

gern entgegengenommen.

## 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 Manuskripts. Frau Mirja Werner 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*



## ■ Vorwort zur zweiten Auflage

Für die zweite Auflage wurden einige inhaltliche Verbesserungen vorgenommen und ein neues Kapitel hinzugefügt. In diesem Kapitel wird gezeigt, wie numerische Berechnungen mit der Python-Bibliothek Scipy durchgeführt werden können. Scipy ist sehr umfangreich. Deshalb werden – dem Ansatz dieses Buches entsprechend – wichtige Teilbereiche dieser Software-Bibliothek auf wenigen Seiten vorgestellt. Behandelt wird die numerische Berechnung von Integralen, die Interpolation, die Berechnung von Nullstellen, die numerische Optimierung, die Signalanalyse mit der schnellen Fourier Transformation (FFT) sowie die numerische Integration gewöhnlicher Differenzialgleichungen.

Ich hoffe, dass diese Erweiterung des Buchs für viele Leser hilfreich und anregend ist und wünsche viel Erfolg und Spaß mit Python.

*Wuppertal, im Juli 2018*

*Hans-Bernhard Woyand*

## ■ Vorwort zur dritten Auflage

Für die dritte Auflage wurde das Kapitel 8 (3D-Grafik mit VPython) völlig überarbeitet, da sich die neue Version VPython 7 deutlich von den Vorgängerversionen unterscheidet. Mit dieser neuen Version von VPython können die dreidimensionalen Szenen im Webbrowser dargestellt werden. Auch die Benutzerinteraktionen wurden erheblich vereinfacht und neu strukturiert.

Weiterhin wurde das Kapitel 10 (Numerische Analysen mit Scipy) um zwei Themen erweitert. Es handelt sich um die Erzeugung von Dreiecksnetzen mit der Delaunay-Triangulierung sowie um die Berechnung der konvexen Hülle einer Punktmenge. Vier weitere Aufgaben mit kommentierten Lösungen wurden dem Buch hinzugefügt.

*Wuppertal, im Januar 2019*

*Hans-Bernhard Woyand*

## ■ Vorwort zur vierten Auflage

Für die vierte Auflage des Buches wurden einige kleinere, inhaltliche Verbesserungen vorgenommen. Zudem wurde ein neues Kapitel 11 mit dem Titel „Bildverarbeitung mit scikit-image“ hinzugefügt. Scikit-image ist ein Erweiterungspaket der Programmiersprache Python, das zur automatisierten Manipulation und Analyse von Bildern (Fotos, Grafiken) entwickelt wurde. Es wird gezeigt, wie Bilder eingelesen, gewandelt und geschrieben werden können. Weiterhin werden Algorithmen zum Auffinden von Kanten, Ecken und Kreisen sowie zum Abgleich von Vorlagen vorgestellt und erläutert. Zwei weitere Aufgaben und deren Lösungen wurden ergänzt.

*Witten, im November 2020*

*Hans-Bernhard Woyand*

## ■ Vorwort zur fünften Auflage

Für die fünfte Auflage dieses Buches wurden weitere inhaltliche Verbesserungen vorgenommen und einige Fehler entfernt. Das Kapitel 7 wurde etwas erweitert. Zusätzlich wurden zwei weitere Kapitel hinzugefügt.

Das Kapitel 12 mit dem Titel „Einführung in die Gestaltung grafischer Benutzeroberflächen mit Tkinter“ führt den Leser in die Erzeugung interaktiver, visueller Benutzerschnittstellen mit Fenstern, Buttons und Menüs ein. Dabei werden mehrere Beispiele vorgestellt und erläutert, die als Grundlage und Ausgangspunkt für eigene Lösungen dienen können.

Das Kapitel 13 mit dem Titel „Grundlagen der funktionalen Programmierung mit Python“ stellt das Paradigma der funktionalen Programmierung vor, die Python ebenso wie die objektorientierte Programmierung unterstützt. Die funktionale Programmierung baut vorwiegend auf der Nutzung von Listen auf und nutzt einen speziellen Typ von Funktionen, die Nebeneffekte ausschließen. Hierdurch entsteht stabiler und effizienter Code. Dieses Paradigma hat sich bei der Programmierung im technisch-wissenschaftlichen Bereich bewährt.

An dieser Stelle möchte ich auch meiner Lektorin Frau Natalia Silakova und Frau Christina Kubiak vom Lektorat des Hanser-Verlages für ihr Engagement beim Zustandekommen dieses Buches danken.

*Witten, im August 2024*

*Hans-Bernhard Woyand*

# Inhalt

<b>Vorwort</b> .....	<b>V</b>
<b>1 Einführung</b> .....	<b>1</b>
1.1 Die Programmiersprache Python .....	1
1.2 Hinweise zur Installation .....	2
1.3 Erste Schritte - der Python-Interpreter .....	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 .....	9
1.3.7 Zeichenketten (Strings) .....	10
1.3.8 Turtle-Grafik .....	10
1.4 Python-Programme mit IDLE erstellen .....	12
1.5 Aufgaben .....	18
1.6 Lösungen .....	22
<b>2 Grundlagen</b> .....	<b>31</b>
2.1 Einfache Objekttypen .....	31
2.1.1 Ganze Zahlen - Integer .....	31
2.1.2 Gleitpunktzahlen - Float .....	33
2.1.3 Komplexe Zahlen - Complex .....	34
2.1.4 Zeichenketten - Strings .....	36
2.1.5 Aufgaben .....	41
2.1.6 Lösungen .....	43
2.2 Operatoren und mathematische Standardfunktionen .....	46
2.2.1 Operatoren zur arithmetischen Berechnung .....	46
2.2.2 Mathematische Standardfunktionen .....	47

2.2.3	Aufgaben	49
2.2.4	Lösungen	50
2.3	Variablen und Zuweisungen	51
2.4	Funktionen	56
2.4.1	Funktionen mit Rückgabewert	57
2.4.2	Funktionen ohne Rückgabewert	60
2.4.3	Aufgaben	62
2.4.4	Lösungen	64
2.5	Ein- und Ausgabe	65
2.6	Programmverzweigungen	68
2.6.1	Einfache if-Anweisung	68
2.6.2	Erweiterte if-Anweisung	70
2.6.3	Aufgaben	72
2.6.4	Lösungen	73
2.7	Bedingungen	73
2.8	Programmschleifen	75
2.8.1	for-Schleifen	76
2.8.2	while-Schleifen	80
2.9	Aufgaben	84
2.10	Lösungen	85
<b>3</b>	<b>Vertiefung</b>	<b>89</b>
3.1	Listen	89
3.1.1	Aufgaben	94
3.1.2	Lösungen	96
3.2	Tupels	100
3.3	Sets – Mengen	101
3.4	Dictionaries	103
3.4.1	Aufgaben	106
3.4.2	Lösungen	107
3.5	Slicing	110
3.6	List Comprehension	113
3.7	Iteratoren und die zip-Funktion	114
3.8	Funktionen, Module und Rekursion	116
3.8.1	Schlüsselwort-Parameter	116
3.8.2	Module	117
3.8.3	Rekursion	119
3.8.4	Globale und lokale Variablen	121

3.9	Turtle-Grafik – verbessert .....	123
3.10	Dateien lesen und schreiben .....	125
3.11	Aufgaben .....	130
3.12	Lösungen .....	136
<b>4</b>	<b>Objektorientiertes Programmieren .....</b>	<b>149</b>
4.1	Klassen und Objekte .....	149
4.1.1	Die Grundidee .....	150
4.1.2	Klassen .....	151
4.1.3	Methoden .....	153
4.2	Konstruktoren und Destruktoren .....	158
4.3	Überladen von Operatoren .....	161
4.4	Vererbung .....	165
4.5	Aufgaben .....	169
4.6	Lösungen .....	172
<b>5</b>	<b>Numerische Berechnungen mit Numpy .....</b>	<b>183</b>
5.1	Hinweise zur Installation .....	183
5.2	Arrays .....	184
5.3	Darstellung von Matrizen .....	185
5.4	Spezielle Funktionen .....	186
5.5	Operationen .....	187
5.6	Lineare Algebra .....	188
5.7	Zufallswerte .....	190
5.8	Aufgaben .....	190
5.9	Lösungen .....	192
<b>6</b>	<b>Grafische Darstellungen mit Matplotlib .....</b>	<b>195</b>
6.1	Hinweise zur Installation .....	195
6.2	XY-Diagramme .....	195
6.3	Balkendiagramme .....	200
6.4	Tortendiagramme .....	202
6.5	Polardiagramme .....	203
6.6	Histogramme .....	204
6.7	Subplots .....	205

6.8	Axes .....	207
6.9	Anmerkungen und Legenden .....	209
6.10	Aufgaben .....	211
6.11	Lösungen .....	211
<b>7</b>	<b>Computeralgebra mit Sympy .....</b>	<b>215</b>
7.1	Hinweise zur Installation .....	216
7.2	Differentiation .....	216
7.3	Integration .....	217
7.3.1	Unbestimmte Integrale .....	218
7.3.2	Bestimmte Integrale .....	218
7.3.3	Uneigentliche Integrale .....	219
7.4	Potenzreihen .....	220
7.5	Matrizenrechnung - lineare Algebra .....	220
7.6	Die Datentypen Rational und Float .....	222
7.7	Nützliche Ergänzungen .....	223
7.8	Reihen und Reihensummen .....	226
7.9	Grenzwerte von Funktionen und Reihen .....	227
7.10	Gleichungen und Gleichungssysteme lösen .....	228
7.11	Aufgaben .....	229
7.12	Lösungen .....	231
<b>8</b>	<b>3D-Grafik mit VPython 7 .....</b>	<b>237</b>
8.1	Hinweise zur Installation .....	237
8.2	Szenen .....	238
8.3	Grundkörper .....	243
8.4	Dreieck- und Viereckflächen (Triangle/Quad) .....	249
8.4.1	triangle .....	249
8.4.2	quad .....	250
8.4.3	STL-Dateien lesen und mit VPython darstellen .....	251
8.5	Widgets .....	254
8.6	Steuerung mit Tastatur und Maus .....	258
8.7	Aufgaben .....	266
8.8	Lösungen .....	268

<b>9</b>	<b>Python-Versionen, Programmbibliotheken und Distributionen</b> .....	<b>277</b>
9.1	Python 2 .....	278
9.2	Die Python-Distribution Anaconda .....	280
9.3	Die Python-Distribution WinPython .....	282
9.4	Aufgaben .....	282
9.5	Lösungen .....	284
<b>10</b>	<b>Numerische Analysen mit Scipy</b> .....	<b>287</b>
10.1	Hinweise zur Installation .....	288
10.2	Numerische Berechnung von Integralen .....	288
10.3	Interpolation .....	290
10.4	Berechnung von Nullstellen – Rootfinding .....	293
10.5	Optimierung .....	295
10.6	Signalanalyse mit der Schnellen Fourier Transformation (FFT) .....	299
10.7	Numerische Integration gewöhnlicher Differenzialgleichungen .....	303
10.8	Delaunay-Triangulierung .....	309
10.9	Berechnung der konvexen Hülle .....	310
10.10	Aufgaben .....	312
10.11	Lösungen .....	314
<b>11</b>	<b>Bildverarbeitung mit scikit-image</b> .....	<b>323</b>
11.1	Hinweise zur Installation .....	323
11.2	Bilder einlesen, darstellen und ausgeben .....	323
11.3	Farbbilder in Graustufenbilder wandeln und Bilder skalieren ..	325
11.4	Graustufenbild durch Programmanweisungen erzeugen .....	326
11.5	Ecken ermitteln – Corner Detection .....	328
11.6	Kanten detektieren – Canny-Filter .....	329
11.7	Kreise erkennen – Hough-Transformation .....	330
11.8	Abgleich von Vorlagen – Template-Matching .....	333
11.9	Aufgaben .....	335
11.10	Lösungen .....	336

<b>12</b>	<b>Einführung in die Gestaltung grafischer Benutzeroberflächen mit Tkinter</b> .....	<b>339</b>
12.1	Ein Fenster erzeugen .....	340
12.2	Text anzeigen – das Label-Widget .....	340
12.3	Der Grid-Packer .....	341
12.4	Die Eigenschaften der Schrift verändern .....	342
12.5	Die Größe des Ausgabefensters verändern .....	343
12.6	Eingabefelder hinzufügen – das Entry-Widget .....	344
12.7	Einen Button erzeugen – das Button-Widget .....	345
12.8	Einen Schieberegler verwenden – das Scale-Widget .....	346
12.9	Eine Auswahlliste erzeugen – das Combobox-Widget .....	348
12.10	Eine Zeichenfläche für Grafiken erzeugen – das Canvas-Widget	349
12.11	Linien, Rechtecke und Kreise in das Canvas-Widget zeichnen ..	350
12.12	Benutzerkoordinaten im Canvas verwenden .....	351
12.13	Eine mathematische Funktion im Canvas darstellen .....	354
12.14	Eine Checkbox erzeugen .....	356
12.15	Grafische Elemente mit Checkboxes ein- und ausblenden .....	358
12.16	Radio-Buttons erzeugen .....	360
12.17	Grafische Elemente mit Radio-Buttons ein- und ausblenden ....	361
12.18	Eine MessageBox erzeugen .....	364
12.19	Eine MessageBox zum Abfangen eines Fehlers verwenden .....	366
12.20	Erweiterte Layoutgestaltung: die pack()- und die place()-Methode .....	368
12.21	Eine Statuszeile erzeugen .....	370
12.22	Ein Menü erzeugen .....	371
12.23	Grafische Elemente mit Menüs erzeugen und löschen .....	373
12.24	Grafische Elemente mit Icons erzeugen und löschen .....	375
12.25	Aufgaben .....	377
12.26	Lösungen .....	378
<b>13</b>	<b>Grundlagen der funktionalen Programmierung mit Python</b> ..	<b>381</b>
13.1	Die Grundidee .....	381
13.2	Anonyme Funktionen – Lambda-Funktionen .....	383
13.3	Die map()-Funktion .....	384



13.4	Die filter()-Funktion .....	384
13.5	Die Funktion reduce() .....	385
13.6	List Comprehension als Alternative .....	386
13.7	Beispiel: Zahlen aus einem Text extrahieren und summieren ...	387
13.8	Aufgaben .....	389
13.9	Lösungen .....	390
	<b>Literaturverzeichnis .....</b>	<b>395</b>
	<b>Index .....</b>	<b>397</b>



# 1

# Einführung

*In diesem Kapitel wird die Programmiersprache Python vorgestellt. Nach Bemerkungen zur Installation dieser Sprache wird gezeigt, wie Python interaktiv ausgeführt werden kann. Schließlich wird dargestellt, wie ein vollständiges Python-Programm geschrieben wird. Neben dem Umgang mit Zahlen und Ausdrücken wird insbesondere auf das Konzept der Variablen eingegangen.*

## ■ 1.1 Die Programmiersprache Python

Die Sprache Python wurde in den neunziger Jahren des letzten Jahrhunderts von *Guido van Rossum* entwickelt. Mittlerweile ist Python eine der meistgenutzten Programmiersprachen. Die wesentlichen Vorzüge dieser Sprache sind:

- Python ist *leicht zu erlernen*, unterstützt mehrere Programmierparadigmen und ist klar strukturiert.
- Python eignet sich insbesondere zur *schnellen Entwicklung* von Softwareprototypen (RAD - Rapid Application Development). Dies ist gerade für Anwender in technisch-naturwissenschaftlichen Bereichen ein wichtiger Aspekt.
- Python ist *portabel*. Die Programme, die mit dieser Sprache geschrieben werden, laufen im Allgemeinen ohne Änderungen auf LINUX-, MAC OS- und Windows-Betriebssystemen.
- Python beinhaltet eine *Fülle von Anwendungspaketen* für unterschiedliche Bereiche. Ob grafische Darstellungen, numerische Berechnungen, Datenbanken oder Webanwendungen zu entwickeln sind: Der Anwender hat zumeist nur das Problem, aus der Vielzahl von angebotenen Lösungen, die für ihn geeignete zu finden. Es gibt kaum einen Anwendungsbereich, für den Python nicht schon vorgefertigte „Tools“ zur Verfügung stellt.
- Python ist *kostenlos*. Die Sprache kann auch für kommerzielle Zwecke kostenfrei genutzt werden.

- Python kann *leicht erweitert* oder selbst in Programme anderer Sprachen „eingebettet“ werden. Ein etwas fortgeschrittener Python-Programmierer ist in der Lage, eigene Anwendungspakete zu entwickeln.
- Mit Python können Programme geschrieben werden. Es ist aber auch möglich, einzelne Anweisungen interaktiv in der sogenannten Python-Shell auszuführen. Diese *interaktive Ausführung* hilft beim Ausprobieren von Sprachstrukturen und auch beim Testen von Programmen.

Im Rahmen dieses Buches steht die *leichte Erlernbarkeit* dieser Programmiersprache im Vordergrund.

## ■ 1.2 Hinweise zur Installation

In diesem Buch wird die Installation der Software *nicht* gezeigt. Für Nutzer des MS-Windows-Betriebssystems gibt es allerdings ein Zusatzdokument im PDF-Format, das die Installation Schritt für Schritt darstellt. Dieses Dokument kann auf den Webseiten:

[http://www.woyand.eu/python\\_buch/](http://www.woyand.eu/python_buch/) bzw. [plus.hanser-fachbuch.de](http://www.plus.hanser-fachbuch.de)

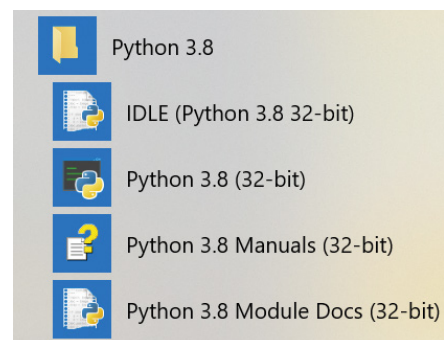
heruntergeladen werden. Zusammen mit der Anleitung kann der Leser dort auch ein ZIP-Verzeichnis mit einigen Installationsdateien sowie die Lösungen der Aufgaben und den Programmcode der Beispiele erhalten.

Eine Darstellung der Installation für alle gängigen Betriebssysteme (LINUX, Mac OS) würde den Rahmen des Buchs sprengen. Auf LINUX-Systemen ist Python oftmals schon vorinstalliert. Alle Installationsdateien sind auf der folgenden Python-Homepage unter dem Menüpunkt „Download“ zu finden:

<http://www.python.org>

Die Sprachversion, die in diesem Buch verwendet wird ist Python 3. Die Beispiele und Aufgaben wurden mit der relativ neuen Version 3.12.3 getestet.

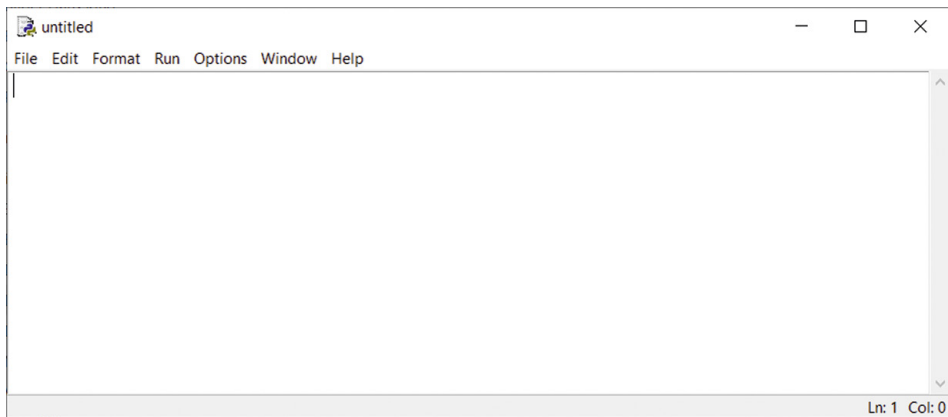
Nach erfolgreicher Installation kann Python gestartet werden. Hierzu wird über das Windows-Startmenü der Eintrag „Alle Programme“ ausgewählt. Um mit Python zu arbeiten, wählen Sie wie im Bild rechts beispielhaft gezeigt einfach „Python 3.8“ und dann den Eintrag „IDLE ...“ aus.



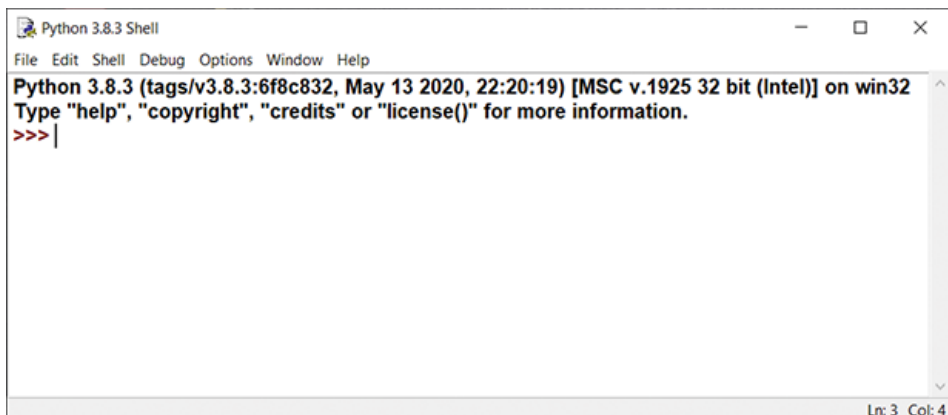
IDLE ist der Name der *Standard-Entwicklungsumgebung* für Python. Um einen Brief zu schreiben, kann beispielsweise Microsoft Word zur Eingabe verwendet werden. Soll dagegen ein Python-Programm geschrieben werden, so gibt man dieses Programm mithilfe von *IDLE* ein und kann es anschließend starten. Für den schnellen Zugriff auf IDLE kann eine Verknüpfung auf dem Desktop abgelegt werden.

## ■ 1.3 Erste Schritte – der Python-Interpreter

Wenn IDLE aufgerufen wird – wie im letzten Abschnitt gezeigt – so wird entweder das Editorfenster wie unten gezeigt oder das Shell-Fenster geöffnet. Welches Fenster geöffnet wird, hängt von den Voreinstellungen der Software ab.



Für den Fall, dass das Editorfenster geöffnet wurde, wählen wir aus dem Menü die Kommandofolge „Run → Python Shell“ aus. Daraufhin öffnet sich die „Python-Shell“ (siehe Bild). In den folgenden Unterkapiteln wird zunächst diese Shell verwendet.



**Hinweis**

Je nach Software-Version kann sich nach dem Aufruf von IDLE zunächst das Editorfenster oder die Python-Shell öffnen. Welches der Fenster nach dem Aufruf von IDLE geöffnet wird, kann über eine Einstellung im Optionen-Menü festgelegt werden. Wählen Sie hierzu in IDLE „Options“ und dann „Configure IDLE“. Unter der Registerkarte „General“ kann unter dem Eintrag „At startup“ die Option „Open Edit Window“ oder „Open Shell Window“ gewählt werden.

Die drei aufeinanderfolgenden Zeichen „>>>“ werden *Eingabeaufforderung* oder *Eingabeprompt* genannt. Sie können nun damit beginnen, Ausdrücke in der Sprache Python einzugeben und diese unmittelbar ausführen zu lassen. Dieser interaktive Modus der Programm-Eingabe und -Ausführung ist typisch für sogenannte Interpreter-Sprachen wie Python. Gerade Programmieranfänger profitieren viel davon, weil sie einzelne Befehle unmittelbar ausprobieren können.

Wir beginnen nun mit einer kleinen Rundtour durch Python. Wenn wir etwas falsch machen, weil wir beispielsweise gegen die Regeln der Sprache verstoßen, so erhalten wir eine Fehlermeldung, die vom System in *roter Farbe* ausgegeben wird.

### 1.3.1 Addition und Subtraktion

Im Folgenden wird nicht mehr das ganze Fenster, sondern nur noch der Eingabeprompt und die darauffolgende Antwort des Computers dargestellt. Probieren Sie am besten die nachfolgenden Eingaben selbst aus!

```
>>> 1+2
3
>>> 1-2
-1
```

Wie wir sehen, wird nach jeder unserer Eingaben die Antwort von Python ausgegeben. Der Operator + bedeutet dabei Addition, der Operator - Subtraktion.

### 1.3.2 Multiplikation und Division

Die folgenden Eingaben zeigen, dass mit den Zeichen \* eine Multiplikation und mit / eine Division durchgeführt werden kann.

```
>>> 1+2*3
7
>>> (1+2)*3
9
```

```
>>> 2+7/5
3.4
>>> 1/2
0.5
```

Diese Beispiele zeigen, dass wir mithilfe von Klammern die Auswertungsreihenfolge innerhalb von mathematischen Ausdrücken beeinflussen können. Dies geschieht so, wie wir das intuitiv aus dem Prinzip „Punkt- vor Strichrechnung“ vermuten. Zuerst wird die Multiplikation bzw. die Division ausgeführt, anschließend die Addition bzw. die Subtraktion. Das Setzen von Klammern erzwingt gegebenenfalls eine andere Auswertungsreihenfolge.

Der Ausdruck  $2+7/5$  ergibt die Zahl 3.4. Statt des Kommas, das wir beim Schreiben auf Papier verwenden, werden Dezimalzahlen in fast allen Programmiersprachen mithilfe eines *Dezimalpunkts* codiert. Statt 5,0 schreiben wir also 5.0. Wir sprechen deshalb in der Programmierung von *Gleitpunktzahlen*. Wenn wir das Divisionszeichen in Python 3.x verwenden, so wird immer eine sogenannte *Gleitpunkt-Division* durchgeführt, d.h. auch wenn beide Operanden ganzzahlig sind, ist das Ergebnis eine Gleitpunktzahl.

```
>>> 7/5
1.4
```

Das ist wichtig zu wissen! In der älteren Python-Version 2.x wurde eine Ganzzahl-Division durchgeführt, wenn beide Operanden ganze Zahlen waren. Dies führte oft zu unbeabsichtigten Fehlern bei Ausdrücken wie  $1/2$ . Das Ergebnis war dann 0. Soll in Python 3.x eine Ganzzahl-Division durchgeführt werden, so muss dies mit dem speziellen Operator `//` codiert werden. Hierzu auch ein Beispiel:

```
>>> 7//5
1
>>> 22//4
5
>>> 22.0//4.0
5.0
```

Wenn man sich für den Rest bei der Durchführung einer ganzzahligen Division interessiert, so kann dieser mit dem Operator `%` ermittelt werden. Dieser Operator wird *Modulo-Operator* genannt.

```
>>> 7%5
2
>>> 22%4
2
```

**Merke**

In Python 2.x – wie auch in anderen Programmiersprachen (z. B. C/C++) – muss beachtet werden, dass bei der Division von ganzen Zahlen eine Ganzzahl-Division durchgeführt wird. Ist jedoch nur einer der Operanden eine Dezimalzahl, so wird eine Gleitpunkt-Division ausgeführt. In der neueren Sprachversion Python 3.x, die wir in diesem Buch benutzen, wurde dies abgeschafft. Dort wird jede Division, die mit dem Operator / ausgeführt wird, als Gleitpunkt-Division durchgeführt. Der Programmierer kann jedoch auch eine Ganzzahl-Division durch die Anwendung eines besonderen Operators (// statt /) ausführen lassen.

Das folgende Beispiel zeigt, dass sogenannte rein periodische Zahlen natürlich nur mit einer endlichen Anzahl von Nachkommastellen dargestellt werden können.

```
>>> 32/3
10.666666666666666
```

Der Digitalrechner kann *nicht jede Zahl* exakt darstellen. Aufgrund der internen Darstellung in Form von binären Zuständen (jede Zahl wird intern durch eine Folge von Nullen und Einsen aufgebaut) können nicht alle Zahlen völlig präzise dargestellt werden. Dieser Sachverhalt spielt eine wichtige Rolle beim numerischen Rechnen. In unserem Fall ist die Abweichung klein und belanglos. Werden jedoch sehr viele arithmetische Operationen durchgeführt, so kann sich ein erheblicher Gesamtfehler akkumulieren.

### 1.3.3 Vergleichsausdrücke

Fahren wir mit unserer Erkundungstour durch Python fort. Von Handrechnungen kennen wir die Vergleichsoperatoren < und >. Wir probieren sie aus:

```
>>> 2<7
True
>>> 2>7
False
```

Vergleichsausdrücke werden von Python als wahr (engl. true) und falsch (engl. false) ausgewertet.

```
>>> 2==7
False
>>> 2!=7
True
>>> 5==5
True
```



Beim Testen auf Gleichheit wird kein Gleichheitszeichen geschrieben, sondern *zwei aufeinanderfolgende Gleichheitszeichen*. Beim Test auf Ungleichheit wird der Operator „!“ verwendet. Dieser Operator wird auch in den populären Programmiersprachen C und C++ verwendet. Python entlehnt viele Sprachelemente aus diesen Sprachen.

```
>>> 3<3
False
>>> 3<=3
True
```

Das letzte Beispiel zeigt uns den Unterschied zwischen „<“ und „<=“. Probieren Sie selbst ein Beispiel mit „>“ und „>=“ aus!

### 1.3.4 Logische Ausdrücke

Ausdrücke, die wahr oder falsch sind, können zu komplexeren Ausdrücken mithilfe der logischen Operatoren *and*, *or* und *not* zusammengefügt werden. Hier einige Beispiele für solche Ausdrücke, die auch boolesche (engl. boolean) Ausdrücke genannt werden.

```
>>> not(5==5)
False
>>> (2<7) and (5>4)
True
>>> (2>7) or (4>5)
False
>>> not(3!=3)
True
```

Ein Ausdruck der mit dem Und-Operator gebildet wird, ist dann und nur dann wahr, wenn beide Teilausdrücke bzw. Operanden wahr sind. Umgekehrt ist ein mit dem Oder-Operator gebildeter Ausdruck schon dann wahr, wenn nur ein einziger Operand wahr ist.

### 1.3.5 Mathematische Funktionen

Wir wollen mathematische Funktionen anwenden und probieren Folgendes aus:

```
>>> sin(90)

Traceback (most recent call last):
  File "<pyshell#32>", line 1, in <module>
    sin(90)
NameError: name 'sin' is not defined
```

Unsere Absicht war es, den Sinuswert von 90 (Grad) auszurechnen. Nun werden wir zum ersten Mal mit einer Fehlermeldung von Python konfrontiert. Diese ist auch hinreichend klar in englischer Sprache formuliert: „name 'sin' is not defined“. Dies bedeutet übersetzt etwa: „der Name ‚sin‘ ist nicht definiert“. Sollte Python keine mathematischen Standardfunktionen bereitstellen? Dann wäre ja jeder Taschenrechner „intelligenter“. Tatsächlich ist es so, dass wir die mathematischen Funktionen zunächst laden müssen, bevor wir diese anwenden können. Dies geschieht durch eine import-Anweisung. Mithilfe dieser Anweisung wird ein sogenanntes „Modul“ geladen. Was dies genau bedeutet, werden wir im Kapitel 3 genauer verstehen lernen.

```
>>> from math import *
>>> sin(90)
0.89399666360055785
>>> sin(180)
-0.80115263573383044
```

Nachdem wir die mathematischen Funktionen aus dem Modul *math* importiert haben, können wir die Sinusfunktion problemlos aufrufen. Dies gilt auch für andere mathematische Standardfunktionen. Probieren Sie einfach mal den Kosinus (cos) aus!

Allerdings wird uns auch klar, dass der Zahlenwert, der dieser Funktion übergeben wird, *nicht* als Winkel interpretiert wird. Sonst müssten im vorigen Beispiel die Werte 1 und 0 herauskommen. Wir vermuten, dass das Argument *im Bogenmaß* einzugeben ist und wandeln das Beispiel etwas ab.

```
>>> pi
3.141592653589793
>>> sin(pi/2)
1.0
>>> sin(pi)
1.2246467991473532e-16
```

Der math-Modul muss nur einmal pro Sitzung importiert werden. In diesem Modul ist eine Variable mit Namen pi definiert. Wir berechnen den Sinuswert von pi/2 (entsprechend 90 Grad) und von pi (entsprechend 180 Grad). Die letzte Zeile im letzten Codeabschnitt ist auf den ersten Blick verwirrend. Es sollte eigentlich genau 0 herauskommen. Stattdessen erhalten wir 1.2246467991473532e-16. Dies bedeutet in der Schreibweise für *Dezimalzahlen*: 1,2246467991473532  $10^{-16}$ .

Dies ist eine sehr kleine Zahl, fast null. Damit haben wir auch eine zweite Darstellungsform für Gleitpunktzahlen kennen gelernt: die *Exponentendarstellung*.



### Merke

Die mathematischen Standardfunktionen Sinus, Cosinus, Tangens etc. werden mit dem Bogenmaß als Argument aufgerufen. Sie müssen einen Winkel also zuerst ins Bogenmaß umrechnen, um dann den Wert der Funktion zu ermitteln.

### 1.3.6 Grundlegendes über Variablen und Zuweisungen

Wir wollen nun das wichtige Konzept der *Variablen* kennenlernen. Betrachten Sie dazu den folgenden Programmcode:

```
>>> a

Traceback (most recent call last):
  File "<pysshell#44>", line 1, in <module>
    a
NameError: name 'a' is not defined
>>> a = 7
>>> b = 3
>>> c = a*b
>>> print("c = ",c)
c = 21
>>> a
7
```

Zunächst geben wir den Buchstaben `a` ein. Python antwortet daraufhin mit der Fehlermeldung, dass der Name `a` nicht definiert ist.

Dies wird mit der folgenden Zeile durch die Anweisung `a = 7` nachgeholt. Bei dieser Anweisung handelt es sich um eine sogenannte *Zuweisung*. Es wird ein Name `a` erklärt und diesem Namen – genannt Variable – wird ein konstanter Wert `7` zugewiesen. Wir wollen uns vorläufig vorstellen, dass die Variable `a` von nun an stellvertretend für die Zahl `7` steht. Genauer gesagt, bildet die Variable einen „Behälter“, der den Wert `7` aufgenommen hat. Etwas Vergleichbares geschieht in den folgenden beiden Zeilen. Dort wird zunächst eine Variable `b` erklärt und erhält den Wert `3` zugewiesen. Schließlich wird eine Variable mit Namen `c` erzeugt. Diese Variable beinhaltet das Ergebnis der Multiplikation von `a` mit `b`. Damit dieser Wert am Bildschirm ausgegeben wird, verwenden wir eine eingebaute Funktion von Python: die `print()`-Funktion. Diese Funktion gibt den Text „`c =` “ in der Python-Shell aus, gefolgt von dem Inhalt der Variablen `c`. Zusammen wird das so geschrieben:

```
print("c = ",c)
```

Schließlich sehen wir an den letzten beiden Zeilen, dass der Name `a` nun bekannt ist, nachdem ihm ein Wert zugewiesen wurde. Gibt man einfach nur den Buchstaben `a` ein, antwortet Python mit dem Inhalt der Variable, d. h. mit dem Wert, der dieser Variablen zugewiesen wurde. Ebenso hätten wir den Inhalt von `c` ausgeben können.

Variablen (sie werden auch Bezeichner) genannt, können beliebig viele Zeichen umfassen. Die Regeln zur Bildung solcher Namen sind in Kurzform:

- Namen für Variablen können aus Buchstaben, Ziffern (0..9) und einem einzigen Sonderzeichen bestehen. Dieses Sonderzeichen ist der Unterstrich „`_`“ (engl. underscore).

- Das erste Zeichen in einem Variablennamen darf *keine Ziffer* sein. Ein Unterstrich ist allerdings als erstes Zeichen erlaubt.
- Deutsche Umlaute (Ä, ä usw.) dürfen in Python 3.x verwendet werden. In Python 2.x sind diese Zeichen *nicht* erlaubt.
- Es wird zwischen Groß- und Kleinschreibung unterschieden.
- Variablennamen dürfen *nicht* mit den reservierten Worten der Programmiersprache Python übereinstimmen.

Es wird empfohlen, möglichst selbsterklärende Namen für die Variablen zu erfinden, die einen Zusammenhang mit der durch das Programm zu lösenden Problematik schon im Namen ausdrücken. Also möglichst nicht „a“, „b“ und „c“ wie im vorangegangenen Beispiel, sondern „Zylinder\_Durchmesser“, „StartZeit“, „Gesamt\_Summe“ usw. Damit werden die Programme verständlicher und besser lesbar.

### 1.3.7 Zeichenketten (Strings)

Auch im technisch-wissenschaftlichen Bereich müssen oft Programme geschrieben werden, die Texte verarbeiten. Eine sogenannte Zeichenkette (engl. string) besteht aus beliebigen Zeichen und wird in Anführungszeichen gesetzt. Das folgende Beispiel zeigt, dass auch für solche Zeichenketten der „+“-Operator existiert. Dieser ist also *kontextsensitiv* und führt etwas anderes durch, wenn seine Operanden Zeichenketten statt Zahlen sind. In diesem Fall werden die Zeichenketten aneinander gehängt (engl. concatenation). In einem String dürfen Umlaute auch verwendet werden.

```
>>> Erster_Name="Bergische "  
>>> Zweiter_Name="Universität"  
>>> Name = Erster_Name + Zweiter_Name  
>>> print(Name)  
Bergische Universität
```

### 1.3.8 Turtle-Grafik

Python beinhaltet ein Modul zur Erstellung von einfachen Liniengrafiken. Die Programmierung funktioniert nach dem „Schildkrötenprinzip“, d. h. der Programmierer steuert ein Symbol – Schildkröte genannt – mit einfachen Befehlen innerhalb eines Grafikensters. Durch die Bewegung der Schildkröte (engl. turtle) wird dann die Grafik erzeugt. Die Turtle-Grafik ist ein gutes Hilfsmittel zum Erlernen der Programmierung. Wir werden allerdings im Rahmen dieses Buches auch anspruchsvollere Grafiken mit diesem Tool erzeugen.

Im Prinzip genügen zunächst elf Befehle (besser gesagt: Funktionsaufrufe), um mit der Schildkröte erste interessante Grafiken erzeugen zu können.

- `forward(steps)`: Mit diesem Befehl kriecht die Schildkröte in ihrer Blickrichtung vorwärts. Sie geht dabei um so viele Schritte bzw. *Pixel* nach vorn, wie der Parameter *steps* vorgibt. `fd(steps)` ist die Abkürzung dieses Befehls.
- `left(angle)`: Mit diesem Befehl ändert die Schildkröte ihre Richtung. Sie dreht sich um den Winkel *angle* nach links. Der Winkel wird dabei in *Grad* angegeben. `lt(angle)` ist eine Abkürzung des Befehls „left“.
- `right(angle)`: Mit diesem Befehl ändert die Schildkröte ihre Richtung. Sie dreht sich um den Winkel *angle* nach rechts. Der Winkel wird dabei auch in *Grad* angegeben. `rt(angle)` ist eine Abkürzung dieses Befehls.
- `pencolor(col)`: Mit diesem Befehl kann die Farbe der Linien, welche die Schildkröte zeichnet, beeinflusst werden. Die Variable *col* kann die Werte „red“, „green“, „blue“, „yellow“ etc. annehmen. Statt der Anführungszeichen können auch Apostrophe verwendet werden (z. B. `color('red')`).
- `pensize(pixel)`: Mit diesem Befehl wird die Dicke der zu zeichnenden Linien in Pixel gesteuert.
- `penup()` und `pendown()` hebt und senkt den „Zeichenstift“ der Schildkröte.
- `reset()`: Dieser Befehl löscht den Inhalt des Grafikfensters und setzt die Schildkröte auf ihren Anfangszustand zurück (Ausrichtung nach rechts).
- `goto(x,y)`: Mit diesem Befehl springt die Schildkröte auf die Position (x,y), wobei x und y in Pixel gemessen werden. Das Bezugskoordinatensystem befindet sich in der oberen rechten Ecke des Zeichenfensters. Die x-Achse ist horizontal ausgerichtet und die y-Achse zeigt senkrecht nach unten.
- `bye()`: Mit diesem Befehl wird das Turtle-Grafikfenster geschlossen.
- `exitonclick()`: Mit diesem Befehl wird das Grafikfenster geschlossen, wenn der Benutzer mit der Maus in das Fenster klickt.



#### Merke

Am Ende einer Turtle-Befehlsfolge sollte der Befehl `bye()` ausgeführt werden, sonst „hängt“ das Turtle-Grafikfenster unter dem Betriebssystem.

Am Anfang – dies ist so vereinbart – steht die Schildkröte mit Blickrichtung nach rechts. Wir wollen dies nun erproben. Hierzu muss zuerst das Modul „turtle“ importiert werden, damit die obigen Befehle zur Verfügung stehen. Dabei bedeutet das Sternchen \*, dass *alle* Befehle aus dem Modul *turtle* importiert werden.

```
>>> from turtle import *
>>> reset()
>>> forward(100)
>>> left(90)
>>> forward(70)
>>> left(90)
>>> forward(100)
>>> left(90)
>>> forward(70)
```



Im Bild ist *links* der Ursprungszustand der Schildkröte zu sehen, nachdem die ersten beiden Zeilen ausgeführt wurden. Die Schildkröte erscheint auf dem Bildschirm als *Pfeilspitze* und ist in horizontaler Richtung ausgerichtet.

Mit dem Befehl `forward(100)` kriecht die Schildkröte um hundert Pixel nach rechts und erzeugt dabei eine horizontale Linie. Daraufhin wird mit dem Befehl `left(90)` die Schildkröte um 90 Grad nach links gedreht. Sie blickt jetzt nach oben. Im nächsten Schritt geht die Schildkröte dann um 70 Schritte vorwärts. Es entsteht eine vertikale Linie mit einer Länge von 70 Pixeln. Der nächste Befehl dreht die Schildkröte wieder um 90 Grad nach links. Sie steht nun wieder horizontal; aber nach links ausgerichtet usw.

Rechts ist das Endergebnis zu sehen. Es wurde ein Rechteck gezeichnet. Zum Schluss behält die Schildkröte ihre Ausrichtung bei. Sie blickt jetzt nach unten. Zum Abschluss des Dialogs geben wir `bye()` ein, um das Grafikenster wieder zu schließen:

```
>>> bye()
```

Jeder Befehl benötigt einige Zeit zu seiner Ausführung. Die Schildkröte bewegt sich deshalb so langsam über das Grafikenster, damit der Programmierer genau die Auswirkungen seiner Befehle beobachten kann.

## ■ 1.4 Python-Programme mit IDLE erstellen

Alle Programmieranweisungen, die wir bis jetzt eingegeben haben, wurden Zeile für Zeile in die „Python-Shell“ eingegeben und dann von Python unmittelbar ausgeführt. Dies ist sehr bequem und Programmieranfänger können damit auf einfache Art ihre ersten Gehversuche machen.