

Python for Scientists

Python is a free, open source, easy-to-use software tool that offers a significant alternative to proprietary packages such as Matlab and Mathematica. This book covers everything the working scientist needs to know to start using Python effectively.

The author explains scientific Python from scratch, showing how easy it is to implement and test non-trivial mathematical algorithms and guiding the reader through the many freely available add-on modules. A range of examples, relevant to many different fields, illustrate the program's capabilities. In particular, readers are shown how to use pre-existing legacy code (usually in Fortran77) within the Python environment, thus avoiding the need to master the original code.

Instead of exercises the book contains useful snippets of tested code which the reader can adapt to handle problems in their own field, allowing students and researchers with little computer expertise to get up and running as soon as possible.

Cambridge University Press
978-1-107-06139-2 - Python for Scientists
John M. Stewart
Frontmatter
[More information](#)

Cambridge University Press
978-1-107-06139-2 - Python for Scientists
John M. Stewart
Frontmatter
[More information](#)

Python for Scientists

JOHN M. STEWART
Department of Applied Mathematics & Theoretical Physics
University of Cambridge



CAMBRIDGE
UNIVERSITY PRESS

University Printing House, Cambridge CB2 8BS, United Kingdom

Cambridge University Press is part of the University of Cambridge.

It furthers the University’s mission by disseminating knowledge in the pursuit of education, learning and research at the highest international levels of excellence.

www.cambridge.org
Information on this title: www.cambridge.org/9781107061392

© John M. Stewart 2014

This publication is in copyright. Subject to statutory exception and to the provisions of relevant collective licensing agreements, no reproduction of any part may take place without the written permission of Cambridge University Press.

First published 2014
Reprinted 2015

Printed in the United Kingdom by Clays, St Ives plc

A catalogue record for this publication is available from the British Library

Library of Congress Cataloguing in Publication data

Stewart, John, 1943 July 1–
Python for scientists / John M. Stewart.
pages cm

Includes bibliographical references and index.
ISBN 978-1-107-06139-2 (hardback)

1. Science – Data processing. 2. Python (Computer program language) I. Title.
Q183.9.S865 2014
005.13’3 – dc23 2014010571

ISBN 978-1-107-06139-2 Hardback
ISBN 978-1-107-68642-7 Paperback

Cambridge University Press has no responsibility for the persistence or accuracy of URLs for external or third-party internet websites referred to in this publication, and does not guarantee that any content on such websites is, or will remain, accurate or appropriate.

Contents

	<i>Preface</i>	<i>page xi</i>
1	Introduction	1
	1.1 Scientific software	1
	1.2 The plan of this book	4
	1.3 Can Python compete with compiled languages?	8
	1.4 Limitations of this book	9
	1.5 Installing Python and add-ons	9
2	Getting started with IPython	11
	2.1 Generalities	11
	2.2 Tab completion	12
	2.3 Introspection	12
	2.4 History	13
	2.5 Magic commands	14
	2.6 The magic %run command	15
3	A short Python tutorial	20
	3.1 Typing Python	20
	3.2 Objects and identifiers	21
	3.3 Numbers	23
	3.3.1 Integers	23
	3.3.2 Real numbers	23
	3.3.3 Boolean numbers	24
	3.3.4 Complex numbers	25
	3.4 Namespaces and modules	25
	3.5 Container objects	27
	3.5.1 Lists	27
	3.5.2 List indexing	28
	3.5.3 List slicing	28
	3.5.4 List mutability	30
	3.5.5 Tuples	31
	3.5.6 Strings	31
	3.5.7 Dictionaries	32

3.6	Python <i>if</i> statements	32
3.7	Loop constructs	33
3.7.1	The Python <i>for</i> loop	34
3.7.2	The Python <i>continue</i> statement	35
3.7.3	The Python <i>break</i> statement	35
3.7.4	List comprehensions	36
3.7.5	Python <i>while</i> loops	37
3.8	Functions	37
3.8.1	Syntax and scope	38
3.8.2	Positional arguments	41
3.8.3	Keyword arguments	41
3.8.4	Variable number of positional arguments	41
3.8.5	Variable number of keyword arguments	42
3.8.6	The Python <i>print</i> function	42
3.8.7	Anonymous functions	44
3.9	Introduction to Python classes	45
3.10	The structure of Python	47
3.11	Prime numbers: a worked example	48
4	Numpy	52
4.1	One-dimensional arrays	54
4.1.1	Ab initio constructors	54
4.1.2	Look alike constructors	55
4.1.3	Arithmetical operations on vectors	56
4.1.4	Ufuncs	58
4.1.5	Logical operations on vectors	59
4.2	Two-dimensional arrays	62
4.2.1	Broadcasting	63
4.2.2	Ab initio constructors	64
4.2.3	Look alike constructors	65
4.2.4	Operations on arrays and ufuncs	66
4.3	Higher-dimensional arrays	66
4.4	Domestic input and output	67
4.4.1	Discursive output and input	67
4.4.2	Numpy text output and input	69
4.4.3	Numpy binary output and input	69
4.5	Foreign input and output	70
4.5.1	Small amounts of data	70
4.5.2	Large amounts of data	71
4.6	Miscellaneous ufuncs	71
4.6.1	Maxima and minima	71
4.6.2	Sums and products	72
4.6.3	Simple statistics	72
4.7	Polynomials	73

	Contents	vii
	4.7.1 Converting data to coefficients	73
	4.7.2 Converting coefficients to data	73
	4.7.3 Manipulating polynomials in coefficient form	74
4.8	Linear algebra	74
	4.8.1 Basic operations on matrices	74
	4.8.2 More specialized operations on matrices	75
	4.8.3 Solving linear systems of equations	75
4.9	More <i>numpy</i> and beyond	76
	4.9.1 <i>Scipy</i>	76
	4.9.2 <i>Scikits</i>	77
5	Two-dimensional graphics	79
5.1	Introduction	79
5.2	Getting started: simple figures	80
	5.2.1 Front-ends	80
	5.2.2 Back-ends	80
	5.2.3 A simple figure	81
	5.2.4 Interactive controls	82
5.3	Cartesian plots	83
	5.3.1 The <i>matplotlib</i> <code>plot</code> function	83
	5.3.2 Curve styles	83
	5.3.3 Marker styles	84
	5.3.4 Axes, grid, labels and title	85
	5.3.5 A not-so-simple example: partial sums of Fourier series	86
5.4	Polar plots	87
5.5	Error bars	88
5.6	Text and annotations	89
5.7	Displaying mathematical formulae	90
	5.7.1 Non- <i>LaTeX</i> users	90
	5.7.2 <i>LaTeX</i> users	91
	5.7.3 Alternatives for <i>LaTeX</i> users	92
5.8	Contour plots	92
5.9	Compound figures	95
	5.9.1 Multiple figures	95
	5.9.2 Multiple plots	96
5.10	Animations	98
	5.10.1 In situ animations	99
	5.10.2 Movies	100
5.11	Mandelbrot sets: a worked example	102
6	Three-dimensional graphics	107
6.1	Introduction	107
	6.1.1 Three-dimensional data sets	107
	6.1.2 The reduction to two dimensions	108

viii	Contents	
6.2	Visualization software	109
6.3	A three-dimensional curve	110
6.3.1	Visualizing the curve with <i>mplot3d</i>	111
6.3.2	Visualizing the curve with <i>mlab</i>	112
6.4	A simple surface	114
6.4.1	Visualizing the simple surface with <i>mplot3d</i>	114
6.4.2	Visualizing the simple surface with <i>mlab</i>	116
6.5	A parametrically defined surface	117
6.5.1	Visualizing Enneper’s surface using <i>mplot3d</i>	117
6.5.2	Visualizing Enneper’s surface using <i>mlab</i>	119
6.6	Three-dimensional visualization of a Julia set	120
7	Ordinary differential equations	122
7.1	Initial value problems	122
7.2	Basic concepts	122
7.3	The <code>odeint</code> function	125
7.3.1	Theoretical background	125
7.3.2	Practical usage	127
7.4	Two-point boundary value problems	132
7.4.1	Introduction	132
7.4.2	Formulation of the boundary value problem	133
7.4.3	A simple example	135
7.4.4	A linear eigenvalue problem	136
7.4.5	A non-linear boundary value problem	138
7.5	Delay differential equations	142
7.5.1	A model equation	143
7.5.2	More general equations and their numerical solution	144
7.5.3	The logistic equation	145
7.5.4	The Mackey–Glass equation	147
7.6	Stochastic differential equations	150
7.6.1	The Wiener process	150
7.6.2	The Itô calculus	152
7.6.3	Itô and Stratanovich stochastic integrals	155
7.6.4	Numerical solution of stochastic differential equations	156
8	Partial differential equations: a pseudospectral approach	163
8.1	Initial-boundary value problems	163
8.2	Method of lines	164
8.3	Spatial derivatives via finite differencing	164
8.4	Spatial derivatives by spectral techniques for periodic problems	165
8.5	The IVP for spatially periodic problems	167
8.6	Spectral techniques for non-periodic problems	170
8.7	An introduction to <code>f2py</code>	172
8.7.1	Simple examples with scalar arguments	172

8.7.2	Vector arguments	174
8.7.3	A simple example with multi-dimensional arguments	175
8.7.4	Undiscussed features of <code>f2py</code>	176
8.8	A real-life <code>f2py</code> example	177
8.9	Worked example: Burgers' equation	178
8.9.1	Boundary conditions: the traditional approach	179
8.9.2	Boundary conditions: the penalty approach	179
9	Case study: multigrid	184
9.1	The one-dimensional case	185
9.1.1	Linear elliptic equations	185
9.1.2	Smooth and rough modes	186
9.2	The tools of multigrid	186
9.2.1	Relaxation methods	186
9.2.2	Residual and error	189
9.2.3	Prolongation and restriction	190
9.3	Multigrid schemes	191
9.3.1	The two-grid algorithm	192
9.3.2	The V-cycle scheme	193
9.3.3	The full multigrid scheme (FMG)	194
9.4	A simple Python multigrid implementation	195
9.4.1	Utility functions	196
9.4.2	Smoothing functions	197
9.4.3	Multigrid functions	199
Appendix A	Installing a Python environment	205
A.1	Installing Python packages	205
A.2	Communicating with Python	206
A.2.1	Editors for programming	206
A.2.2	The <i>IPython</i> -editor interaction	207
A.2.3	The two windows approach	207
A.2.4	Calling the editor from within <i>IPython</i>	208
A.2.5	Calling <i>IPython</i> from within the editor	208
A.2.6	The <i>IPython</i> pager	208
A.3	The Python Package Index	209
Appendix B	Fortran77 subroutines for pseudospectral methods	210
	<i>References</i>	216
	<i>Index</i>	218

Cambridge University Press
978-1-107-06139-2 - Python for Scientists
John M. Stewart
Frontmatter
[More information](#)

Preface

I have used computers as an aid to scientific research for over 40 years. During that time, hardware has become cheap, fast and powerful. However, software relevant to the working scientist has become progressively more complicated. My favourite textbooks on Fortran90 and C++ run to 1200 and 1600 pages respectively. And then we need documentation on mathematics libraries and graphics packages. A newcomer going down this route is going to have to invest significant amounts of time and energy in order to write useful programmes. This has led to the emergence of “scientific packages” such as Matlab or Mathematica which avoid the complications of compiled languages, separate mathematics libraries and graphics packages. I have used them and found them very convenient for executing the tasks envisaged by their developers. However, I also found them very difficult to extend beyond these boundaries, and so I looked for alternative approaches.

Some years ago, a computer science colleague suggested that I should take a look at Python. At that time, it was clear that Python had great potential but a very flaky implementation. It was however free and open-source, and was attracting what has turned out to be a very effective army of developers. More recently, their efforts have coordinated to produce a formidable package consisting of a small core language surrounded by a wealth of add-on libraries or *modules*. A select group of these can and do replicate the facilities of the conventional scientific packages. More importantly an informed, intelligent user of Python and its modules can carry out major projects usually entrusted to dedicated programmers using Fortran, C etc. There is a marginal loss of execution speed, but this is more than compensated for by the vastly telescoped development time. The purpose of this book is to explain to working scientists the utility of this relatively unknown resource.

Most scientists will have some computer familiarity and programming awareness, although not necessarily with Python and I shall take advantage of this. Therefore, unlike many books which set out to “teach” a language, this one is not just a brisk trot through the reference manuals. Python has many powerful but unfamiliar facets, and these need more explanation than the familiar ones. In particular, if you encounter in this text a reference to the “beginner” or the “unwary”, it signifies a point which is not made clear in the documentation, and has caught out this author at least once.

The first six chapters, plus Appendix A, cover almost everything the working scientist needs to know in order to get started in using Python effectively. My editor and some referees suggested that I should devote the second half of the book to problems in

a particular field. This would have led to a series of books, “Python for Biochemists”, “Python for Crystallographers”, . . . , all with a common first half. Instead I have chosen to cover just three topics, which however, should be far more widely applicable in many different fields. Chapter 7 covers four radically different types of ordinary differential equations and shows how to use the various relevant black boxes, which are often Python wrappers around tried and trusted Fortran codes. The next chapter while ostensibly about pseudospectral approaches to evolutionary partial differential equations, actually covers a topic of great utility to many scientists, namely how to reuse legacy code, usually written in Fortran⁷⁷ within Python at Fortran-like speeds, without understanding Fortran. The final chapter about solving very large linear systems via multigrid is also a case history in how to use object-oriented programming meaningfully in a scientific context. If readers look carefully and critically at these later chapters, they should gain the practical expertise to handle problems in their own field.

Acknowledgments are due to the many Python developers who have produced and documented a very useful tool, and also to the very many who have published code snippets on the web, a great aid to the tyro, such as this author. Many of my colleagues have offered valuable advice. Des Higham generously consented to my borrowing his ideas for the last quarter of Chapter 7. I am especially grateful to Oliver Rinne who read carefully and critically an early draft. At Cambridge University Press, my Production Editor, Jennifer Murphy and my Copy Editor, Anne Rix have exhibited their customary expertise. Last but not least I thank the Department of Applied Mathematics and Theoretical Physics, Cambridge for continuing to offer me office space after my retirement, which has facilitated the production of this book.

Writing a serious book is not a trivial task and so I am rather more than deeply grateful for the near-infinite patience of *Mary*, the “Python-widow”, which made this book possible!