

---

# Table of Contents

Preface .....	xxxiii
---------------	--------

---

## Part I. Getting Started

<b>1. A Python Q&amp;A Session .....</b>	<b>3</b>
Why Do People Use Python?	3
Software Quality	4
Developer Productivity	5
Is Python a “Scripting Language”?	5
OK, but What’s the Downside?	7
Who Uses Python Today?	9
What Can I Do with Python?	10
Systems Programming	11
GUIs	11
Internet Scripting	11
Component Integration	12
Database Programming	12
Rapid Prototyping	13
Numeric and Scientific Programming	13
And More: Gaming, Images, Data Mining, Robots, Excel...	14
How Is Python Developed and Supported?	15
Open Source Tradeoffs	15
What Are Python’s Technical Strengths?	16
It’s Object-Oriented and Functional	16
It’s Free	17
It’s Portable	17
It’s Powerful	18
It’s Mixable	19
It’s Relatively Easy to Use	19
It’s Relatively Easy to Learn	20
It’s Named After Monty Python	20

---

How Does Python Stack Up to Language X?	21
Chapter Summary	22
Test Your Knowledge: Quiz	23
Test Your Knowledge: Answers	23
<b>2. How Python Runs Programs .....</b>	<b>27</b>
Introducing the Python Interpreter	27
Program Execution	28
The Programmer's View	28
Python's View	30
Execution Model Variations	33
Python Implementation Alternatives	33
Execution Optimization Tools	37
Frozen Binaries	39
Future Possibilities?	40
Chapter Summary	40
Test Your Knowledge: Quiz	41
Test Your Knowledge: Answers	41
<b>3. How You Run Programs .....</b>	<b>43</b>
The Interactive Prompt	43
Starting an Interactive Session	44
The System Path	45
New Windows Options in 3.3: PATH, Launcher	46
Where to Run: Code Directories	47
What Not to Type: Prompts and Comments	48
Running Code Interactively	49
Why the Interactive Prompt?	50
Usage Notes: The Interactive Prompt	52
System Command Lines and Files	54
A First Script	55
Running Files with Command Lines	56
Command-Line Usage Variations	57
Usage Notes: Command Lines and Files	58
Unix-Style Executable Scripts: #!	59
Unix Script Basics	59
The Unix env Lookup Trick	60
The Python 3.3 Windows Launcher: #! Comes to Windows	60
Clicking File Icons	62
Icon-Click Basics	62
Clicking Icons on Windows	63
The input Trick on Windows	63
Other Icon-Click Limitations	66

Module Imports and Reloads	66
Import and Reload Basics	66
The Grander Module Story: Attributes	68
Usage Notes: import and reload	71
Using exec to Run Module Files	72
The IDLE User Interface	73
IDLE Startup Details	74
IDLE Basic Usage	75
IDLE Usability Features	76
Advanced IDLE Tools	77
Usage Notes: IDLE	78
Other IDEs	79
Other Launch Options	81
Embedding Calls	81
Frozen Binary Executables	82
Text Editor Launch Options	82
Still Other Launch Options	82
Future Possibilities?	83
Which Option Should I Use?	83
Chapter Summary	85
Test Your Knowledge: Quiz	85
Test Your Knowledge: Answers	86
Test Your Knowledge: Part I Exercises	87

---

## Part II. Types and Operations

<b>4. Introducing Python Object Types .....</b>	<b>93</b>
The Python Conceptual Hierarchy	93
Why Use Built-in Types?	94
Python's Core Data Types	95
Numbers	97
Strings	99
Sequence Operations	99
Immutability	101
Type-Specific Methods	102
Getting Help	104
Other Ways to Code Strings	105
Unicode Strings	106
Pattern Matching	108
Lists	109
Sequence Operations	109
Type-Specific Operations	109

Bounds Checking	110
Nesting	110
Comprehensions	111
Dictionaries	113
Mapping Operations	114
Nesting Revisited	115
Missing Keys: if Tests	116
Sorting Keys: for Loops	118
Iteration and Optimization	120
Tuples	121
Why Tuples?	122
Files	122
Binary Bytes Files	123
Unicode Text Files	124
Other File-Like Tools	126
Other Core Types	126
How to Break Your Code's Flexibility	128
User-Defined Classes	129
And Everything Else	130
Chapter Summary	130
Test Your Knowledge: Quiz	131
Test Your Knowledge: Answers	131
<b>5. Numeric Types .....</b>	<b>133</b>
Numeric Type Basics	133
Numeric Literals	134
Built-in Numeric Tools	136
Python Expression Operators	136
Numbers in Action	141
Variables and Basic Expressions	141
Numeric Display Formats	143
Comparisons: Normal and Chained	144
Division: Classic, Floor, and True	146
Integer Precision	150
Complex Numbers	151
Hex, Octal, Binary: Literals and Conversions	151
Bitwise Operations	153
Other Built-in Numeric Tools	155
Other Numeric Types	157
Decimal Type	157
Fraction Type	160
Sets	163
Booleans	171

Numeric Extensions	172
Chapter Summary	172
Test Your Knowledge: Quiz	173
Test Your Knowledge: Answers	173
<b>6. The Dynamic Typing Interlude .....</b>	<b>175</b>
The Case of the Missing Declaration Statements	175
Variables, Objects, and References	176
Types Live with Objects, Not Variables	177
Objects Are Garbage-Collected	178
Shared References	180
Shared References and In-Place Changes	181
Shared References and Equality	183
Dynamic Typing Is Everywhere	185
Chapter Summary	186
Test Your Knowledge: Quiz	186
Test Your Knowledge: Answers	186
<b>7. String Fundamentals .....</b>	<b>189</b>
This Chapter's Scope	189
Unicode: The Short Story	189
String Basics	190
String Literals	192
Single- and Double-Quoted Strings Are the Same	193
Escape Sequences Represent Special Characters	193
Raw Strings Suppress Escapes	196
Triple Quotes Code Multiline Block Strings	198
Strings in Action	200
Basic Operations	200
Indexing and Slicing	201
String Conversion Tools	205
Changing Strings I	208
String Methods	209
Method Call Syntax	209
Methods of Strings	210
String Method Examples: Changing Strings II	211
String Method Examples: Parsing Text	213
Other Common String Methods in Action	214
The Original string Module's Functions (Gone in 3.X)	215
String Formatting Expressions	216
Formatting Expression Basics	217
Advanced Formatting Expression Syntax	218
Advanced Formatting Expression Examples	220

Dictionary-Based Formatting Expressions	221
String Formatting Method Calls	222
Formatting Method Basics	222
Adding Keys, Attributes, and Offsets	223
Advanced Formatting Method Syntax	224
Advanced Formatting Method Examples	225
Comparison to the % Formatting Expression	227
Why the Format Method?	230
General Type Categories	235
Types Share Operation Sets by Categories	235
Mutable Types Can Be Changed in Place	236
Chapter Summary	237
Test Your Knowledge: Quiz	237
Test Your Knowledge: Answers	237
<b>8. Lists and Dictionaries .....</b>	<b>239</b>
Lists	239
Lists in Action	242
Basic List Operations	242
List Iteration and Comprehensions	242
Indexing, Slicing, and Matrixes	243
Changing Lists in Place	244
Dictionaries	250
Dictionaries in Action	252
Basic Dictionary Operations	253
Changing Dictionaries in Place	254
More Dictionary Methods	254
Example: Movie Database	256
Dictionary Usage Notes	258
Other Ways to Make Dictionaries	262
Dictionary Changes in Python 3.X and 2.7	264
Chapter Summary	271
Test Your Knowledge: Quiz	272
Test Your Knowledge: Answers	272
<b>9. Tuples, Files, and Everything Else .....</b>	<b>275</b>
Tuples	276
Tuples in Action	277
Why Lists and Tuples?	279
Records Revisited: Named Tuples	280
Files	282
Opening Files	283
Using Files	284

Files in Action	285
Text and Binary Files: The Short Story	287
Storing Python Objects in Files: Conversions	288
Storing Native Python Objects: pickle	290
Storing Python Objects in JSON Format	291
Storing Packed Binary Data: struct	293
File Context Managers	294
Other File Tools	294
Core Types Review and Summary	295
Object Flexibility	297
References Versus Copies	297
Comparisons, Equality, and Truth	300
The Meaning of True and False in Python	304
Python's Type Hierarchies	306
Type Objects	306
Other Types in Python	308
Built-in Type Gotchas	308
Assignment Creates References, Not Copies	308
Repetition Adds One Level Deep	309
Beware of Cyclic Data Structures	310
Immutable Types Can't Be Changed in Place	311
Chapter Summary	311
Test Your Knowledge: Quiz	311
Test Your Knowledge: Answers	312
Test Your Knowledge: Part II Exercises	313

---

## Part III. Statements and Syntax

<b>10. Introducing Python Statements</b> .....	<b>319</b>
The Python Conceptual Hierarchy Revisited	319
Python's Statements	320
A Tale of Two ifs	322
What Python Adds	322
What Python Removes	323
Why Indentation Syntax?	324
A Few Special Cases	327
A Quick Example: Interactive Loops	329
A Simple Interactive Loop	329
Doing Math on User Inputs	331
Handling Errors by Testing Inputs	332
Handling Errors with try Statements	333
Nesting Code Three Levels Deep	335

Chapter Summary	336
Test Your Knowledge: Quiz	336
Test Your Knowledge: Answers	336
<b>11. Assignments, Expressions, and Prints .....</b>	<b>339</b>
Assignment Statements	339
Assignment Statement Forms	340
Sequence Assignments	341
Extended Sequence Unpacking in Python 3.X	344
Multiple-Target Assignments	348
Augmented Assignments	350
Variable Name Rules	352
Expression Statements	356
Expression Statements and In-Place Changes	357
Print Operations	358
The Python 3.X print Function	359
The Python 2.X print Statement	361
Print Stream Redirection	363
Version-Neutral Printing	366
Chapter Summary	369
Test Your Knowledge: Quiz	370
Test Your Knowledge: Answers	370
<b>12. if Tests and Syntax Rules .....</b>	<b>371</b>
if Statements	371
General Format	371
Basic Examples	372
Multiway Branching	372
Python Syntax Revisited	375
Block Delimiters: Indentation Rules	376
Statement Delimiters: Lines and Continuations	378
A Few Special Cases	379
Truth Values and Boolean Tests	380
The if/else Ternary Expression	382
Chapter Summary	385
Test Your Knowledge: Quiz	386
Test Your Knowledge: Answers	386
<b>13. while and for Loops .....</b>	<b>387</b>
while Loops	387
General Format	388
Examples	388
break, continue, pass, and the Loop else	389

General Loop Format	389
pass	390
continue	391
break	391
Loop else	392
for Loops	395
General Format	395
Examples	395
Loop Coding Techniques	402
Counter Loops: range	402
Sequence Scans: while and range Versus for	403
Sequence Shufflers: range and len	404
Nonexhaustive Traversals: range Versus Slices	405
Changing Lists: range Versus Comprehensions	406
Parallel Traversals: zip and map	407
Generating Both Offsets and Items: enumerate	410
Chapter Summary	413
Test Your Knowledge: Quiz	414
Test Your Knowledge: Answers	414
<b>14. Iterations and Comprehensions .....</b>	<b>415</b>
Iterations: A First Look	416
The Iteration Protocol: File Iterators	416
Manual Iteration: iter and next	419
Other Built-in Type Iterables	422
List Comprehensions: A First Detailed Look	424
List Comprehension Basics	425
Using List Comprehensions on Files	426
Extended List Comprehension Syntax	427
Other Iteration Contexts	429
New Iterables in Python 3.X	434
Impacts on 2.X Code: Pros and Cons	434
The range Iterable	435
The map, zip, and filter Iterables	436
Multiple Versus Single Pass Iterators	438
Dictionary View Iterables	439
Other Iteration Topics	440
Chapter Summary	441
Test Your Knowledge: Quiz	441
Test Your Knowledge: Answers	441
<b>15. The Documentation Interlude .....</b>	<b>443</b>
Python Documentation Sources	443

# Comments	444
The dir Function	444
Docstrings: <code>__doc__</code>	446
PyDoc: The help Function	449
PyDoc: HTML Reports	452
Beyond docstrings: Sphinx	461
The Standard Manual Set	461
Web Resources	462
Published Books	463
Common Coding Gotchas	463
Chapter Summary	465
Test Your Knowledge: Quiz	466
Test Your Knowledge: Answers	466
Test Your Knowledge: Part III Exercises	467

---

## Part IV. Functions and Generators

<b>16. Function Basics</b> .....	<b>473</b>
Why Use Functions?	474
Coding Functions	475
def Statements	476
def Executes at Runtime	477
A First Example: Definitions and Calls	478
Definition	478
Calls	478
Polymorphism in Python	479
A Second Example: Intersecting Sequences	480
Definition	481
Calls	481
Polymorphism Revisited	482
Local Variables	483
Chapter Summary	483
Test Your Knowledge: Quiz	483
Test Your Knowledge: Answers	484
<b>17. Scopes</b> .....	<b>485</b>
Python Scope Basics	485
Scope Details	486
Name Resolution: The LEGB Rule	488
Scope Example	490
The Built-in Scope	491
The global Statement	494

Program Design: Minimize Global Variables	495
Program Design: Minimize Cross-File Changes	497
Other Ways to Access Globals	498
Scopes and Nested Functions	499
Nested Scope Details	500
Nested Scope Examples	500
Factory Functions: Closures	501
Retaining Enclosing Scope State with Defaults	504
The nonlocal Statement in 3.X	508
nonlocal Basics	508
nonlocal in Action	509
Why nonlocal? State Retention Options	512
State with nonlocal: 3.X only	512
State with Globals: A Single Copy Only	513
State with Classes: Explicit Attributes (Preview)	513
State with Function Attributes: 3.X and 2.X	515
Chapter Summary	519
Test Your Knowledge: Quiz	519
Test Your Knowledge: Answers	520
<b>18. Arguments .....</b>	<b>523</b>
Argument-Passing Basics	523
Arguments and Shared References	524
Avoiding Mutable Argument Changes	526
Simulating Output Parameters and Multiple Results	527
Special Argument-Matching Modes	528
Argument Matching Basics	529
Argument Matching Syntax	530
The Gritty Details	531
Keyword and Default Examples	532
Arbitrary Arguments Examples	534
Python 3.X Keyword-Only Arguments	539
The min Wakeup Call!	542
Full Credit	542
Bonus Points	544
The Punch Line...	544
Generalized Set Functions	545
Emulating the Python 3.X print Function	547
Using Keyword-Only Arguments	548
Chapter Summary	550
Test Your Knowledge: Quiz	551
Test Your Knowledge: Answers	552

<b>19. Advanced Function Topics .....</b>	<b>553</b>
Function Design Concepts	553
Recursive Functions	555
Summation with Recursion	555
Coding Alternatives	556
Loop Statements Versus Recursion	557
Handling Arbitrary Structures	558
Function Objects: Attributes and Annotations	562
Indirect Function Calls: “First Class” Objects	562
Function Introspection	563
Function Attributes	564
Function Annotations in 3.X	565
Anonymous Functions: lambda	567
lambda Basics	568
Why Use lambda?	569
How (Not) to Obfuscate Your Python Code	571
Scopes: lambdas Can Be Nested Too	572
Functional Programming Tools	574
Mapping Functions over Iterables: map	574
Selecting Items in Iterables: filter	576
Combining Items in Iterables: reduce	576
Chapter Summary	578
Test Your Knowledge: Quiz	578
Test Your Knowledge: Answers	578
<b>20. Comprehensions and Generations .....</b>	<b>581</b>
List Comprehensions and Functional Tools	581
List Comprehensions Versus map	582
Adding Tests and Nested Loops: filter	583
Example: List Comprehensions and Matrixes	586
Don’t Abuse List Comprehensions: KISS	588
Generator Functions and Expressions	591
Generator Functions: yield Versus return	592
Generator Expressions: Iterables Meet Comprehensions	597
Generator Functions Versus Generator Expressions	602
Generators Are Single-Iteration Objects	604
Generation in Built-in Types, Tools, and Classes	606
Example: Generating Scrambled Sequences	609
Don’t Abuse Generators: EIBTI	614
Example: Emulating zip and map with Iteration Tools	617
Comprehension Syntax Summary	622
Scopes and Comprehension Variables	623
Comprehending Set and Dictionary Comprehensions	624

Extended Comprehension Syntax for Sets and Dictionaries	625
Chapter Summary	626
Test Your Knowledge: Quiz	626
Test Your Knowledge: Answers	626
<b>21. The Benchmarking Interlude .....</b>	<b>629</b>
Timing Iteration Alternatives	629
Timing Module: Homegrown	630
Timing Script	634
Timing Results	635
Timing Module Alternatives	638
Other Suggestions	642
Timing Iterations and Pythons with timeit	642
Basic timeit Usage	643
Benchmark Module and Script: timeit	647
Benchmark Script Results	649
More Fun with Benchmarks	651
Other Benchmarking Topics: pystones	656
Function Gotchas	656
Local Names Are Detected Statically	657
Defaults and Mutable Objects	658
Functions Without returns	660
Miscellaneous Function Gotchas	661
Chapter Summary	661
Test Your Knowledge: Quiz	662
Test Your Knowledge: Answers	662
Test Your Knowledge: Part IV Exercises	663

---

## Part V. Modules and Packages

<b>22. Modules: The Big Picture .....</b>	<b>669</b>
Why Use Modules?	669
Python Program Architecture	670
How to Structure a Program	671
Imports and Attributes	671
Standard Library Modules	673
How Imports Work	674
1. Find It	674
2. Compile It (Maybe)	675
3. Run It	675
Byte Code Files: <code>__pycache__</code> in Python 3.2+	676
Byte Code File Models in Action	677

The Module Search Path	678
Configuring the Search Path	681
Search Path Variations	681
The sys.path List	681
Module File Selection	682
Chapter Summary	685
Test Your Knowledge: Quiz	685
Test Your Knowledge: Answers	685
<b>23. Module Coding Basics .....</b>	<b>687</b>
Module Creation	687
Module Filenames	687
Other Kinds of Modules	688
Module Usage	688
The import Statement	689
The from Statement	689
The from * Statement	689
Imports Happen Only Once	690
import and from Are Assignments	691
import and from Equivalence	692
Potential Pitfalls of the from Statement	693
Module Namespaces	694
Files Generate Namespaces	695
Namespace Dictionaries: __dict__	696
Attribute Name Qualification	697
Imports Versus Scopes	698
Namespace Nesting	699
Reloading Modules	700
reload Basics	701
reload Example	702
Chapter Summary	703
Test Your Knowledge: Quiz	704
Test Your Knowledge: Answers	704
<b>24. Module Packages .....</b>	<b>707</b>
Package Import Basics	708
Packages and Search Path Settings	708
Package __init__.py Files	709
Package Import Example	711
from Versus import with Packages	713
Why Use Package Imports?	713
A Tale of Three Systems	714
Package Relative Imports	717

Changes in Python 3.X	718
Relative Import Basics	718
Why Relative Imports?	720
The Scope of Relative Imports	722
Module Lookup Rules Summary	723
Relative Imports in Action	723
Pitfalls of Package-Relative Imports: Mixed Use	729
Python 3.3 Namespace Packages	734
Namespace Package Semantics	735
Impacts on Regular Packages: Optional <code>__init__.py</code>	736
Namespace Packages in Action	737
Namespace Package Nesting	738
Files Still Have Precedence over Directories	740
Chapter Summary	742
Test Your Knowledge: Quiz	742
Test Your Knowledge: Answers	742
<b>25. Advanced Module Topics .....</b>	<b>745</b>
Module Design Concepts	745
Data Hiding in Modules	747
Minimizing from * Damage: <code>_X</code> and <code>__all__</code>	747
Enabling Future Language Features: <code>__future__</code>	748
Mixed Usage Modes: <code>__name__</code> and <code>__main__</code>	749
Unit Tests with <code>__name__</code>	750
Example: Dual Mode Code	751
Currency Symbols: Unicode in Action	754
Docstrings: Module Documentation at Work	756
Changing the Module Search Path	756
The <code>as</code> Extension for <code>import</code> and <code>from</code>	758
Example: Modules Are Objects	759
Importing Modules by Name String	761
Running Code Strings	762
Direct Calls: Two Options	762
Example: Transitive Module Reloads	763
A Recursive Reloader	764
Alternative Codings	767
Module Gotchas	770
Module Name Clashes: Package and Package-Relative Imports	771
Statement Order Matters in Top-Level Code	771
<code>from</code> Copies Names but Doesn't Link	772
<code>from *</code> Can Obscure the Meaning of Variables	773
<code>reload</code> May Not Impact <code>from</code> Imports	773
<code>reload</code> , <code>from</code> , and Interactive Testing	774

Recursive from Imports May Not Work	775
Chapter Summary	776
Test Your Knowledge: Quiz	777
Test Your Knowledge: Answers	777
Test Your Knowledge: Part V Exercises	778

---

## Part VI. Classes and OOP

<b>26. OOP: The Big Picture</b> .....	<b>783</b>
Why Use Classes?	784
OOP from 30,000 Feet	785
Attribute Inheritance Search	785
Classes and Instances	788
Method Calls	788
Coding Class Trees	789
Operator Overloading	791
OOP Is About Code Reuse	792
Chapter Summary	795
Test Your Knowledge: Quiz	795
Test Your Knowledge: Answers	795
<b>27. Class Coding Basics</b> .....	<b>797</b>
Classes Generate Multiple Instance Objects	797
Class Objects Provide Default Behavior	798
Instance Objects Are Concrete Items	798
A First Example	799
Classes Are Customized by Inheritance	801
A Second Example	802
Classes Are Attributes in Modules	804
Classes Can Intercept Python Operators	805
A Third Example	806
Why Use Operator Overloading?	808
The World's Simplest Python Class	809
Records Revisited: Classes Versus Dictionaries	812
Chapter Summary	814
Test Your Knowledge: Quiz	815
Test Your Knowledge: Answers	815
<b>28. A More Realistic Example</b> .....	<b>817</b>
Step 1: Making Instances	818
Coding Constructors	818
Testing As You Go	819

Using Code Two Ways	820
Step 2: Adding Behavior Methods	822
Coding Methods	824
Step 3: Operator Overloading	826
Providing Print Displays	826
Step 4: Customizing Behavior by Subclassing	828
Coding Subclasses	828
Augmenting Methods: The Bad Way	829
Augmenting Methods: The Good Way	829
Polymorphism in Action	832
Inherit, Customize, and Extend	833
OOP: The Big Idea	833
Step 5: Customizing Constructors, Too	834
OOP Is Simpler Than You May Think	836
Other Ways to Combine Classes	836
Step 6: Using Introspection Tools	840
Special Class Attributes	840
A Generic Display Tool	842
Instance Versus Class Attributes	843
Name Considerations in Tool Classes	844
Our Classes' Final Form	845
Step 7 (Final): Storing Objects in a Database	847
Pickles and Shelves	847
Storing Objects on a Shelf Database	848
Exploring Shelves Interactively	849
Updating Objects on a Shelf	851
Future Directions	853
Chapter Summary	855
Test Your Knowledge: Quiz	855
Test Your Knowledge: Answers	856

<b>29. Class Coding Details .....</b>	<b>859</b>
The class Statement	859
General Form	860
Example	860
Methods	862
Method Example	863
Calling Superclass Constructors	864
Other Method Call Possibilities	864
Inheritance	865
Attribute Tree Construction	865
Specializing Inherited Methods	866
Class Interface Techniques	867

Abstract Superclasses	869
Namespaces: The Conclusion	872
Simple Names: Global Unless Assigned	872
Attribute Names: Object Namespaces	872
The “Zen” of Namespaces: Assignments Classify Names	873
Nested Classes: The LEGB Scopes Rule Revisited	875
Namespace Dictionaries: Review	878
Namespace Links: A Tree Climber	880
Documentation Strings Revisited	882
Classes Versus Modules	884
Chapter Summary	884
Test Your Knowledge: Quiz	884
Test Your Knowledge: Answers	885
<b>30. Operator Overloading .....</b>	<b>887</b>
The Basics	887
Constructors and Expressions: <code>__init__</code> and <code>__sub__</code>	888
Common Operator Overloading Methods	888
Indexing and Slicing: <code>__getitem__</code> and <code>__setitem__</code>	890
Intercepting Slices	891
Slicing and Indexing in Python 2.X	893
But 3.X’s <code>__index__</code> Is Not Indexing!	894
Index Iteration: <code>__getitem__</code>	894
Iterable Objects: <code>__iter__</code> and <code>__next__</code>	895
User-Defined Iterables	896
Multiple Iterators on One Object	899
Coding Alternative: <code>__iter__</code> plus <code>yield</code>	902
Membership: <code>__contains__</code> , <code>__iter__</code> , and <code>__getitem__</code>	906
Attribute Access: <code>__getattr__</code> and <code>__setattr__</code>	909
Attribute Reference	909
Attribute Assignment and Deletion	910
Other Attribute Management Tools	912
Emulating Privacy for Instance Attributes: Part 1	912
String Representation: <code>__repr__</code> and <code>__str__</code>	913
Why Two Display Methods?	914
Display Usage Notes	916
Right-Side and In-Place Uses: <code>__radd__</code> and <code>__iadd__</code>	917
Right-Side Addition	917
In-Place Addition	920
Call Expressions: <code>__call__</code>	921
Function Interfaces and Callback-Based Code	923
Comparisons: <code>__lt__</code> , <code>__gt__</code> , and Others	925
The <code>__cmp__</code> Method in Python 2.X	926

Boolean Tests: <code>__bool__</code> and <code>__len__</code>	927
Boolean Methods in Python 2.X	928
Object Destruction: <code>__del__</code>	929
Destructor Usage Notes	930
Chapter Summary	931
Test Your Knowledge: Quiz	931
Test Your Knowledge: Answers	931
<b>31. Designing with Classes .....</b>	<b>933</b>
Python and OOP	933
Polymorphism Means Interfaces, Not Call Signatures	934
OOP and Inheritance: “Is-a” Relationships	935
OOP and Composition: “Has-a” Relationships	937
Stream Processors Revisited	938
OOP and Delegation: “Wrapper” Proxy Objects	942
Pseudoprivate Class Attributes	944
Name Mangling Overview	945
Why Use Pseudoprivate Attributes?	945
Methods Are Objects: Bound or Unbound	948
Unbound Methods Are Functions in 3.X	950
Bound Methods and Other Callable Objects	951
Classes Are Objects: Generic Object Factories	954
Why Factories?	955
Multiple Inheritance: “Mix-in” Classes	956
Coding Mix-in Display Classes	957
Other Design-Related Topics	977
Chapter Summary	977
Test Your Knowledge: Quiz	978
Test Your Knowledge: Answers	978
<b>32. Advanced Class Topics .....</b>	<b>979</b>
Extending Built-in Types	980
Extending Types by Embedding	980
Extending Types by Subclassing	981
The “New Style” Class Model	983
Just How New Is New-Style?	984
New-Style Class Changes	985
Attribute Fetch for Built-ins Skips Instances	987
Type Model Changes	992
All Classes Derive from “object”	995
Diamond Inheritance Change	997
More on the MRO: Method Resolution Order	1001
Example: Mapping Attributes to Inheritance Sources	1004

New-Style Class Extensions	1010
Slots: Attribute Declarations	1010
Properties: Attribute Accessors	1020
__getattr__ and Descriptors: Attribute Tools	1023
Other Class Changes and Extensions	1023
Static and Class Methods	1024
Why the Special Methods?	1024
Static Methods in 2.X and 3.X	1025
Static Method Alternatives	1027
Using Static and Class Methods	1028
Counting Instances with Static Methods	1030
Counting Instances with Class Methods	1031
Decorators and Metaclasses: Part 1	1034
Function Decorator Basics	1035
A First Look at User-Defined Function Decorators	1037
A First Look at Class Decorators and Metaclasses	1038
For More Details	1040
The super Built-in Function: For Better or Worse?	1041
The Great super Debate	1041
Traditional Superclass Call Form: Portable, General	1042
Basic super Usage and Its Tradeoffs	1043
The super Upsides: Tree Changes and Dispatch	1049
Runtime Class Changes and super	1049
Cooperative Multiple Inheritance Method Dispatch	1050
The super Summary	1062
Class Gotchas	1064
Changing Class Attributes Can Have Side Effects	1064
Changing Mutable Class Attributes Can Have Side Effects, Too	1066
Multiple Inheritance: Order Matters	1066
Scopes in Methods and Classes	1068
Miscellaneous Class Gotchas	1069
KISS Revisited: “Overwrapping-itis”	1070
Chapter Summary	1070
Test Your Knowledge: Quiz	1071
Test Your Knowledge: Answers	1071
Test Your Knowledge: Part VI Exercises	1072

---

## Part VII. Exceptions and Tools

<b>33. Exception Basics</b> .....	<b>1081</b>
Why Use Exceptions?	1081
Exception Roles	1082

Exceptions: The Short Story	1083
Default Exception Handler	1083
Catching Exceptions	1084
Raising Exceptions	1085
User-Defined Exceptions	1086
Termination Actions	1087
Chapter Summary	1089
Test Your Knowledge: Quiz	1090
Test Your Knowledge: Answers	1090
<b>34. Exception Coding Details .....</b>	<b>1093</b>
The try/except/else Statement	1093
How try Statements Work	1094
try Statement Clauses	1095
The try else Clause	1098
Example: Default Behavior	1098
Example: Catching Built-in Exceptions	1100
The try/finally Statement	1100
Example: Coding Termination Actions with try/finally	1101
Unified try/except/finally	1102
Unified try Statement Syntax	1104
Combining finally and except by Nesting	1104
Unified try Example	1105
The raise Statement	1106
Raising Exceptions	1107
Scopes and try except Variables	1108
Propagating Exceptions with raise	1110
Python 3.X Exception Chaining: raise from	1110
The assert Statement	1112
Example: Trapping Constraints (but Not Errors!)	1113
with/as Context Managers	1114
Basic Usage	1114
The Context Management Protocol	1116
Multiple Context Managers in 3.1, 2.7, and Later	1118
Chapter Summary	1119
Test Your Knowledge: Quiz	1120
Test Your Knowledge: Answers	1120
<b>35. Exception Objects .....</b>	<b>1123</b>
Exceptions: Back to the Future	1124
String Exceptions Are Right Out!	1124
Class-Based Exceptions	1125
Coding Exceptions Classes	1126

Why Exception Hierarchies?	1128
Built-in Exception Classes	1131
Built-in Exception Categories	1132
Default Printing and State	1133
Custom Print Displays	1135
Custom Data and Behavior	1136
Providing Exception Details	1136
Providing Exception Methods	1137
Chapter Summary	1139
Test Your Knowledge: Quiz	1139
Test Your Knowledge: Answers	1139
<b>36. Designing with Exceptions .....</b>	<b>1141</b>
Nesting Exception Handlers	1141
Example: Control-Flow Nesting	1143
Example: Syntactic Nesting	1143
Exception Idioms	1145
Breaking Out of Multiple Nested Loops: “go to”	1145
Exceptions Aren’t Always Errors	1146
Functions Can Signal Conditions with raise	1147
Closing Files and Server Connections	1148
Debugging with Outer try Statements	1149
Running In-Process Tests	1149
More on sys.exc_info	1150
Displaying Errors and Tracebacks	1151
Exception Design Tips and Gotchas	1152
What Should Be Wrapped	1152
Catching Too Much: Avoid Empty except and Exception	1153
Catching Too Little: Use Class-Based Categories	1155
Core Language Summary	1155
The Python Toolset	1156
Development Tools for Larger Projects	1157
Chapter Summary	1160
Test Your Knowledge: Quiz	1161
Test Your Knowledge: Answers	1161
Test Your Knowledge: Part VII Exercises	1161

---

## Part VIII. Advanced Topics

<b>37. Unicode and Byte Strings .....</b>	<b>1165</b>
String Changes in 3.X	1166
String Basics	1167

Character Encoding Schemes	1167
How Python Stores Strings in Memory	1170
Python's String Types	1171
Text and Binary Files	1173
Coding Basic Strings	1174
Python 3.X String Literals	1175
Python 2.X String Literals	1176
String Type Conversions	1177
Coding Unicode Strings	1178
Coding ASCII Text	1178
Coding Non-ASCII Text	1179
Encoding and Decoding Non-ASCII text	1180
Other Encoding Schemes	1181
Byte String Literals: Encoded Text	1183
Converting Encodings	1184
Coding Unicode Strings in Python 2.X	1185
Source File Character Set Encoding Declarations	1187
Using 3.X bytes Objects	1189
Method Calls	1189
Sequence Operations	1190
Other Ways to Make bytes Objects	1191
Mixing String Types	1192
Using 3.X/2.6+ bytearray Objects	1192
bytearrays in Action	1193
Python 3.X String Types Summary	1195
Using Text and Binary Files	1195
Text File Basics	1196
Text and Binary Modes in 2.X and 3.X	1197
Type and Content Mismatches in 3.X	1198
Using Unicode Files	1199
Reading and Writing Unicode in 3.X	1199
Handling the BOM in 3.X	1201
Unicode Files in 2.X	1204
Unicode Filenames and Streams	1205
Other String Tool Changes in 3.X	1206
The re Pattern-Matching Module	1206
The struct Binary Data Module	1207
The pickle Object Serialization Module	1209
XML Parsing Tools	1211
Chapter Summary	1215
Test Your Knowledge: Quiz	1215
Test Your Knowledge: Answers	1216

<b>38. Managed Attributes .....</b>	<b>1219</b>
Why Manage Attributes?	1219
Inserting Code to Run on Attribute Access	1220
Properties	1221
The Basics	1222
A First Example	1222
Computed Attributes	1224
Coding Properties with Decorators	1224
Descriptors	1226
The Basics	1227
A First Example	1229
Computed Attributes	1231
Using State Information in Descriptors	1232
How Properties and Descriptors Relate	1236
__getattr__ and __getattribute__	1237
The Basics	1238
A First Example	1241
Computed Attributes	1243
__getattr__ and __getattribute__ Compared	1245
Management Techniques Compared	1246
Intercepting Built-in Operation Attributes	1249
Example: Attribute Validations	1256
Using Properties to Validate	1256
Using Descriptors to Validate	1259
Using __getattr__ to Validate	1263
Using __getattribute__ to Validate	1265
Chapter Summary	1266
Test Your Knowledge: Quiz	1266
Test Your Knowledge: Answers	1267
<b>39. Decorators .....</b>	<b>1269</b>
What's a Decorator?	1269
Managing Calls and Instances	1270
Managing Functions and Classes	1270
Using and Defining Decorators	1271
Why Decorators?	1271
The Basics	1273
Function Decorators	1273
Class Decorators	1277
Decorator Nesting	1279
Decorator Arguments	1281
Decorators Manage Functions and Classes, Too	1282
Coding Function Decorators	1283

Tracing Calls	1283
Decorator State Retention Options	1285
Class Blunders I: Decorating Methods	1289
Timing Calls	1295
Adding Decorator Arguments	1298
Coding Class Decorators	1301
Singleton Classes	1301
Tracing Object Interfaces	1303
Class Blunders II: Retaining Multiple Instances	1308
Decorators Versus Manager Functions	1309
Why Decorators? (Revisited)	1310
Managing Functions and Classes Directly	1312
Example: “Private” and “Public” Attributes	1314
Implementing Private Attributes	1314
Implementation Details I	1317
Generalizing for Public Declarations, Too	1318
Implementation Details II	1320
Open Issues	1321
Python Isn’t About Control	1329
Example: Validating Function Arguments	1330
The Goal	1330
A Basic Range-Testing Decorator for Positional Arguments	1331
Generalizing for Keywords and Defaults, Too	1333
Implementation Details	1336
Open Issues	1338
Decorator Arguments Versus Function Annotations	1340
Other Applications: Type Testing (If You Insist!)	1342
Chapter Summary	1343
Test Your Knowledge: Quiz	1344
Test Your Knowledge: Answers	1345
<b>40. Metaclasses .....</b>	<b>1355</b>
To Metaclass or Not to Metaclass	1356
Increasing Levels of “Magic”	1357
A Language of Hooks	1358
The Downside of “Helper” Functions	1359
Metaclasses Versus Class Decorators: Round 1	1361
The Metaclass Model	1364
Classes Are Instances of type	1364
Metaclasses Are Subclasses of Type	1366
Class Statement Protocol	1367
Declaring Metaclasses	1368
Declaration in 3.X	1369

Declaration in 2.X	1369
Metaclass Dispatch in Both 3.X and 2.X	1370
Coding Metaclasses	1370
A Basic Metaclass	1371
Customizing Construction and Initialization	1372
Other Metaclass Coding Techniques	1373
Inheritance and Instance	1378
Metaclass Versus Superclass	1381
Inheritance: The Full Story	1382
Metaclass Methods	1388
Metaclass Methods Versus Class Methods	1389
Operator Overloading in Metaclass Methods	1390
Example: Adding Methods to Classes	1391
Manual Augmentation	1391
Metaclass-Based Augmentation	1393
Metaclasses Versus Class Decorators: Round 2	1394
Example: Applying Decorators to Methods	1400
Tracing with Decoration Manually	1400
Tracing with Metaclasses and Decorators	1401
Applying Any Decorator to Methods	1403
Metaclasses Versus Class Decorators: Round 3 (and Last)	1404
Chapter Summary	1407
Test Your Knowledge: Quiz	1407
Test Your Knowledge: Answers	1408
<b>41. All Good Things .....</b>	<b>1409</b>
The Python Paradox	1409
On “Optional” Language Features	1410
Against Disquieting Improvements	1411
Complexity Versus Power	1412
Simplicity Versus Elitism	1412
Closing Thoughts	1413
Where to Go From Here	1414
Encore: Print Your Own Completion Certificate!	1414

---

## Part IX. Appendixes

<b>A. Installation and Configuration .....</b>	<b>1421</b>
Installing the Python Interpreter	1421
Is Python Already Present?	1421
Where to Get Python	1422
Installation Steps	1423

Configuring Python	1427
Python Environment Variables	1427
How to Set Configuration Options	1429
Python Command-Line Arguments	1432
Python 3.3 Windows Launcher Command Lines	1435
For More Help	1436
<b>B. The Python 3.3 Windows Launcher .....</b>	<b>1437</b>
The Unix Legacy	1437
The Windows Legacy	1438
Introducing the New Windows Launcher	1439
A Windows Launcher Tutorial	1441
Step 1: Using Version Directives in Files	1441
Step 2: Using Command-Line Version Switches	1444
Step 3: Using and Changing Defaults	1445
Pitfalls of the New Windows Launcher	1447
Pitfall 1: Unrecognized Unix <code>#!</code> Lines Fail	1447
Pitfall 2: The Launcher Defaults to 2.X	1448
Pitfall 3: The New PATH Extension Option	1449
Conclusions: A Net Win for Windows	1450
<b>C. Python Changes and This Book .....</b>	<b>1451</b>
Major 2.X/3.X Differences	1451
3.X Differences	1452
3.X-Only Extensions	1453
General Remarks: 3.X Changes	1454
Changes in Libraries and Tools	1454
Migrating to 3.X	1455
Fifth Edition Python Changes: 2.7, 3.2, 3.3	1456
Changes in Python 2.7	1456
Changes in Python 3.3	1457
Changes in Python 3.2	1458
Fourth Edition Python Changes: 2.6, 3.0, 3.1	1458
Changes in Python 3.1	1458
Changes in Python 3.0 and 2.6	1459
Specific Language Removals in 3.0	1460
Third Edition Python Changes: 2.3, 2.4, 2.5	1462
Earlier and Later Python Changes	1463
<b>D. Solutions to End-of-Part Exercises .....</b>	<b>1465</b>
Part I, Getting Started	1465
Part II, Types and Operations	1467
Part III, Statements and Syntax	1473

Part IV, Functions and Generators	1475
Part V, Modules and Packages	1485
Part VI, Classes and OOP	1489
Part VII, Exceptions and Tools	1497
<b>Index .....</b>	<b>1507</b>