

Contents

Preface

v

Chapter 1. Getting Started

1

1.1	Integrated development environment	1
1.2	Jupyter notebook	2
1.3	Python distribution	2
1.4	Python console	2
1.5	Python as a calculator	3

Chapter 2. Python Basics

5

2.1	Exponentiation	5
2.2	Order of operations and parentheses	5
2.3	Integer division	6
2.4	NumPy	7
2.5	Strings	8
2.6	Variables and the = operator	8
2.7	Projectile motion	10
2.8	User input	12
2.9	Output formatting	13
2.10	Lists	14
2.11	Line continuation and indentation	14
2.12	Order of execution	15

Chapter 3.	Control Structures	17
3.1	The <code>for</code> loop	17
3.2	The <code>range()</code> function	19
3.3	More about <code>range()</code>	20
3.4	Conditions	22
3.5	<code>if, elif, else</code>	23
3.6	<code>and, or, not</code>	24
3.7	<code>continue</code> and <code>break</code>	25
3.8	More exercises	28
Chapter 4.	Libraries, Arrays, and Plots	29
4.1	Importing libraries	29
4.2	NumPy arrays	30
4.3	Creating NumPy arrays	32
4.4	Machine roundoff error	33
4.5	Plotting graphs	35
Chapter 5.	Indexing Lists and Arrays	39
5.1	NumPy and Matplotlib	39
5.2	Lists and arrays	39
5.3	Indexing	40
5.4	Accessing elements	41
5.5	Using arrays	42
5.6	Counters	44
5.7	Appending to an array	45
5.8	Appending to a list	46
5.9	More exercises	47
Chapter 6.	Data Types and Variable Assignment	49
6.1	Python data types	49
6.2	NumPy data types	50
6.3	Lists of lists	52
6.4	Type conversion	53
6.5	Variable assignment for numbers	54
6.6	Variable assignment for lists and arrays	56

6.7	Copying arrays and lists	58
6.8	Cellular automata	59
Chapter 7.	Functions and More Loops	63
7.1	Function definitions	63
7.2	Functions and variables	65
7.3	Variables and parameters	68
7.4	Beyond functions	71
7.5	More <code>for</code> loops	73
7.6	<code>while</code> loops	74
7.7	More exercises	77
Chapter 8.	Random Topics	79
8.1	Subplots, legends and <code>Mathtext</code>	79
8.2	Data input and output	82
8.3	Random numbers	85
8.4	Real number formats	87
8.5	Negative indexing	88
8.6	Object oriented programming	89
Chapter 9.	Programming Practice	91
9.1	Building complex programs	91
9.2	Roulette	94
9.3	More cellular automata	96
9.4	Mandelbrot set	98
9.5	Colored beads	102
Chapter 10.	Symbolic Computation with SymPy	103
10.1	Basic SymPy commands	103
10.2	Calculus with SymPy	109
10.3	Plotting graphs with SymPy	113
10.4	Linear algebra with SymPy	114
10.5	From SymPy to NumPy	117
10.6	Printing with SymPy	118

Chapter 11.	Root Finding	119
11.1	Motion with air resistance	119
11.2	The general problem	120
11.3	The bisection method	121
11.4	Newton's method	122
11.5	Execution time	125
11.6	Multiple roots	126
11.7	SciPy optimize	127
Chapter 12.	Curve Fitting and Interpolation	131
12.1	Expanding Universe	131
12.2	Least squares method	133
12.3	Best-fit line	134
12.4	Best-fit curve	135
12.5	Data interpolation	138
12.6	Cubic splines	139
12.7	More data points	141
12.8	Bilinear interpolation	143
Chapter 13.	Numerical Integration I	147
13.1	A simple example	147
13.2	Left endpoint rule	148
13.3	Right endpoint and midpoint rules	150
13.4	Errors and convergence tests	152
13.5	Estimating errors and significant figures	154
Chapter 14.	Numerical Integration II	157
14.1	Takeaways from before	157
14.2	Error analysis	158
14.3	Trapezoid rule	160
14.4	Simpson's rule	163
14.5	Integration with SciPy	165
14.6	Nonuniform data	167
14.7	Monte Carlo integration	168
14.8	Multidimensional integrals	170
14.9	More Exercises	172

Chapter 15.	Linear Algebra	173
15.1	Matrix multiplication	173
15.2	Linear systems	174
15.3	The <code>linalg.solve()</code> function	176
15.4	Statics	178
15.5	Kirchoff's laws	180
15.6	Eigenvalues and eigenvectors	181
15.7	Normal mode analysis	183
Chapter 16.	Numerical Differentiation	189
16.1	Two-point difference formulas	189
16.2	Truncation errors and machine errors	192
16.3	Two-point forward difference, again	194
16.4	Centered difference formulas	195
16.5	Other stencils	198
Chapter 17.	Ordinary Differential Equations I	201
17.1	Newton's law of cooling	201
17.2	Euler's method	202
17.3	Truncation error	205
17.4	Error estimation	206
17.5	Three-point convergence test	209
17.6	Richardson extrapolation	211
17.7	Practical issues with error estimation	214
Chapter 18.	Ordinary Differential Equations II	217
18.1	Systems of equations	217
18.2	Euler's method in general	218
18.3	Second-order equations	219
18.4	Second-order Runge–Kutta	220
18.5	Fourth-order Runge–Kutta	222
18.6	<code>solve_ivp()</code>	224
18.7	ODE solvers with <code>solve_ivp()</code>	226

Chapter 19.	Driven Damped Pendulum and Chaos	229
19.1	Equation of motion	229
19.2	Long-term motion	231
19.3	Code tests	232
19.4	State space	234
19.5	Period doubling	235
19.6	Chaos	238
19.7	Sensitivity to initial conditions	240
Chapter 20.	Boundary Value Problems	245
20.1	The hanging cord	245
20.2	Differential equation for the cord	245
20.3	Relaxation	248
20.4	Gauss–Seidel and Jacobi methods	250
20.5	Speeding up your code	251
20.6	Residual	253
20.7	Length of the cord	254
20.8	Euler–Bernoulli beam theory	255
Chapter 21.	Partial Differential Equations I	259
21.1	Waves on a string	259
21.2	CTCS algorithm	261
21.3	Numerical instability	266
21.4	An unphysical solution	269
Chapter 22.	Partial Differential Equations II	271
22.1	First-order time derivatives	271
22.2	Lax–Friedrich method	273
22.3	Energy of a vibrating string	274
22.4	Method of lines	275
22.5	Other initial conditions	278
22.6	Other boundary conditions	279
22.7	Other PDEs	280
22.8	Animation	282
22.9	First-order form of the wave equation	284

Chapter 23. Fourier Analysis	287
23.1 Fourier series	287
23.2 Discrete Fourier transform	290
23.3 Complex form of the DFT	293
23.4 Fast Fourier transform	298
23.5 Nyquist frequency and aliasing	299
23.6 More Exercises	300
Appendix A. Data Files	303
<i>Index</i>	307

Scientific computing covers a wide range of topics. This book provides an introduction to some of the basics: integration, differentiation, root finding, linear systems, eigenvalue problems, least squares fitting, ordinary and partial differential equations, interpolation, Fourier analysis, visualization, and symbolic computation. Scientific computing gives you the power to analyze and explore mathematical models of complex physical systems.

The algorithms discussed in this book can be implemented in almost any computer language. We will use Python because Python is powerful, easy to learn, well-supported, and free.

The first task is to get Python running. You can download the latest Python language from *Python.org*, but this is not the best option. Most scientists and engineers access Python in one or both of the following ways: through an *integrated development environment* or in a *Jupyter notebook*.

1.1 Integrated development environment

An *integrated development environment* (IDE) is a computer application that provides tools for software development. Some IDEs are designed specifically for Python, others are multi-language. Some IDEs are free, others are not.

A Python program (or code) consists of one or more text files containing Python language statements. An IDE will include a text editor for creating these text files. The text editor will have helpful