

The book serves as a first introduction to computer programming of scientific applications, using the high-level Python language. The exposition is example- and problem-oriented, where the applications are taken from mathematics, numerical calculus, statistics, physics, biology and finance. The book teaches “Matlab-style” and procedural programming as well as object-oriented programming. High school mathematics is a required background, and it is advantageous to study classical and numerical one-variable calculus in parallel with reading this book. Besides learning how to program computers, the reader will also learn how to solve mathematical problems, arising in various branches of science and engineering, with the aid of numerical methods and programming. By blending programming, mathematics and scientific applications, the book lays a solid foundation for practicing computational science.

TEXTS IN COMPUTATIONAL SCIENCE AND ENGINEERING

From the reviews:

Langtangen ... does an excellent job of introducing programming as a set of skills in problem solving. He guides the reader into thinking properly about producing program logic and data structures for modeling real-world problems using objects and functions and embracing the object-oriented paradigm. ... Summing Up: Highly recommended.

F. H. Wild III, Choice, Vol. 47 (8), April 2010

Those of us who have learned scientific programming in Python ‘on the streets’ could be a little jealous of students who have the opportunity to take a course out of Langtangen’s Primer.

John D. Cook, The Mathematical Association of America, September 2011

This book goes through Python in particular, and programming in general, via tasks that scientists will likely perform. It contains valuable information for students new to scientific computing and would be the perfect bridge between an introduction to programming and an advanced course on numerical methods or computational science.

Alex Small, IEEE, CISE Vol. 14 (2), March/April 2012

This fourth edition is a wonderful, inclusive textbook that covers pretty much everything one needs to know to go from zero to fairly sophisticated scientific programming in Python...

Joan Horvath, Computing Reviews, March 2015

Mathematics



► springer.com

1	Computing with Formulas	1
1.1	The First Programming Encounter: a Formula	1
1.1.1	Using a Program as a Calculator	2
1.1.2	About Programs and Programming	2
1.1.3	Tools for Writing Programs	3
1.1.4	Writing and Running Your First Python Program	4
1.1.5	Warning About Typing Program Text	5
1.1.6	Verifying the Result	6
1.1.7	Using Variables	6
1.1.8	Names of Variables	6
1.1.9	Reserved Words in Python	7
1.1.10	Comments	8
1.1.11	Formatting Text and Numbers	9
1.2	Computer Science Glossary	12
1.3	Another Formula: Celsius-Fahrenheit Conversion	16
1.3.1	Potential Error: Integer Division	16
1.3.2	Objects in Python	17
1.3.3	Avoiding Integer Division	18
1.3.4	Arithmetic Operators and Precedence	20
1.4	Evaluating Standard Mathematical Functions	20
1.4.1	Example: Using the Square Root Function	20
1.4.2	Example: Computing with $\sinh x$	23
1.4.3	A First Glimpse of Rounding Errors	23
1.5	Interactive Computing	24
1.5.1	Using the Python Shell	25
1.5.2	Type Conversion	26
1.5.3	IPython	27
1.6	Complex Numbers	29
1.6.1	Complex Arithmetics in Python	30
1.6.2	Complex Functions in Python	31
1.6.3	Unified Treatment of Complex and Real Functions	31
1.7	Symbolic Computing	33
1.7.1	Basic Differentiation and Integration	33
1.7.2	Equation Solving	34

1.7.3	Taylor Series and More	35
1.8	Summary	35
1.8.1	Chapter Topics	35
1.8.2	Example: Trajectory of a Ball	39
1.8.3	About Typesetting Conventions in This Book	40
1.9	Exercises	41
2	Loops and Lists	51
2.1	While Loops	51
2.1.1	A Naive Solution	51
2.1.2	While Loops	52
2.1.3	Boolean Expressions	54
2.1.4	Loop Implementation of a Sum	56
2.2	Lists	57
2.2.1	Basic List Operations	57
2.2.2	For Loops	60
2.3	Alternative Implementations with Lists and Loops	62
2.3.1	While Loop Implementation of a for Loop	62
2.3.2	The Range Construction	63
2.3.3	For Loops with List Indices	64
2.3.4	Changing List Elements	65
2.3.5	List Comprehension	66
2.3.6	Traversing Multiple Lists Simultaneously	66
2.4	Nested Lists	67
2.4.1	A table as a List of Rows or Columns	67
2.4.2	Printing Objects	68
2.4.3	Extracting Sublists	70
2.4.4	Traversing Nested Lists	72
2.5	Tuples	74
2.6	Summary	75
2.6.1	Chapter Topics	75
2.6.2	Example: Analyzing List Data	78
2.6.3	How to Find More Python Information	80
2.7	Exercises	82
3	Functions and Branching	91
3.1	Functions	91
3.1.1	Mathematical Functions as Python Functions	91
3.1.2	Understanding the Program Flow	93
3.1.3	Local and Global Variables	94
3.1.4	Multiple Arguments	96
3.1.5	Function Argument or Global Variable?	97
3.1.6	Beyond Mathematical Functions	98
3.1.7	Multiple Return Values	99
3.1.8	Computing Sums	100
3.1.9	Functions with No Return Values	101
3.1.10	Keyword Arguments	103
3.1.11	Doc Strings	105

3.1.12	Functions as Arguments to Functions	107
3.1.13	The Main Program	109
3.1.14	Lambda Functions	110
3.2	Branching	110
3.2.1	If-else Blocks	111
3.2.2	Inline if Tests	113
3.3	Mixing Loops, Branching, and Functions in Bioinformatics Examples	113
3.3.1	Counting Letters in DNA Strings	114
3.3.2	Efficiency Assessment	118
3.3.3	Verifying the Implementations	120
3.4	Summary	121
3.4.1	Chapter Topics	121
3.4.2	Example: Numerical Integration	123
3.5	Exercises	127
4	User Input and Error Handling	149
4.1	Asking Questions and Reading Answers	150
4.1.1	Reading Keyboard Input	150
4.2	Reading from the Command Line	151
4.2.1	Providing Input on the Command Line	151
4.2.2	A Variable Number of Command-Line Arguments	152
4.2.3	More on Command-Line Arguments	153
4.3	Turning User Text into Live Objects	154
4.3.1	The Magic Eval Function	154
4.3.2	The Magic Exec Function	158
4.3.3	Turning String Expressions into Functions	160
4.4	Option-Value Pairs on the Command Line	161
4.4.1	Basic Usage of the Argparse Module	162
4.4.2	Mathematical Expressions as Values	163
4.5	Reading Data from File	165
4.5.1	Reading a File Line by Line	166
4.5.2	Alternative Ways of Reading a File	167
4.5.3	Reading a Mixture of Text and Numbers	169
4.6	Writing Data to File	171
4.6.1	Example: Writing a Table to File	171
4.6.2	Standard Input and Output as File Objects	173
4.6.3	What is a File, Really?	176
4.7	Handling Errors	179
4.7.1	Exception Handling	180
4.7.2	Raising Exceptions	183
4.8	A Glimpse of Graphical User Interfaces	185
4.9	Making Modules	188
4.9.1	Example: Interest on Bank Deposits	188
4.9.2	Collecting Functions in a Module File	189
4.9.3	Test Block	190
4.9.4	Verification of the Module Code	192
4.9.5	Getting Input Data	193

4.9.6	Doc Strings in Modules	195
4.9.7	Using Modules	196
4.9.8	Distributing Modules	199
4.9.9	Making Software Available on the Internet	200
4.10	Making Code for Python 2 and 3	201
4.10.1	Basic Differences Between Python 2 and 3	201
4.10.2	Turning Python 2 Code into Python 3 Code	202
4.11	Summary	204
4.11.1	Chapter Topics	204
4.11.2	Example: Bisection Root Finding	208
4.12	Exercises	216
5	Array Computing and Curve Plotting	227
5.1	Vectors	228
5.1.1	The Vector Concept	228
5.1.2	Mathematical Operations on Vectors	229
5.1.3	Vector Arithmetics and Vector Functions	231
5.2	Arrays in Python Programs	232
5.2.1	Using Lists for Collecting Function Data	232
5.2.2	Basics of Numerical Python Arrays	233
5.2.3	Computing Coordinates and Function Values	235
5.2.4	Vectorization	236
5.3	Curve Plotting	238
5.3.1	MATLAB-Style Plotting with Matplotlib	238
5.3.2	Matplotlib; Pyplot Prefix	243
5.3.3	SciTools and Easyviz	244
5.3.4	Making Animations	249
5.3.5	Making Videos	254
5.3.6	Curve Plots in Pure Text	255
5.4	Plotting Difficulties	256
5.4.1	Piecewisely Defined Functions	256
5.4.2	Rapidly Varying Functions	259
5.5	More Advanced Vectorization of Functions	260
5.5.1	Vectorization of StringFunction Objects	260
5.5.2	Vectorization of the Heaviside Function	261
5.5.3	Vectorization of a Hat Function	265
5.6	More on Numerical Python Arrays	267
5.6.1	Copying Arrays	267
5.6.2	In-Place Arithmetics	268
5.6.3	Allocating Arrays	269
5.6.4	Generalized Indexing	269
5.6.5	Testing for the Array Type	270
5.6.6	Compact Syntax for Array Generation	271
5.6.7	Shape Manipulation	271
5.7	High-Performance Computing with Arrays	272
5.7.1	Scalar Implementation	272
5.7.2	Vectorized Implementation	273
5.7.3	Memory-Saving Implementation	273

5.7.4	Analysis of Memory Usage	275
5.7.5	Analysis of the CPU Time	276
5.8	Higher-Dimensional Arrays	277
5.8.1	Matrices and Arrays	277
5.8.2	Two-Dimensional Numerical Python Arrays	278
5.8.3	Array Computing	281
5.8.4	Matrix Objects	282
5.9	Some Common Linear Algebra Operations	283
5.9.1	Inverse, Determinant, and Eigenvalues	283
5.9.2	Products	283
5.9.3	Norms	284
5.9.4	Sum and Extreme Values	284
5.9.5	Indexing	286
5.9.6	Transpose and Upper/Lower Triangular Parts	286
5.9.7	Solving Linear Systems	287
5.9.8	Matrix Row and Column Operations	287
5.9.9	Computing the Rank of a Matrix	288
5.9.10	Symbolic Linear Algebra	289
5.10	Plotting of Scalar and Vector Fields	292
5.10.1	Installation	292
5.10.2	Surface Plots	293
5.10.3	Parameterized Curve	293
5.10.4	Contour Lines	294
5.10.5	The Gradient Vector Field	294
5.11	Matplotlib	296
5.11.1	Surface Plots	296
5.11.2	Contour Plots	297
5.11.3	Vector Field Plots	299
5.12	Mayavi	299
5.12.1	Surface Plots	300
5.12.2	Contour Plots	303
5.12.3	Vector Field Plots	303
5.12.4	A 3D Scalar Field and Its Gradient Field	304
5.12.5	Animations	306
5.13	Summary	307
5.13.1	Chapter Topics	307
5.13.2	Example: Animating a Function	308
5.14	Exercises	313
6	Dictionaries and Strings	333
6.1	Dictionaries	333
6.1.1	Making Dictionaries	334
6.1.2	Dictionary Operations	334
6.1.3	Example: Polynomials as Dictionaries	336
6.1.4	Dictionaries with Default Values and Ordering	338
6.1.5	Example: Storing File Data in Dictionaries	341
6.1.6	Example: Storing File Data in Nested Dictionaries	342

6.1.7	Example: Reading and Plotting Data Recorded at Specific Dates	347
6.2	Strings	351
6.2.1	Common Operations on Strings	351
6.2.2	Example: Reading Pairs of Numbers	355
6.2.3	Example: Reading Coordinates	358
6.3	Reading Data from Web Pages	360
6.3.1	About Web Pages	361
6.3.2	How to Access Web Pages in Programs	362
6.3.3	Example: Reading Pure Text Files	363
6.3.4	Example: Extracting Data from HTML	365
6.3.5	Handling Non-English Text	366
6.4	Reading and Writing Spreadsheet Files	369
6.4.1	CSV Files	369
6.4.2	Reading CSV Files	370
6.4.3	Processing Spreadsheet Data	371
6.4.4	Writing CSV Files	372
6.4.5	Representing Number Cells with Numerical Python Arrays	373
6.4.6	Using More High-Level Numerical Python Functionality	374
6.5	Examples from Analyzing DNA	375
6.5.1	Computing Frequencies	375
6.5.2	Analyzing the Frequency Matrix	382
6.5.3	Finding Base Frequencies	385
6.5.4	Translating Genes into Proteins	388
6.5.5	Some Humans Can Drink Milk, While Others Cannot	393
6.6	Making Code that is Compatible with Python 2 and 3	394
6.6.1	More Basic Differences Between Python 2 and 3	394
6.6.2	Turning Python 2 Code into Python 3 Code	396
6.7	Summary	396
6.7.1	Chapter Topics	396
6.7.2	Example: A File Database	398
6.8	Exercises	402
7	Introduction to Classes	409
7.1	Simple Function Classes	409
7.1.1	Challenge: Functions with Parameters	410
7.1.2	Representing a Function as a Class	412
7.1.3	The Self Variable	417
7.1.4	Another Function Class Example	419
7.1.5	Alternative Function Class Implementations	420
7.1.6	Making Classes Without the Class Construct	422
7.1.7	Closures	424
7.2	More Examples on Classes	426
7.2.1	Bank Accounts	426
7.2.2	Phone Book	428
7.2.3	A Circle	430
7.3	Special Methods	432
7.3.1	The Call Special Method	432

7.3.2	Example: Automagic Differentiation	433
7.3.3	Example: Automagic Integration	438
7.3.4	Turning an Instance into a String	440
7.3.5	Example: Phone Book with Special Methods	441
7.3.6	Adding Objects	443
7.3.7	Example: Class for Polynomials	443
7.3.8	Arithmetic Operations and Other Special Methods	449
7.3.9	Special Methods for String Conversion	449
7.4	Example: Class for Vectors in the Plane	451
7.4.1	Some Mathematical Operations on Vectors	451
7.4.2	Implementation	452
7.4.3	Usage	454
7.5	Example: Class for Complex Numbers	455
7.5.1	Implementation	455
7.5.2	Illegal Operations	457
7.5.3	Mixing Complex and Real Numbers	457
7.5.4	Dynamic, Static, Strong, Weak, and Duck Typing	459
7.5.5	Special Methods for “Right” Operands	460
7.5.6	Inspecting Instances	461
7.6	Static Methods and Attributes	463
7.7	Summary	464
7.7.1	Chapter Topics	464
7.7.2	Example: Interval Arithmetic	466
7.8	Exercises	470
8	Random Numbers and Simple Games	489
8.1	Drawing Random Numbers	489
8.1.1	The Seed	490
8.1.2	Uniformly Distributed Random Numbers	491
8.1.3	Visualizing the Distribution	492
8.1.4	Vectorized Drawing of Random Numbers	493
8.1.5	Computing the Mean and Standard Deviation	494
8.1.6	The Gaussian or Normal Distribution	496
8.2	Drawing Integers	497
8.2.1	Random Integer Functions	498
8.2.2	Example: Throwing a Die	498
8.2.3	Drawing a Random Element from a List	501
8.2.4	Example: Drawing Cards from a Deck	502
8.2.5	Example: Class Implementation of a Deck	504
8.3	Computing Probabilities	507
8.3.1	Principles of Monte Carlo Simulation	507
8.3.2	Example: Throwing Dice	508
8.3.3	Example: Drawing Balls from a Hat	511
8.3.4	Random Mutations of Genes	513
8.3.5	Example: Policies for Limiting Population Growth	519
8.4	Simple Games	522
8.4.1	Guessing a Number	522
8.4.2	Rolling Two Dice	523

8.5	Monte Carlo Integration	526
8.5.1	Derivation of Monte Carlo Integration	526
8.5.2	Implementation of Standard Monte Carlo Integration	528
8.5.3	Area Computing by Throwing Random Points	531
8.6	Random Walk in One Space Dimension	534
8.6.1	Basic Implementation	534
8.6.2	Visualization	535
8.6.3	Random Walk as a Difference Equation	536
8.6.4	Computing Statistics of the Particle Positions	536
8.6.5	Vectorized Implementation	537
8.7	Random Walk in Two Space Dimensions	539
8.7.1	Basic Implementation	539
8.7.2	Vectorized Implementation	541
8.8	Summary	542
8.8.1	Chapter Topics	542
8.8.2	Example: Random Growth	544
8.9	Exercises	549
9	Object-Oriented Programming	567
9.1	Inheritance and Class Hierarchies	567
9.1.1	A Class for Straight Lines	568
9.1.2	A First Try on a Class for Parabolas	569
9.1.3	A Class for Parabolas Using Inheritance	569
9.1.4	Checking the Class Type	571
9.1.5	Attribute vs Inheritance: has-a vs is-a Relationship	572
9.1.6	Superclass for Defining an Interface	574
9.2	Class Hierarchy for Numerical Differentiation	576
9.2.1	Classes for Differentiation	577
9.2.2	Verification	579
9.2.3	A flexible Main Program	581
9.2.4	Extensions	582
9.2.5	Alternative Implementation via Functions	585
9.2.6	Alternative Implementation via Functional Programming	586
9.2.7	Alternative Implementation via a Single Class	587
9.3	Class Hierarchy for Numerical Integration	589
9.3.1	Numerical Integration Methods	589
9.3.2	Classes for Integration	590
9.3.3	Verification	594
9.3.4	Using the Class Hierarchy	595
9.3.5	About Object-Oriented Programming	597
9.4	Class Hierarchy for Making Drawings	599
9.4.1	Using the Object Collection	600
9.4.2	Example of Classes for Geometric Objects	609
9.4.3	Adding Functionality via Recursion	614
9.4.4	Scaling, Translating, and Rotating a Figure	618
9.5	Classes for DNA Analysis	620
9.5.1	Class for Regions	620
9.5.2	Class for Genes	621

9.5.3	Subclasses	626
9.6	Summary	627
9.6.1	Chapter Topics	627
9.6.2	Example: Input Data Reader	629
9.7	Exercises	635
A	Sequences and Difference Equations	645
A.1	Mathematical Models Based on Difference Equations	646
A.1.1	Interest Rates	647
A.1.2	The Factorial as a Difference Equation	649
A.1.3	Fibonacci Numbers	650
A.1.4	Growth of a Population	651
A.1.5	Logistic Growth	652
A.1.6	Payback of a Loan	654
A.1.7	The Integral as a Difference Equation	655
A.1.8	Taylor Series as a Difference Equation	657
A.1.9	Making a Living from a Fortune	658
A.1.10	Newton's Method	659
A.1.11	The Inverse of a Function	663
A.2	Programming with Sound	665
A.2.1	Writing Sound to File	666
A.2.2	Reading Sound from File	667
A.2.3	Playing Many Notes	667
A.2.4	Music of a Sequence	668
A.3	Exercises	671
B	Introduction to Discrete Calculus	683
B.1	Discrete Functions	683
B.1.1	The Sine Function	684
B.1.2	Interpolation	685
B.1.3	Evaluating the Approximation	686
B.1.4	Generalization	687
B.2	Differentiation Becomes Finite Differences	688
B.2.1	Differentiating the Sine Function	689
B.2.2	Differences on a Mesh	690
B.2.3	Generalization	692
B.3	Integration Becomes Summation	693
B.3.1	Dividing into Subintervals	693
B.3.2	Integration on Subintervals	695
B.3.3	Adding the Subintervals	696
B.3.4	Generalization	697
B.4	Taylor Series	699
B.4.1	Approximating Functions Close to One Point	699
B.4.2	Approximating the Exponential Function	699
B.4.3	More Accurate Expansions	700
B.4.4	Accuracy of the Approximation	702
B.4.5	Derivatives Revisited	704
B.4.6	More Accurate Difference Approximations	705

B.4.7	Second-Order Derivatives	707
B.5	Exercises	709
C	Introduction to differential equations	715
C.1	The simplest case	716
C.2	Exponential Growth	718
C.3	Logistic Growth	723
C.4	A Simple Pendulum	724
C.5	A Model for the Spreading of a Disease	727
C.6	Exercises	729
D	A Complete Differential Equation Project	731
D.1	About the Problem: Motion and Forces in Physics	731
D.1.1	The Physical Problem	731
D.1.2	The Computational Algorithm	733
D.1.3	Derivation of the Mathematical Model	734
D.1.4	Derivation of the Algorithm	736
D.2	Program Development and Testing	737
D.2.1	Implementation	737
D.2.2	Callback Functionality	740
D.2.3	Making a Module	742
D.2.4	Verification	743
D.3	Visualization	746
D.3.1	Simultaneous Computation and Plotting	746
D.3.2	Some Applications	748
D.3.3	Remark on Choosing Δt	749
D.3.4	Comparing Several Quantities in Subplots	750
D.3.5	Comparing Approximate and Exact Solutions	751
D.3.6	Evolution of the Error as Δt Decreases	752
D.4	Exercises	755
E	Programming of Differential Equations	757
E.1	Scalar Ordinary Differential Equations	758
E.1.1	Examples on Right-Hand-Side Functions	758
E.1.2	The Forward Euler Scheme	759
E.1.3	Function Implementation	760
E.1.4	Verifying the Implementation	761
E.1.5	From Discrete to Continuous Solution	763
E.1.6	Switching Numerical Method	764
E.1.7	Class Implementation	764
E.1.8	Logistic Growth via a Function-Based Approach	769
E.1.9	Logistic Growth via a Class-Based Approach	769
E.2	Systems of Ordinary Differential Equations	772
E.2.1	Mathematical Problem	773
E.2.2	Example of a System of ODEs	774
E.2.3	Function Implementation	775
E.2.4	Class Implementation	777
E.3	<i>The ODESolver Class Hierarchy</i>	<i>779</i>

E.3.1	Numerical Methods	779
E.3.2	Construction of a Solver Hierarchy	780
E.3.3	The Backward Euler Method	783
E.3.4	Verification	785
E.3.5	Example: Exponential Decay	787
E.3.6	Example: The Logistic Equation with Problem and Solver Classes	789
E.3.7	Example: An Oscillating System	797
E.3.8	Application 4: The Trajectory of a Ball	799
E.3.9	Further Developments of ODESolver	801
E.4	Exercises	802
F	Debugging	835
F.1	Using a Debugger	835
F.2	How to Debug	838
F.2.1	A Recipe for Program Writing and Debugging	838
F.2.2	Application of the Recipe	841
F.2.3	Getting Help from a Code Analyzer	853
G	Migrating Python to Compiled Code	857
G.1	Pure Python Code for Monte Carlo Simulation	857
G.1.1	The Computational Problem	858
G.1.2	A Scalar Python Implementation	858
G.1.3	A Vectorized Python Implementation	859
G.2	Migrating Scalar Python Code to Cython	860
G.2.1	A Plain Cython Implementation	860
G.2.2	A Better Cython Implementation	863
G.3	Migrating Code to C	865
G.3.1	Writing a C Program	865
G.3.2	Migrating Loops to C Code via F2PY	866
G.3.3	Migrating Loops to C Code via Cython	867
G.3.4	Comparing Efficiency	868
H	Technical Topics	871
H.1	Getting Access to Python	871
H.1.1	Required Software	871
H.1.2	Installing Software on Your Laptop: Mac OS X and Windows	872
H.1.3	Anaconda and Spyder	873
H.1.4	VMWare Fusion Virtual Machine	874
H.1.5	Dual Boot on Windows	877
H.1.6	Vagrant Virtual Machine	877
H.2	How to Write and Run a Python Program	878
H.2.1	The Need for a Text Editor	878
H.2.2	Terminal Windows	880
H.3	The SageMathCloud and Wakari Web Services	880
H.3.1	Basic Intro to SageMathCloud	880
H.3.2	Basic Intro to Wakari	881

H.3.3	Installing Your Own Python Packages	881
H.4	Writing IPython Notebooks	882
H.4.1	A Simple Program in the Notebook	882
H.4.2	Mixing Text, Mathematics, Code, and Graphics	882
H.5	Different Ways of Running Python Programs	884
H.5.1	Executing Python Programs in iPython	884
H.5.2	Executing Python Programs in Unix	884
H.5.3	Executing Python Programs in Windows	885
H.5.4	Executing Python Programs in Mac OS X	887
H.5.5	Making a Complete Stand-Alone Executable	887
H.6	Doing Operating System Tasks in Python	888
H.7	Variable Number of Function Arguments	891
H.7.1	Variable Number of Positional Arguments	891
H.7.2	Variable Number of Keyword Arguments	894
H.8	Evaluating Program Efficiency	896
H.8.1	Making Time Measurements	896
H.8.2	Profiling Python Programs	898
H.9	Software Testing	899
H.9.1	Requirements of the Test Function	900
H.9.2	Writing the Test Function; Precomputed Data	900
H.9.3	Writing the Test Function; Exact Numerical Solution	901
H.9.4	Testing of Function Robustness	902
H.9.5	Automatic Execution of Tests	904
References	907
Index	909