

Table of Contents

Preface	1
Chapter 1: Getting started	9
Installing Python	10
Python Implementations	10
Jython	10
IronPython	11
PyPy	11
Other Implementations	11
Linux Installation	12
Package Installation	12
Compiling the Sources	13
Windows Installation	14
Installing Python	14
Installing MinGW	15
Installing MSYS	16
Mac OS X Installation	17
Package Installation	17
Compiling the Source	18
The Python Prompt	18
Customizing the Interactive Prompt	19
iPython: An Advanced Prompt	20
Installing setuptools	21
Understanding How It Works	21
setuptools Installation Using EasyInstall	22
Hooking MinGW into distutils	23
Working Environment	24
Using an Editor and Complementary Tools	24
Code Editor	25
Installing and Configuring Vim	25
Using Another Editor	27
Extra Binaries	28

Using an Integrated Development Environment	28
Installing Eclipse with PyDev	29
Summary	32
Chapter 2: Syntax Best Practices—Below the Class Level	33
List Comprehensions	34
Iterators and Generators	36
Generators	37
Coroutines	41
Generator Expressions	43
The itertools Module	44
islice: The Window Iterator	44
tee: The Back and Forth Iterator	45
groupby: The uniq Iterator	45
Other Functions	46
Decorators	47
How to Write a Decorator	48
Argument checking	50
Caching	52
Proxy	54
Context Provider	55
with and contextlib	56
The contextlib Module	58
Context Example	59
Summary	61
Chapter 3: Syntax Best Practices—Above the Class Level	63
Subclassing Built-in Types	63
Accessing Methods from Superclasses	65
Understanding Python's Method Resolution Order (MRO)	66
super Pitfalls	70
Mixing super and classic Calls	70
Heterogeneous Arguments	72
Best Practices	73
Descriptors and Properties	74
Descriptors	74
Introspection Descriptor	77
Meta-descriptor	79
Properties	81
Slots	83
Meta-programming	84
The__new__ Method	84
__metaclass__ Method	86
Summary	89

Chapter 4: Choosing Good Names	91
PEP 8 and Naming Best Practices	91
Naming Styles	92
Variables	92
Constants	92
Public and Private Variables	95
Functions and Methods	96
The Private Controversy	97
Special Methods	98
Arguments	98
Properties	99
Classes	99
Modules and Packages	99
Naming Guide	100
Use "has" or "is" Prefix for Boolean Elements	100
Use Plural for Elements That Are Sequences	100
Use Explicit Names for Dictionaries	101
Avoid Generic Names	101
Avoid Existing Names	101
Best Practices for Arguments	102
Build Arguments by Iterative Design	102
Trust the Arguments and Your Tests	103
Use *args and **kw Magic Arguments Carefully	104
Class Names	106
Module and Package Names	107
Working on APIs	107
Tracking Verbosity	108
Building the Namespace Tree	108
Splitting the Code	110
Using Eggs	111
Using a Deprecation Process	112
Useful Tools	113
Pylint	113
CloneDigger	115
Summary	116
Chapter 5: Writing a Package	117
A Common Pattern for All Packages	117
setup.py, the Script That Controls Everything	118
sdist	119
build and bdist	121
bdist_egg	122
install	123

How to Uninstall a Package	123
develop	124
test	124
register and upload	125
Creating a New Command	128
setup.py Usage Summary	129
Other Important Metadata	129
The Template-Based Approach	131
Python Paste	131
Creating Templates	133
Creating the Package Template	133
Development Cycle	138
Summary	141
Chapter 6: Writing an Application	143
Atomisator: An Introduction	143
Overall Picture	144
Working Environment	146
Adding a Test Runner	148
Adding a packages Structure	148
Writing the Packages	149
atomisator.parser	149
Creating the Initial Package	150
Creating the Initial doctest	151
Building the Test Environment	153
Writing the Code	153
atomisator.db	154
SQLAlchemy	154
Providing the APIs	158
atomisator.feed	159
atomisator.main	160
Distributing Atomisator	162
Dependencies between Packages	164
Summary	165
Chapter 7: Working with zc.buildout	167
zc.buildout Philosophy	168
Configuration File Structure	168
Minimum Configuration File	169
[buildout] Section Options	169
The buildout Command	170
Recipes	172
Notable Recipes	174
Creating Recipes	174
Atomisator buildout Environment	175

buildout Folder Structure	176
Going Further	177
Releasing and Distributing	178
Releasing the Packages	178
Adding a Release Configuration File	179
Building and Releasing the Application	180
Summary	181
Chapter 8: Managing Code	183
Version Control Systems	183
Centralized Systems	184
Distributed Systems	186
Distributed Strategies	188
Centralized or Distributed?	188
Mercurial	189
Project Management with Mercurial	193
Setting Up a Dedicated Folder	193
Configuring hgwebdir	194
Configuring Apache	195
Setting Up Authorizations	198
Setting Up the Client Side	199
Continuous Integration	200
Buildbot	201
Installing Buildbot	202
Hooking Buildbot and Mercurial	204
Hooking Apache and Buildbot	205
Summary	206
Chapter 9: Managing Life Cycle	207
Different Approaches	207
Waterfall Development Model	207
Spiral Development Model	208
Iterative Development Model	210
Defining a Life Cycle	210
Planning	212
Development	212
Global Debug	212
Release	213
Setting Up a Tracking System	213
Trac	213
Installation	215
Apache Settings	217
Permission Settings	218
Project Life Cycle with Trac	219
Planning	219

Development	221
Cleaning	221
Release	221
Summary	222
Chapter 10: Documenting Your Project	223
The Seven Rules of Technical Writing	223
Write in Two Steps	224
Target the Readership	225
Use a Simple Style	226
Limit the Scope of the Information	227
Use Realistic Code Examples	227
Use a Light but Sufficient Approach	228
Use Templates	228
A reStructuredText Primer	229
Section Structure	230
Lists	232
Inline Markup	232
Literal Block	232
Links	233
Building the Documentation	234
Building the Portfolio	234
Design	235
Usage	238
Operations	242
Make Your Own Portfolio	242
Building the Landscape	243
Producer's Layout	243
Consumer's Layout	244
Summary	249
Chapter 11: Test-Driven Development	251
I Don't Test	251
Test-Driven Development Principles	251
Preventing Software Regression	253
Improving Code Quality	254
Providing the Best Developer Documentation	254
Producing Robust Code Faster	255
What Kind of Tests?	255
Acceptance Tests	255
Unit Tests	256
Python Standard Test Tools	256
I Do Test	260
Unittest Pitfalls	260
Unittest Alternatives	261

nose	262
py.test	264
Fakes and Mocks	267
Building a Fake	268
Using Mocks	271
Document-Driven Development	273
Writing a Story	273
Summary	274
Chapter 12: Optimization: General Principles and Profiling Techniques	275
The Three Rules of Optimization	275
Make It Work First	275
Work from the User's Point of View	276
Keep the Code Readable(and thus maintainable)	277
Optimization Strategy	277
Find Another Culprit	278
Scale the Hardware	278
Write a Speed Test	279
Finding Bottlenecks	280
Profiling CPU Usage	280
Macro-Profiling	280
Micro-Profiling	284
Measuring Pystones	287
Profiling Memory Usage	288
How Python Deals with Memory	288
Profiling Memory	290
Profiling Network Usage	295
Summary	296
Chapter 13: Optimization: Solutions	297
Reducing the Complexity	298
Measuring Cyclomatic Complexity	298
Measuring the Big-O Notation	298
Simplifying	301
Searching in a List	301
Using a Set Instead of a List	302
Cut the External Calls, Reduce the Workload	303
Using Collections	303
Multithreading	306
What is Multithreading?	307
How Python Deals with Threads	307
When Should Threading Be Used?	309
Building Responsive Interfaces	309
Delegating Work	309

Multi-User Applications	310
Simple Example	310
Multiprocessing	314
Pyprocessing	315
Caching	317
Deterministic Caching	318
Non-Deterministic Caching	321
Pro-Active Caching	322
Memcached	322
Summary	323
Chapter 14: Useful Design Patterns	325
 Creational Patterns	325
Singleton	326
 Structural Patterns	328
Adapter	329
Interfaces	331
Proxy	332
Facade	333
 Behavioral Patterns	334
Observer	334
Visitor	336
Template	339
 Summary	341
Index	343
