

Python数据分析 (影印版)

Python for Data Analysis



O'REILLY®

東南大學出版社

Wes McKinney 著

Python数据分析 (影印版)

Python for Data Analysis

Wes McKinney 著



Beijing • Cambridge • Farnham • Köln • Sebastopol • Tokyo

O'Reilly Media, Inc. 授权东南大学出版社出版

南京 东南大学出版社

图书在版编目 (CIP) 数据

Python 数据分析: 英文/(美)麦金尼 (McKinney, W.)
著. —影印本. —南京: 东南大学出版社, 2013.5
书名原文: Python for Data Analysis
ISBN 978-7-5641-4204-9

I. ① P… II. ①麦… III. ①软件工具—程序设计—
英文 IV. ① TP311.56

中国版本图书馆 CIP 数据核字 (2013) 第 097341 号

江苏省版权局著作权合同登记

图字: 10-2013-131 号

©2012 by O'Reilly Media, Inc.

Reprint of the English Edition, jointly published by O'Reilly Media, Inc. and Southeast University Press, 2013. Authorized reprint of the original English edition, 2013 O'Reilly Media, Inc., the owner of all rights to publish and sell the same.

All rights reserved including the rights of reproduction in whole or in part in any form.

英文原版由 O'Reilly Media, Inc. 出版 2012。

英文影印版由东南大学出版社出版 2013。此影印版的出版和销售得到出版权和销售权的所有者——O'Reilly Media, Inc. 的许可。

版权所有, 未得书面许可, 本书的任何部分和全部不得以任何形式重制。

Python 数据分析 (影印版)

出版发行: 东南大学出版社
地 址: 南京四牌楼 2 号 邮编: 210096
出 版 人: 江建中
网 址: <http://www.seupress.com>
电子邮件: press@seupress.com
印 刷: 扬中市印刷有限公司
开 本: 787 毫米 × 980 毫米 16 开本
印 张: 29.5
字 数: 578 千字
版 次: 2013 年 5 月第 1 版
印 次: 2013 年 5 月第 1 次印刷
书 号: ISBN 978-7-5641-4204-9
定 价: 74.00 元 (册)

本社图书若有印装质量问题, 请直接与营销部联系。电话 (传真): 025-83791830

Preface

The scientific Python ecosystem of open source libraries has grown substantially over the last 10 years. By late 2011, I had long felt that the lack of centralized learning resources for data analysis and statistical applications was a stumbling block for new Python programmers engaged in such work. Key projects for data analysis (especially NumPy, IPython, matplotlib, and pandas) had also matured enough that a book written about them would likely not go out-of-date very quickly. Thus, I mustered the nerve to embark on this writing project. This is the book that I wish existed when I started using Python for data analysis in 2007. I hope you find it useful and are able to apply these tools productively in your work.

Conventions Used in This Book

The following typographical conventions are used in this book:

Italic

Indicates new terms, URLs, email addresses, filenames, and file extensions.

Constant width

Used for program listings, as well as within paragraphs to refer to program elements such as variable or function names, databases, data types, environment variables, statements, and keywords.

Constant width bold

Shows commands or other text that should be typed literally by the user.

Constant width italic

Shows text that should be replaced with user-supplied values or by values determined by context.



This icon signifies a tip, suggestion, or general note.



This icon indicates a warning or caution.

Using Code Examples

This book is here to help you get your job done. In general, you may use the code in this book in your programs and documentation. You do not need to contact us for permission unless you're reproducing a significant portion of the code. For example, writing a program that uses several chunks of code from this book does not require permission. Selling or distributing a CD-ROM of examples from O'Reilly books does require permission. Answering a question by citing this book and quoting example code does not require permission. Incorporating a significant amount of example code from this book into your product's documentation does require permission.

We appreciate, but do not require, attribution. An attribution usually includes the title, author, publisher, and ISBN. For example: "*Python for Data Analysis* by William Wesley McKinney (O'Reilly). Copyright 2012 William McKinney, 978-1-449-31979-3."

If you feel your use of code examples falls outside fair use or the permission given above, feel free to contact us at permissions@oreilly.com.

Safari® Books Online



Safari Books Online (www.safaribooksonline.com) is an on-demand digital library that delivers expert content in both book and video form from the world's leading authors in technology and business.

Technology professionals, software developers, web designers, and business and creative professionals use Safari Books Online as their primary resource for research, problem solving, learning, and certification training.

Safari Books Online offers a range of product mixes and pricing programs for organizations, government agencies, and individuals. Subscribers have access to thousands of books, training videos, and prepublication manuscripts in one fully searchable database from publishers like O'Reilly Media, Prentice Hall Professional, Addison-Wesley Professional, Microsoft Press, Sams, Que, Peachpit Press, Focal Press, Cisco Press, John Wiley & Sons, Syngress, Morgan Kaufmann, IBM Redbooks, Packt, Adobe Press, FT Press, Apress, Manning, New Riders, McGraw-Hill, Jones & Bartlett, Course Technology, and dozens more. For more information about Safari Books Online, please visit us online.

How to Contact Us

Please address comments and questions concerning this book to the publisher:

O'Reilly Media, Inc.
1005 Gravenstein Highway North
Sebastopol, CA 95472
800-998-9938 (in the United States or Canada)
707-829-0515 (international or local)
707-829-0104 (fax)

We have a web page for this book, where we list errata, examples, and any additional information. You can access this page at http://oreil.ly/python_for_data_analysis.

To comment or ask technical questions about this book, send email to bookquestions@oreilly.com.

For more information about our books, courses, conferences, and news, see our website at <http://www.oreilly.com>.

Find us on Facebook: <http://facebook.com/oreilly>

Follow us on Twitter: <http://twitter.com/oreillymedia>

Watch us on YouTube: <http://www.youtube.com/oreillymedia>

Table of Contents

Preface	xi
 1. Preliminaries	 1
What Is This Book About?	1
Why Python for Data Analysis?	2
Python as Glue	2
Solving the “Two-Language” Problem	2
Why Not Python?	3
Essential Python Libraries	3
NumPy	4
pandas	4
matplotlib	5
IPython	5
SciPy	6
Installation and Setup	6
Windows	7
Apple OS X	9
GNU/Linux	10
Python 2 and Python 3	11
Integrated Development Environments (IDEs)	11
Community and Conferences	12
Navigating This Book	12
Code Examples	13
Data for Examples	13
Import Conventions	13
Jargon	13
Acknowledgements	14
 2. Introductory Examples	 17
1.usa.gov data from bit.ly	17
Counting Time Zones in Pure Python	19

Counting Time Zones with pandas	21
MovieLens 1M Data Set	26
Measuring rating disagreement	30
US Baby Names 1880-2010	32
Analyzing Naming Trends	36
Conclusions and The Path Ahead	43
3. IPython: An Interactive Computing and Development Environment	45
IPython Basics	46
Tab Completion	47
Introspection	48
The %run Command	49
Executing Code from the Clipboard	50
Keyboard Shortcuts	52
Exceptions and Tracebacks	53
Magic Commands	54
Qt-based Rich GUI Console	55
Matplotlib Integration and Pylab Mode	56
Using the Command History	58
Searching and Reusing the Command History	58
Input and Output Variables	58
Logging the Input and Output	59
Interacting with the Operating System	60
Shell Commands and Aliases	60
Directory Bookmark System	62
Software Development Tools	62
Interactive Debugger	62
Timing Code: %time and %timeit	67
Basic Profiling: %prun and %run -p	68
Profiling a Function Line-by-Line	70
IPython HTML Notebook	72
Tips for Productive Code Development Using IPython	72
Reloading Module Dependencies	74
Code Design Tips	74
Advanced IPython Features	76
Making Your Own Classes IPython-friendly	76
Profiles and Configuration	77
Credits	78
4. NumPy Basics: Arrays and Vectorized Computation	79
The NumPy ndarray: A Multidimensional Array Object	80
Creating ndarrays	81
Data Types for ndarrays	83

Operations between Arrays and Scalars	85
Basic Indexing and Slicing	86
Boolean Indexing	89
Fancy Indexing	92
Transposing Arrays and Swapping Axes	93
Universal Functions: Fast Element-wise Array Functions	95
Data Processing Using Arrays	97
Expressing Conditional Logic as Array Operations	98
Mathematical and Statistical Methods	100
Methods for Boolean Arrays	101
Sorting	101
Unique and Other Set Logic	102
File Input and Output with Arrays	103
Storing Arrays on Disk in Binary Format	103
Saving and Loading Text Files	104
Linear Algebra	105
Random Number Generation	106
Example: Random Walks	108
Simulating Many Random Walks at Once	109
5. Getting Started with pandas	111
Introduction to pandas Data Structures	112
Series	112
DataFrame	115
Index Objects	120
Essential Functionality	122
Reindexing	122
Dropping entries from an axis	125
Indexing, selection, and filtering	125
Arithmetic and data alignment	128
Function application and mapping	132
Sorting and ranking	133
Axis indexes with duplicate values	136
Summarizing and Computing Descriptive Statistics	137
Correlation and Covariance	139
Unique Values, Value Counts, and Membership	141
Handling Missing Data	142
Filtering Out Missing Data	143
Filling in Missing Data	145
Hierarchical Indexing	147
Reordering and Sorting Levels	149
Summary Statistics by Level	150
Using a DataFrame's Columns	150

Other pandas Topics	151
Integer Indexing	151
Panel Data	152
6. Data Loading, Storage, and File Formats	155
Reading and Writing Data in Text Format	155
Reading Text Files in Pieces	160
Writing Data Out to Text Format	162
Manually Working with Delimited Formats	163
JSON Data	165
XML and HTML: Web Scraping	166
Binary Data Formats	171
Using HDF5 Format	171
Reading Microsoft Excel Files	172
Interacting with HTML and Web APIs	173
Interacting with Databases	174
Storing and Loading Data in MongoDB	176
7. Data Wrangling: Clean, Transform, Merge, Reshape	177
Combining and Merging Data Sets	177
Database-style DataFrame Merges	178
Merging on Index	182
Concatenating Along an Axis	185
Combining Data with Overlap	188
Reshaping and Pivoting	189
Reshaping with Hierarchical Indexing	190
Pivoting “long” to “wide” Format	192
Data Transformation	194
Removing Duplicates	194
Transforming Data Using a Function or Mapping	195
Replacing Values	196
Renaming Axis Indexes	197
Discretization and Binning	199
Detecting and Filtering Outliers	201
Permutation and Random Sampling	202
Computing Indicator/Dummy Variables	203
String Manipulation	205
String Object Methods	206
Regular expressions	207
Vectorized string functions in pandas	210
Example: USDA Food Database	212

8. Plotting and Visualization	219
A Brief matplotlib API Primer	219
Figures and Subplots	220
Colors, Markers, and Line Styles	224
Ticks, Labels, and Legends	225
Annotations and Drawing on a Subplot	228
Saving Plots to File	231
matplotlib Configuration	231
Plotting Functions in pandas	232
Line Plots	232
Bar Plots	235
Histograms and Density Plots	238
Scatter Plots	239
Plotting Maps: Visualizing Haiti Earthquake Crisis Data	241
Python Visualization Tool Ecosystem	247
Chaco	248
mayavi	248
Other Packages	249
The Future of Visualization Tools?	249
 9. Data Aggregation and Group Operations	 251
GroupBy Mechanics	252
Iterating Over Groups	255
Selecting a Column or Subset of Columns	256
Grouping with Dicts and Series	257
Grouping with Functions	258
Grouping by Index Levels	259
Data Aggregation	259
Column-wise and Multiple Function Application	262
Returning Aggregated Data in “unindexed” Form	264
Group-wise Operations and Transformations	264
Apply: General split-apply-combine	266
Quantile and Bucket Analysis	268
Example: Filling Missing Values with Group-specific Values	270
Example: Random Sampling and Permutation	271
Example: Group Weighted Average and Correlation	273
Example: Group-wise Linear Regression	274
Pivot Tables and Cross-Tabulation	275
Cross-Tabulations: Crosstab	277
Example: 2012 Federal Election Commission Database	278
Donation Statistics by Occupation and Employer	280
Bucketing Donation Amounts	283
Donation Statistics by State	285

10. Time Series	289
Date and Time Data Types and Tools	290
Converting between string and datetime	291
Time Series Basics	293
Indexing, Selection, Subsetting	294
Time Series with Duplicate Indices	296
Date Ranges, Frequencies, and Shifting	297
Generating Date Ranges	298
Frequencies and Date Offsets	299
Shifting (Leading and Lagging) Data	301
Time Zone Handling	303
Localization and Conversion	304
Operations with Time Zone-aware Timestamp Objects	305
Operations between Different Time Zones	306
Periods and Period Arithmetic	307
Period Frequency Conversion	308
Quarterly Period Frequencies	309
Converting Timestamps to Periods (and Back)	311
Creating a PeriodIndex from Arrays	312
Resampling and Frequency Conversion	312
Downsampling	314
Upsampling and Interpolation	316
Resampling with Periods	318
Time Series Plotting	319
Moving Window Functions	320
Exponentially-weighted functions	324
Binary Moving Window Functions	324
User-Defined Moving Window Functions	326
Performance and Memory Usage Notes	327
 11. Financial and Economic Data Applications	 329
Data Munging Topics	329
Time Series and Cross-Section Alignment	330
Operations with Time Series of Different Frequencies	332
Time of Day and “as of” Data Selection	334
Splicing Together Data Sources	336
Return Indexes and Cumulative Returns	338
Group Transforms and Analysis	340
Group Factor Exposures	342
Decile and Quartile Analysis	343
More Example Applications	345
Signal Frontier Analysis	345
Future Contract Rolling	347

Rolling Correlation and Linear Regression	350
12. Advanced NumPy	353
ndarray Object Internals	353
NumPy dtype Hierarchy	354
Advanced Array Manipulation	355
Reshaping Arrays	355
C versus Fortran Order	356
Concatenating and Splitting Arrays	357
Repeating Elements: Tile and Repeat	360
Fancy Indexing Equivalents: Take and Put	361
Broadcasting	362
Broadcasting Over Other Axes	364
Setting Array Values by Broadcasting	367
Advanced ufunc Usage	367
ufunc Instance Methods	368
Custom ufuncs	370
Structured and Record Arrays	370
Nested dtypes and Multidimensional Fields	371
Why Use Structured Arrays?	372
Structured Array Manipulations: numpy.lib.recfunctions	372
More About Sorting	373
Indirect Sorts: argsort and lexsort	374
Alternate Sort Algorithms	375
numpy.searchsorted: Finding elements in a Sorted Array	376
NumPy Matrix Class	377
Advanced Array Input and Output	379
Memory-mapped Files	379
HDF5 and Other Array Storage Options	380
Performance Tips	380
The Importance of Contiguous Memory	381
Other Speed Options: Cython, f2py, C	382
Appendix: Python Language Essentials	385
Index	433

Preliminaries

What Is This Book About?

This book is concerned with the nuts and bolts of manipulating, processing, cleaning, and crunching data in Python. It is also a practical, modern introduction to scientific computing in Python, tailored for data-intensive applications. This is a book about the parts of the Python language and libraries you'll need to effectively solve a broad set of data analysis problems. This book is *not* an exposition on analytical methods using Python as the implementation language.

When I say “data”, what am I referring to exactly? The primary focus is on *structured data*, a deliberately vague term that encompasses many different common forms of data, such as

- Multidimensional arrays (matrices)
- Tabular or spreadsheet-like data in which each column may be a different type (string, numeric, date, or otherwise). This includes most kinds of data commonly stored in relational databases or tab- or comma-delimited text files
- Multiple tables of data interrelated by key columns (what would be primary or foreign keys for a SQL user)
- Evenly or unevenly spaced time series

This is by no means a complete list. Even though it may not always be obvious, a large percentage of data sets can be transformed into a structured form that is more suitable for analysis and modeling. If not, it may be possible to extract features from a data set into a structured form. As an example, a collection of news articles could be processed into a word frequency table which could then be used to perform sentiment analysis.

Most users of spreadsheet programs like Microsoft Excel, perhaps the most widely used data analysis tool in the world, will not be strangers to these kinds of data.

Why Python for Data Analysis?

For many people (myself among them), the Python language is easy to fall in love with. Since its first appearance in 1991, Python has become one of the most popular dynamic, programming languages, along with Perl, Ruby, and others. Python and Ruby have become especially popular in recent years for building websites using their numerous web frameworks, like Rails (Ruby) and Django (Python). Such languages are often called *scripting* languages as they can be used to write quick-and-dirty small programs, or *scripts*. I don't like the term "scripting language" as it carries a connotation that they cannot be used for building mission-critical software. Among interpreted languages Python is distinguished by its large and active *scientific computing* community. Adoption of Python for scientific computing in both industry applications and academic research has increased significantly since the early 2000s.

For data analysis and interactive, exploratory computing and data visualization, Python will inevitably draw comparisons with the many other domain-specific open source and commercial programming languages and tools in wide use, such as R, MATLAB, SAS, Stata, and others. In recent years, Python's improved library support (primarily pandas) has made it a strong alternative for data manipulation tasks. Combined with Python's strength in general purpose programming, it is an excellent choice as a single language for building data-centric applications.

Python as Glue

Part of Python's success as a scientific computing platform is the ease of integrating C, C++, and FORTRAN code. Most modern computing environments share a similar set of legacy FORTRAN and C libraries for doing linear algebra, optimization, integration, fast fourier transforms, and other such algorithms. The same story has held true for many companies and national labs that have used Python to glue together 30 years' worth of legacy software.

Most programs consist of small portions of code where most of the time is spent, with large amounts of "glue code" that doesn't run often. In many cases, the execution time of the glue code is insignificant; effort is most fruitfully invested in optimizing the computational bottlenecks, sometimes by moving the code to a lower-level language like C.

In the last few years, the Cython project (<http://cython.org>) has become one of the preferred ways of both creating fast compiled extensions for Python and also interfacing with C and C++ code.

Solving the "Two-Language" Problem

In many organizations, it is common to research, prototype, and test new ideas using a more domain-specific computing language like MATLAB or R then later port those

ideas to be part of a larger production system written in, say, Java, C#, or C++. What people are increasingly finding is that Python is a suitable language not only for doing research and prototyping but also building the production systems, too. I believe that more and more companies will go down this path as there are often significant organizational benefits to having both scientists and technologists using the same set of programmatic tools.

Why Not Python?

While Python is an excellent environment for building computationally-intensive scientific applications and building most kinds of general purpose systems, there are a number of uses for which Python may be less suitable.

As Python is an interpreted programming language, in general most Python code will run substantially slower than code written in a compiled language like Java or C++. As *programmer time* is typically more valuable than *CPU time*, many are happy to make this tradeoff. However, in an application with very low latency requirements (for example, a high frequency trading system), the time spent programming in a lower-level, lower-productivity language like C++ to achieve the maximum possible performance might be time well spent.

Python is not an ideal language for highly concurrent, multithreaded applications, particularly applications with many CPU-bound threads. The reason for this is that it has what is known as the *global interpreter lock* (GIL), a mechanism which prevents the interpreter from executing more than one Python bytecode instruction at a time. The technical reasons for why the GIL exists are beyond the scope of this book, but as of this writing it does not seem likely that the GIL will disappear anytime soon. While it is true that in many big data processing applications, a cluster of computers may be required to process a data set in a reasonable amount of time, there are still situations where a single-process, multithreaded system is desirable.

This is not to say that Python cannot execute truly multithreaded, parallel code; that code just cannot be executed in a single Python process. As an example, the Cython project features easy integration with OpenMP, a C framework for parallel computing, in order to parallelize loops and thus significantly speed up numerical algorithms.

Essential Python Libraries

For those who are less familiar with the scientific Python ecosystem and the libraries used throughout the book, I present the following overview of each library.

NumPy

NumPy, short for Numerical Python, is the foundational package for scientific computing in Python. The majority of this book will be based on NumPy and libraries built on top of NumPy. It provides, among other things:

- A fast and efficient multidimensional array object *ndarray*
- Functions for performing element-wise computations with arrays or mathematical operations between arrays
- Tools for reading and writing array-based data sets to disk
- Linear algebra operations, Fourier transform, and random number generation
- Tools for integrating connecting C, C++, and Fortran code to Python

Beyond the fast array-processing capabilities that NumPy adds to Python, one of its primary purposes with regards to data analysis is as the primary container for data to be passed between algorithms. For numerical data, NumPy arrays are a much more efficient way of storing and manipulating data than the other built-in Python data structures. Also, libraries written in a lower-level language, such as C or Fortran, can operate on the data stored in a NumPy array without copying any data.

pandas

pandas provides rich data structures and functions designed to make working with structured data fast, easy, and expressive. It is, as you will see, one of the critical ingredients enabling Python to be a powerful and productive data analysis environment. The primary object in pandas that will be used in this book is the *DataFrame*, a two-dimensional tabular, column-oriented data structure with both row and column labels:

```
>>> frame
   total_bill  tip  sex  smoker  day  time  size
1    16.99    1.01 Female    No   Sun  Dinner    2
2    10.34    1.66  Male    No   Sun  Dinner    3
3    21.01    3.5  Male    No   Sun  Dinner    3
4    23.68    3.31 Male    No   Sun  Dinner    2
5    24.59    3.61 Female  No   Sun  Dinner    4
6    25.29    4.71  Male    No   Sun  Dinner    4
7     8.77     2  Male    No   Sun  Dinner    2
8    26.88    3.12 Male    No   Sun  Dinner    4
9    15.04    1.96 Male    No   Sun  Dinner    2
10   14.78    3.23 Male    No   Sun  Dinner    2
```

pandas combines the high performance array-computing features of NumPy with the flexible data manipulation capabilities of spreadsheets and relational databases (such as SQL). It provides sophisticated indexing functionality to make it easy to reshape, slice and dice, perform aggregations, and select subsets of data. pandas is the primary tool that we will use in this book.