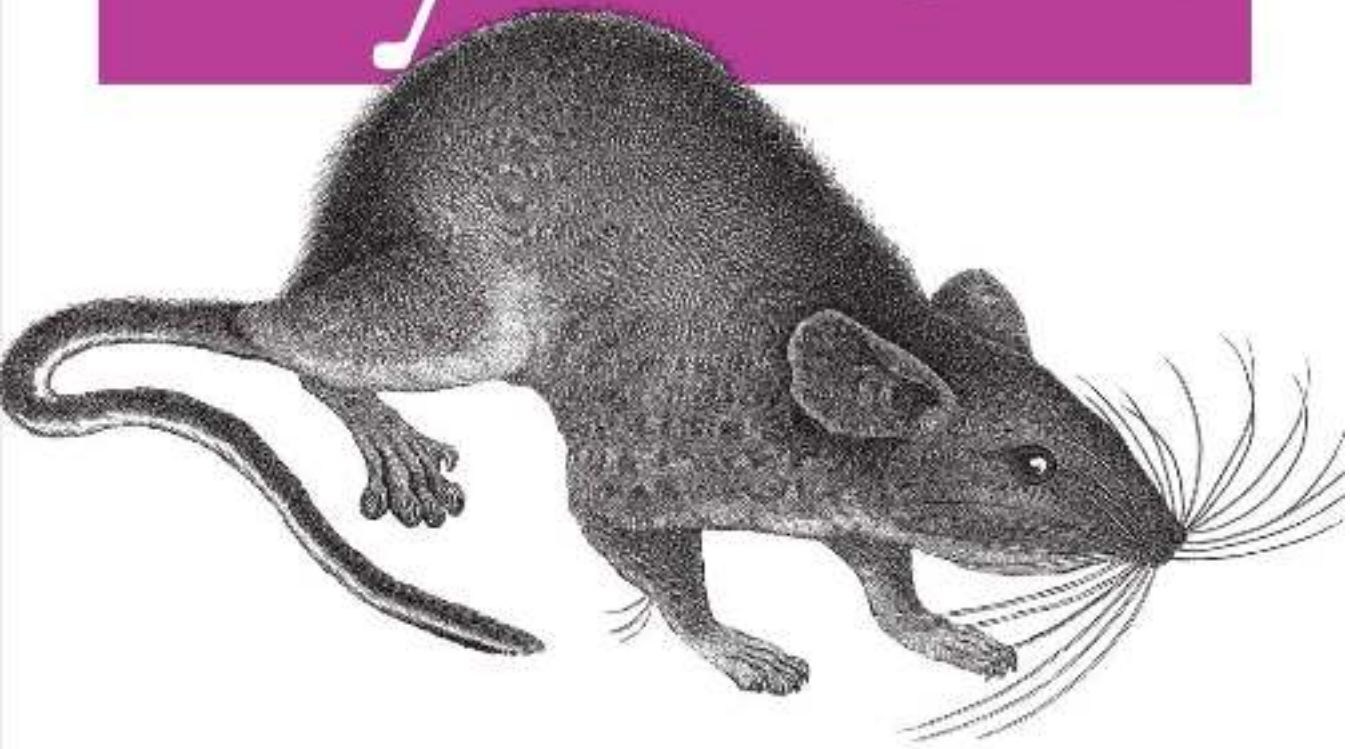


*Powerful Object-Oriented Programming*

**3rd Edition**  
Covers Python 2.5

*Learning*

# Python



**O'REILLY®**

*Mark Lutz*

---

# Learning Python

---

## Other resources from O'Reilly

---

**Related titles**    Programming Python                      Python Pocket Reference  
                         Python Cookbook™                      Twisted Network  
                         Python in a Nutshell                      Programming Essentials

---

**oreilly.com**    *oreilly.com* is more than a complete catalog of O'Reilly books. You'll also find links to news, events, articles, weblogs, sample chapters, and code examples.



---

*oreilynet.com* is the essential portal for developers interested in open and emerging technologies, including new platforms, programming languages, and operating systems.

---

**Conferences**    O'Reilly brings diverse innovators together to nurture the ideas that spark revolutionary industries. We specialize in documenting the latest tools and systems, translating the innovator's knowledge into useful skills for those in the trenches. Visit *conferences.oreilly.com* for our upcoming events.



---

Safari Bookshelf (*safari.oreilly.com*) is the premier online reference library for programmers and IT professionals. Conduct searches across more than 1,000 books. Subscribers can zero in on answers to time-critical questions in a matter of seconds. Read the books on your Bookshelf from cover to cover or simply flip to the page you need. Try it today for free.

---

THIRD EDITION

---

# Learning Python

*Mark Lutz*

O'REILLY®

Beijing • Cambridge • Farnham • Köln • Paris • Sebastopol • Taipei • Tokyo

---

## Learning Python, Third Edition

by Mark Lutz

Copyright © 2008 O'Reilly Media, Inc. All rights reserved.  
Printed in the United States of America.

Published by O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472.

O'Reilly books may be purchased for educational, business, or sales promotional use. Online editions are also available for most titles (*safari.oreilly.com*). For more information, contact our corporate/institutional sales department: (800) 998-9938 or *corporate@oreilly.com*.

**Editor:** Tatiana Apani

**Production Editor:** Sumita Mukherji

**Copyeditor:** Rachel Head

**Proofreader:** Sumita Mukherji

**Indexer:** Julie Hawks

**Cover Designer:** Karen Montgomery

**Interior Designer:** David Futato

**Illustrator:** Robert Romano

### Printing History:

March 1999: First Edition.

December 2003: Second Edition.

October 2007: Third Edition.

Nutshell Handbook, the Nutshell Handbook logo, and the O'Reilly logo are registered trademarks of O'Reilly Media, Inc., *Learning Python*, the image of a wood rat, and related trade dress are trademarks of O'Reilly Media, Inc.

Java™ is a trademark of Sun Microsystems, Inc. .NET is a registered trademark of Microsoft Corporation.

Many of the designations used by manufacturers and sellers to distinguish their products are claimed as trademarks. Where those designations appear in this book, and O'Reilly Media, Inc. was aware of a trademark claim, the designations have been printed in caps or initial caps.

While every precaution has been taken in the preparation of this book, the publisher and author assume no responsibility for errors or omissions, or for damages resulting from the use of the information contained herein.



This book uses RepKover™, a durable and flexible lay-flat binding.

ISBN-10: 0-596-51398-4

ISBN-13: 978-0-596-51398-6

[M]

---

*To Vera.  
You are my life.*



---

# Table of Contents

Preface .....	xxix
---------------	------

---

## Part I. Getting Started

<b>1. A Python Q&amp;A Session .....</b>	<b>3</b>
Why Do People Use Python?	3
Software Quality	5
Developer Productivity	5
Is Python a “Scripting Language”?	6
OK, but What’s the Downside?	7
Who Uses Python Today?	8
What Can I Do with Python?	9
Systems Programming	9
GUIs	9
Internet Scripting	10
Component Integration	10
Database Programming	11
Rapid Prototyping	11
Numeric and Scientific Programming	11
Gaming, Images, AI, XML, Robots, and More	12
What Are Python’s Technical Strengths?	12
It’s Object Oriented	12
It’s Free	13
It’s Portable	13
It’s Powerful	14
It’s Mixable	15



---

It's Easy to Use	15
It's Easy to Learn	17
It's Named After Monty Python	17
How Does Python Stack Up to Language X?	18
Chapter Summary	19
<b>Brain Builder</b>	<b>20</b>
Chapter Quiz	20
Quiz Answers	20
<b>2. How Python Runs Programs</b> .....	<b>22</b>
Introducing the Python Interpreter	22
Program Execution	24
The Programmer's View	24
Python's View	25
Byte code compilation	25
The Python Virtual Machine (PVM)	26
Performance implications	26
Development implications	27
Execution Model Variations	27
Python Implementation Alternatives	28
CPython	28
Jython	28
IronPython	29
Execution Optimization Tools	29
The Psyco just-in-time compiler	29
The Shedskin C++ translator	30
Frozen Binaries	31
Future Possibilities?	32
Chapter Summary	32
<b>Brain Builder</b>	<b>33</b>
Chapter Quiz	33
Quiz Answers	33

---

<b>3. How You Run Programs</b> .....	<b>34</b>
Interactive Coding	34
Using the Interactive Prompt	37
System Command Lines and Files	37
Using Command Lines and Files	40
Unix Executable Scripts (#!)	41
Clicking File Icons	42
Clicking Icons on Windows	42
The raw_input Trick	44
Other Icon-Click Limitations	45
Module Imports and Reloads	45
The Grander Module Story: Attributes	47
Modules and namespaces	49
import and reload Usage Notes	49
The IDLE User Interface	50
IDLE Basics	51
Using IDLE	52
Advanced IDLE Tools	54
Other IDEs	54
Embedding Calls	56
Frozen Binary Executables	56
Text Editor Launch Options	57
Other Launch Options	57
Future Possibilities?	57
Which Option Should I Use?	58
Chapter Summary	58
<b>Brain Builder</b>	<b>59</b>
Chapter Quiz	59
Quiz Answers	59
<b>Brain Builder: Part I Exercises</b>	<b>61</b>

---

## Part II. Types and Operations

<b>4. Introducing Python Object Types</b> .....	<b>65</b>
Why Use Built-in Types?	66
Python's Core Data Types	67
Numbers	68
Strings	69
Sequence Operations	70
Immutability	71
Type-Specific Methods	72
Getting Help	73
Other Ways to Code Strings	74
Pattern Matching	75
Lists	75
Sequence Operations	76
Type-Specific Operations	76
Bounds Checking	77
Nesting	77
List Comprehensions	78
Dictionaries	79
Mapping Operations	79
Nesting Revisited	80
Sorting Keys: for Loops	81
Iteration and Optimization	83
Missing Keys: if Tests	84
Tuples	85
Why Tuples?	85
Files	85
Other File-Like Tools	86
Other Core Types	87
How to Break Your Code's Flexibility	88
User-Defined Classes	88
And Everything Else	89
Chapter Summary	90
<b>Brain Builder</b>	<b>91</b>
Chapter Quiz	91
Quiz Answers	91

---

<b>5. Numbers</b> .....	<b>93</b>
Python Numeric Types	93
Numeric Literals	94
Built-in Numeric Tools and Extensions	95
Python Expression Operators	96
Mixed Operators Follow Operator Precedence	97
Parentheses Group Subexpressions	97
Mixed Types Are Converted Up	97
Preview: Operator Overloading	98
Numbers in Action	99
Variables and Basic Expressions	99
Numeric Display Formats	100
Division: Classic, Floor, and True	102
Bitwise Operations	103
Long Integers	103
Complex Numbers	104
Hexadecimal and Octal Notation	105
Other Built-in Numeric Tools	106
Other Numeric Types	107
Decimal Numbers	107
Sets	108
Booleans	109
Third-Party Extensions	110
Chapter Summary	110
<b>Brain Builder</b>	<b>111</b>
Chapter Quiz	111
Quiz Answers	111
<b>6. The Dynamic Typing Interlude</b> .....	<b>112</b>
The Case of the Missing Declaration Statements	112
Variables, Objects, and References	112
Types Live with Objects, Not Variables	114
Objects Are Garbage-Collected	115
Shared References	116
Shared References and In-Place Changes	118
Shared References and Equality	119
Dynamic Typing Is Everywhere	121

---

Chapter Summary	121
<b>Brain Builder</b>	<b>122</b>
Chapter Quiz	122
Quiz Answers	122
<b>7. Strings</b> .....	<b>123</b>
String Literals	124
Single- and Double-Quoted Strings Are the Same	125
Escape Sequences Represent Special Bytes	125
Raw Strings Suppress Escapes	127
Triple Quotes Code Multiline Block Strings	129
Unicode Strings Encode Larger Character Sets	130
Strings in Action	132
Basic Operations	132
Indexing and Slicing	133
Extended slicing: the third limit	135
String Conversion Tools	136
Character code conversions	138
Changing Strings	139
String Formatting	140
Advanced String Formatting	141
Dictionary-Based String Formatting	142
String Methods	143
String Method Examples: Changing Strings	144
String Method Examples: Parsing Text	146
Other Common String Methods in Action	147
The Original string Module	148
General Type Categories	149
Types Share Operation Sets by Categories	149
Mutable Types Can Be Changed In-Place	150
Chapter Summary	150
<b>Brain Builder</b>	<b>151</b>
Chapter Quiz	151
Quiz Answers	151

---

<b>8. Lists and Dictionaries</b> .....	<b>152</b>
Lists	152
Lists in Action	154
Basic List Operations	154
Indexing, Slicing, and Matrixes	155
Changing Lists In-Place	156
Index and slice assignments	156
List method calls	157
Other common list operations	159
Dictionaries	160
Dictionaries in Action	161
Basic Dictionary Operations	162
Changing Dictionaries In-Place	163
More Dictionary Methods	163
A Languages Table	165
Dictionary Usage Notes	166
Using dictionaries to simulate flexible lists	166
Using dictionaries for sparse data structures	167
Avoiding missing-key errors	167
Using dictionaries as “records”	168
Other ways to make dictionaries	169
Chapter Summary	170
<b>Brain Builder</b>	<b>171</b>
Chapter Quiz	171
Quiz Answers	171
<b>9. Tuples, Files, and Everything Else</b> .....	<b>172</b>
Tuples	172
Tuples in Action	173
Tuple syntax peculiarities: commas and parentheses	174
Conversions and immutability	174
Why Lists and Tuples?	175
Files	176
Opening Files	176
Using Files	177

---

Files in Action	178
Storing and parsing Python objects in files	178
Storing native Python objects with pickle	180
Storing and parsing packed binary data in files	181
Other File Tools	182
Type Categories Revisited	182
Object Flexibility	183
References Versus Copies	184
Comparisons, Equality, and Truth	186
The Meaning of True and False in Python	188
Python's Type Hierarchies	189
Other Types in Python	191
Built-in Type Gotchas	191
Assignment Creates References, Not Copies	191
Repetition Adds One Level Deep	192
Beware of Cyclic Data Structures	193
Immutable Types Can't Be Changed In-Place	193
Chapter Summary	193
<b>Brain Builder</b>	<b>195</b>
Chapter Quiz	195
Quiz Answers	195
<b>Brain Builder: Part II Exercises</b>	<b>196</b>

---

## Part III. Statements and Syntax

<b>10. Introducing Python Statements</b> .....	<b>201</b>
Python Program Structure Revisited	201
Python's Statements	202
A Tale of Two ifs	203
What Python Adds	204
What Python Removes	204
Parentheses are optional	204
End of line is end of statement	204
End of indentation is end of block	205
Why Indentation Syntax?	206
A Few Special Cases	208

---

Statement rule special cases	208
Block rule special case	209
A Quick Example: Interactive Loops	210
A Simple Interactive Loop	210
Doing Math on User Inputs	211
Handling Errors by Testing Inputs	212
Handling Errors with try Statements	213
Nesting Code Three Levels Deep	214
Chapter Summary	215
<b>Brain Builder</b>	<b>216</b>
Chapter Quiz	216
Quiz Answers	216
<b>11. Assignment, Expressions, and print</b> .....	<b>217</b>
Assignment Statements	217
Assignment Statement Forms	218
Sequence Assignments	219
Advanced sequence assignment patterns	220
Multiple-Target Assignments	222
Multiple-target assignment and shared references	222
Augmented Assignments	223
Augmented assignment and shared references	225
Variable Name Rules	225
Naming conventions	227
Names have no type, but objects do	227
Expression Statements	228
Expression Statements and In-Place Changes	229
print Statements	229
The Python “Hello World” Program	230
Redirecting the Output Stream	231
The print >> file Extension	232
Chapter Summary	234
<b>Brain Builder</b>	<b>235</b>
Chapter Quiz	235
Quiz Answers	235



---

<b>12. if Tests</b> .....	<b>236</b>
if Statements	236
General Format	236
Basic Examples	237
Multiway Branching	237
Python Syntax Rules	239
Block Delimiters	240
Statement Delimiters	241
A Few Special Cases	242
Truth Tests	243
The if/else Ternary Expression	244
Chapter Summary	246
<b>Brain Builder</b>	<b>247</b>
Chapter Quiz	247
Quiz Answers	247
<b>13. while and for Loops</b> .....	<b>248</b>
while Loops	248
General Format	249
Examples	249
break, continue, pass, and the Loop else	250
General Loop Format	250
Examples	251
pass	251
continue	251
break	252
else	252
More on the loop else clause	253
for Loops	254
General Format	254
Examples	256
Basic usage	256
Other data types	256
Tuple assignment in for	257
Nested for loops	257

---

Iterators: A First Look	258
File Iterators	260
Other Built-in Type Iterators	262
Other Iteration Contexts	263
User-Defined Iterators	264
Loop Coding Techniques	265
Counter Loops: while and range	265
Nonexhaustive Traversals: range	266
Changing Lists: range	267
Parallel Traversals: zip and map	268
Dictionary construction with zip	270
Generating Both Offsets and Items: enumerate	271
List Comprehensions: A First Look	272
List Comprehension Basics	272
Using List Comprehensions on Files	273
Extended List Comprehension Syntax	274
Chapter Summary	275
<b>Brain Builder</b>	<b>276</b>
Chapter Quiz	276
Quiz Answers	276
<b>14. The Documentation Interlude</b> .....	<b>278</b>
Python Documentation Sources	278
# Comments	279
The dir Function	279
Docstrings: <code>__doc__</code>	280
User-defined docstrings	281
Docstring standards	282
Built-in docstrings	282
PyDoc: The help Function	283
PyDoc: HTML Reports	285
Standard Manual Set	289
Web Resources	289
Published Books	290
Common Coding Gotchas	291

Chapter Summary	293
<b>Brain Builder</b>	<b>294</b>
Chapter Quiz	294
Quiz Answers	294
<b>Brain Builder: Part III Exercises</b>	<b>295</b>

---

## Part IV. Functions

<b>15. Function Basics</b> .....	<b>299</b>
Why Use Functions?	300
Coding Functions	300
def Statements	302
def Executes at Runtime	303
A First Example: Definitions and Calls	303
Definition	304
Calls	304
Polymorphism in Python	305
A Second Example: Intersecting Sequences	306
Definition	306
Calls	306
Polymorphism Revisited	307
Local Variables	308
Chapter Summary	308
<b>Brain Builder</b>	<b>309</b>
Chapter Quiz	309
Quiz Answers	309
<b>16. Scopes and Arguments</b> .....	<b>310</b>
Scope Rules	310
Python Scope Basics	311
Name Resolution: The LEGB Rule	312
Scope Example	314
The Built-in Scope	314
The global Statement	316
Minimize Global Variables	317
Minimize Cross-File Changes	318
Other Ways to Access Globals	319

---

Scopes and Nested Functions	320
Nested Scope Details	320
Nested Scope Examples	321
Factory functions	321
Retaining enclosing scopes' state with defaults	323
Nested scopes and lambdas	324
Scopes versus defaults with loop variables	324
Arbitrary scope nesting	326
Passing Arguments	326
Arguments and Shared References	327
Avoiding Mutable Argument Changes	329
Simulating Output Parameters	329
Special Argument-Matching Modes	330
Keyword and Default Examples	332
Keywords	332
Defaults	333
Arbitrary Arguments Examples	333
Collecting arguments	334
Unpacking arguments	334
Combining Keywords and Defaults	335
The min Wakeup Call	336
Full credit	336
Bonus points	337
The punch line	338
A More Useful Example: General Set Functions	338
Argument Matching: The Gritty Details	339
Chapter Summary	340
<b>Brain Builder</b>	<b>342</b>
Chapter Quiz	342
Quiz Answers	343
<b>17. Advanced Function Topics</b> .....	<b>344</b>
Anonymous Functions: lambda	344
lambda Expressions	344
Why Use lambda?	346
How (Not) to Obfuscate Your Python Code	347
Nested lambdas and Scopes	348

---

Applying Functions to Arguments	350
The apply Built-in	350
Passing keyword arguments	351
apply-Like Call Syntax	351
Mapping Functions over Sequences: map	352
Functional Programming Tools: filter and reduce	353
List Comprehensions Revisited: Mappings	355
List Comprehension Basics	355
Adding Tests and Nested Loops	356
List Comprehensions and Matrixes	358
Comprehending List Comprehensions	360
Iterators Revisited: Generators	360
Generator Function Example	362
Extended Generator Function Protocol: send Versus next	364
Iterators and Built-in Types	364
Generator Expressions: Iterators Meet List Comprehensions	365
Timing Iteration Alternatives	366
Function Design Concepts	369
Functions Are Objects: Indirect Calls	370
Function Gotchas	371
Local Names Are Detected Statically	372
Defaults and Mutable Objects	373
Functions Without returns	375
Enclosing Scope Loop Variables	375
Chapter Summary	375
<b>Brain Builder</b>	<b>377</b>
Chapter Quiz	377
Quiz Answers	377
<b>Brain Builder: Part IV Exercises</b>	<b>379</b>

---

## Part V. Modules

<b>18. Modules: The Big Picture</b> .....	<b>385</b>
Why Use Modules?	385
Python Program Architecture	386
How to Structure a Program	387
Imports and Attributes	387
Standard Library Modules	389

---

How Imports Work	389
1. Find It	390
The module search path	390
The sys.path list	392
Module file selection	393
Advanced module selection concepts	393
2. Compile It (Maybe)	394
3. Run It	394
Chapter Summary	395
<b>Brain Builder</b>	<b>397</b>
Chapter Quiz	397
Quiz Answers	397
<b>19. Module Coding Basics</b> .....	<b>398</b>
Module Creation	398
Module Usage	399
The import Statement	399
The from statement	400
The from * Statement	400
Imports Happen Only Once	400
import and from Are Assignments	401
Cross-File Name Changes	402
import and from Equivalence	402
Potential Pitfalls of the from Statement	403
When import is required	404
Module Namespaces	404
Files Generate Namespaces	405
Attribute Name Qualification	406
Imports Versus Scopes	407
Namespace Nesting	408
Reloading Modules	409
reload Basics	410
reload Example	411
Chapter Summary	412
<b>Brain Builder</b>	<b>414</b>
Chapter Quiz	414
Quiz Answers	414

---

<b>20. Module Packages</b> .....	<b>415</b>
Package Import Basics	415
Packages and Search Path Settings	416
Package <code>__init__.py</code> Files	416
Package Import Example	418
from Versus import with Packages	419
Why Use Package Imports?	420
A Tale of Three Systems	421
Chapter Summary	424
<b>Brain Builder</b>	<b>425</b>
Chapter Quiz	425
Quiz Answers	425
<b>21. Advanced Module Topics</b> .....	<b>426</b>
Data Hiding in Modules	426
Minimizing from * Damage: <code>_X</code> and <code>__all__</code>	426
Enabling Future Language Features	427
Mixed Usage Modes: <code>__name__</code> and <code>__main__</code>	428
Unit Tests with <code>__name__</code>	429
Changing the Module Search Path	430
The import as Extension	431
Relative Import Syntax	431
Why Relative Imports?	432
Module Design Concepts	434
Modules Are Objects: Metaprograms	435
Module Gotchas	437
Statement Order Matters in Top-Level Code	437
Importing Modules by Name String	438
from Copies Names but Doesn't Link	439
from * Can Obscure the Meaning of Variables	440
reload May Not Impact from Imports	440
reload, from, and Interactive Testing	441
reload Isn't Applied Transitively	442
Recursive from Imports May Not Work	443
Chapter Summary	444

---

<b>Brain Builder</b>	<b>445</b>
Chapter Quiz	445
Quiz Answers	445
<b>Brain Builder: Part V Exercises</b>	<b>446</b>

---

## Part VI. Classes and OOP

<b>22. OOP: The Big Picture</b> .....	<b>451</b>
Why Use Classes?	452
OOP from 30,000 Feet	453
Attribute Inheritance Search	453
Classes and Instances	455
Class Method Calls	456
Coding Class Trees	456
OOP Is About Code Reuse	459
Chapter Summary	462
<b>Brain Builder</b>	<b>463</b>
Chapter Quiz	463
Quiz Answers	463
<b>23. Class Coding Basics</b> .....	<b>465</b>
Classes Generate Multiple Instance Objects	465
Class Objects Provide Default Behavior	466
Instance Objects Are Concrete Items	466
A First Example	467
Classes Are Customized by Inheritance	469
A Second Example	470
Classes Are Attributes in Modules	471
Classes Can Intercept Python Operators	472
A Third Example	474
Why Use Operator Overloading?	475
The World's Simplest Python Class	476
Chapter Summary	478
<b>Brain Builder</b>	<b>479</b>
Chapter Quiz	479
Quiz Answers	479



- [The Children of Aataentsic: A History of the Huron People to 1660 book](#)
- [download The Public Relations of Everything: The Ancient, Modern and Postmodern Dramatic History of an Idea \(Routledge New Directions in Public Relations and Communication Research\)](#)
- **[click Un rastro de sirena pdf, azw \(kindle\), epub](#)**
- [read Kiselev's Geometry, Book 1: Planimetry](#)
- [download online Aesthetics, Method, and Epistemology \(Essential Works of Foucault, 1954-1984, Volume 2\)](#)
  
- <http://cavalldcartro.highlandagency.es/library/The-Children-of-Aataentsic--A-History-of-the-Huron-People-to-1660.pdf>
- <http://patrickvincitore.com/?ebooks/Science-Matters--Achieving-Scientific-Literacy.pdf>
- <http://test.markblaustein.com/library/Un-rastro-de-sirena.pdf>
- <http://patrickvincitore.com/?ebooks/The-Florentine-Emerald.pdf>
- <http://test.markblaustein.com/library/Newborn-Surgery--Arnold-Publication-.pdf>