

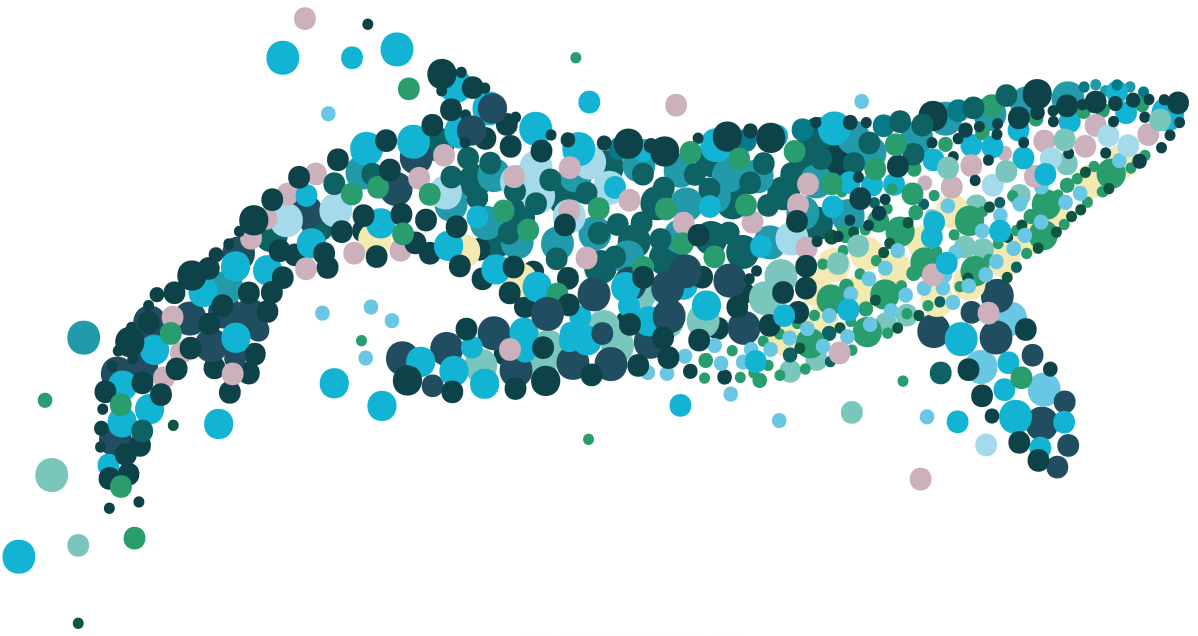
GLOBAL
EDITION



Intro to Python[®]

for Computer Science and Data Science

Paul Deitel • Harvey Deitel



Learning to Program with AI, Big Data and the Cloud



Digital Resources for Students

Your eBook provides 12-month access to digital resources that may include VideoNotes (step-by-step video tutorials on programming concepts), source code, and more. Refer to the preface in the textbook for a detailed list of resources.

Follow the instructions below to register for the Companion Website for Paul and Harvey Deitel's *Intro to Python®: for Computer Science and Data Science*, Global Edition.

1. Go to www.pearsonglobaleditions.com.
2. Enter the title of your textbook or browse by author name.
3. Click Companion Website.
4. Click Register and follow the on-screen instructions to create a login name and password.

ISSPCD-WAHOO-DIARY-KALPA-FRACK-OOSSE

Use the login name and password you created during registration to start using the online resources that accompany your textbook.

IMPORTANT:

This access code can only be used once. This subscription is valid for 12 months upon activation and is not transferable.

For technical support, go to <https://support.pearson.com/getsupport>.

Intro to Python® for Computer Science and Data Science

Learning to Program with AI, Big Data and the Cloud
by Paul Deitel & Harvey Deitel

PART I

CS: Python Fundamentals Quickstart

CS 1. Introduction to Computers and Python

DS Intro: AI—at the Intersection of CS and DS

CS 2. Introduction to Python Programming

DS Intro: Basic Descriptive Stats

CS 3. Control Statements and Program Development

DS Intro: Measures of Central Tendency—Mean, Median, Mode

CS 4. Functions

DS Intro: Basic Statistics—Measures of Dispersion

CS 5. Lists and Tuples

DS Intro: Simulation and Static Visualization

1. Chapters 1–11 marked CS are traditional Python programming and computer-science topics.

2. Light-tinted bottom boxes in Chapters 1–10 marked DS Intro are brief, friendly introductions to data-science topics.

PART 2

CS: Python Data Structures, Strings and Files

CS 6. Dictionaries and Sets

DS Intro: Simulation and Dynamic Visualization

CS 7. Array-Oriented Programming with NumPy

High-Performance NumPy Arrays

DS Intro:
Pandas Series and DataFrames

CS 8. Strings: A Deeper Look

Includes Regular Expressions

DS Intro: Pandas,
Regular Expressions and
Data Wrangling

CS 9. Files and Exceptions

DS Intro: Loading Datasets from CSV Files into Pandas DataFrames

3. Chapters 12–17 marked DS are Python-based, AI, big data and cloud chapters, each containing several full-implementation studies.

4. Functional-style programming is integrated book wide.

PART 3

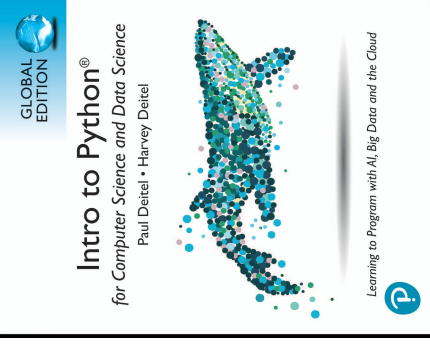
CS: Python High-End Topics

CS 10. Object-Oriented Programming

DS Intro: Time Series and Simple Linear Regression

CS 11. Computer Science Thinking: Recursion, Searching, Sorting and Big O

CS and DS Other Topics Blog



5. Preface explains the dependencies among the chapters.

6. Visualizations throughout.

PART 4

AI, Big Data and Cloud Case Studies

DS 12. Natural Language Processing (NLP)

Web Scraping in the Exercises

DS 13. Data Mining Twitter®

Sentiment Analysis; JSON and Web Services

DS 14. IBM Watson® and Cognitive Computing

DS 15. Machine Learning: Classification, Regression and Clustering

DS 16. Deep Learning
Convolutional and Recurrent Neural Networks; Reinforcement Learning in the Exercises

DS 17. Big Data: Hadoop®, Spark™, NoSQL and IoT

7. CS courses may cover more of the Python chapters and less of the DS content. Vice versa for Data Science courses.

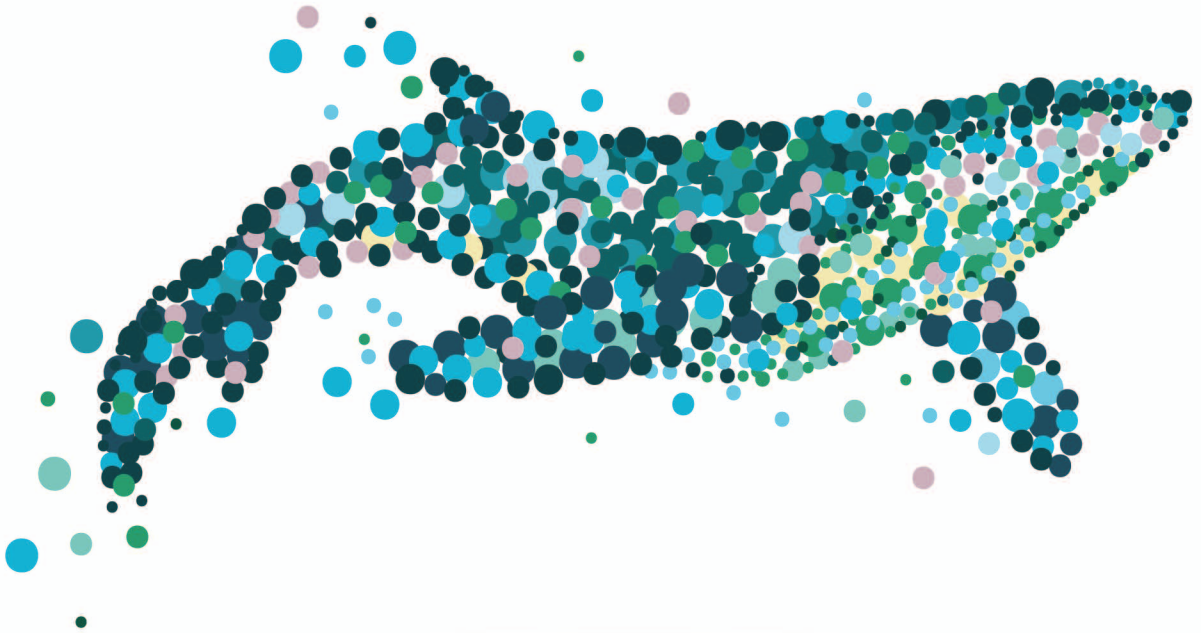
8. We put Chapter 5 in Part 1. It's also a natural fit with Part 2.

Questions? deitel@deitel.com

This page is intentionally left blank

Intro to Python[®]

for Computer Science and Data Science



Cover Designer: Straive

Cover Art: ©Yuriy2012/Shutterstock

Pearson Education Limited

KAO Two
KAO Park
Hockham Way
Harlow
CM17 9SR
United Kingdom

and Associated Companies throughout the world

Visit us on the World Wide Web at: www.pearsonglobaleditions.com

Please contact <https://support.pearson.com/getsupport/s/contactsupport> with any queries on this content.

© Pearson Education Limited 2022

The rights of Paul Deitel and Harvey Deitel to be identified as the authors of this work have been asserted by them in accordance with the Copyright, Designs and Patents Act 1988.

Authorized adaptation from the United States edition, entitled Intro to Python for Computer Science and Data Science: Learning to Program with AI, Big Data and The Cloud, ISBN 978-0-13-540467-6 by Paul Deitel and Harvey Deitel published by Pearson Education © 2020.

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, electronic, mechanical, photocopying, recording or otherwise, without either the prior written permission of the publisher or a license permitting restricted copying in the United Kingdom issued by the Copyright Licensing Agency Ltd, Saffron House, 6–10 Kirby Street, London EC1N 8TS. For information regarding permissions, request forms and the appropriate contacts within the Pearson Education Global Rights & Permissions department, please visit www.pearsoned.com/permissions/.

All trademarks used herein are the property of their respective owners. The use of any trademark in this text does not vest in the author or publisher any trademark ownership rights in such trademarks, nor does the use of such trademarks imply any affiliation with or endorsement of this book by such owners.

Attributions of third-party content appear on the appropriate page within the text.

PEARSON, ALWAYS LEARNING, and MYLAB are exclusive trademarks owned by Pearson Education, Inc. or its affiliates in the U.S. and/or other countries. Deitel and the double-thumbs-up bug are registered trademarks of Deitel and Associates, Inc.

Unless otherwise indicated herein, any third-party trademarks that may appear in this work are the property of their respective owners and any references to third-party trademarks, logos or other trade dress are for demonstrative or descriptive purposes only. Such references are not intended to imply any sponsorship, endorsement, authorization, or promotion of Pearson's products by the owners of such marks, or any relationship between the owner and Pearson Education, Inc. or its affiliates, authors, licensees, or distributors.

This eBook is a standalone product and may or may not include all assets that were part of the print version. It also does not provide access to other Pearson digital products like MyLab and Mastering. The publisher reserves the right to remove any material in this eBook at any time.

British Library Cataloguing-in-Publication Data

A catalogue record for this book is available from the British Library

ISBN 10: 1-292-36490-4 (print)

ISBN 13: 978-1-292-36490-2 (print)

ISBN 13: 978-1-292-36493-3 (uPDF eBook)

GLOBAL
EDITION



Intro to Python[®]

for Computer Science and Data Science

Paul Deitel • Harvey Deitel



Learning to Program with AI, Big Data and the Cloud



*In Memory of Marvin Minsky,
a founding father of
artificial intelligence*

*It was a privilege to be your student in two
artificial-intelligence graduate courses at M.I.T.
You inspired your students to think beyond limits.*

Harvey Deitel



Contents

Preface	19
Before You Begin	45
I Introduction to Computers and Python	49
1.1 Introduction	50
1.2 Hardware and Software	51
1.2.1 Moore's Law	52
1.2.2 Computer Organization	52
1.3 Data Hierarchy	54
1.4 Machine Languages, Assembly Languages and High-Level Languages	57
1.5 Introduction to Object Technology	58
1.6 Operating Systems	61
1.7 Python	64
1.8 It's the Libraries!	66
1.8.1 Python Standard Library	66
1.8.2 Data-Science Libraries	66
1.9 Other Popular Programming Languages	68
1.10 Test-Drives: Using IPython and Jupyter Notebooks	69
1.10.1 Using IPython Interactive Mode as a Calculator	69
1.10.2 Executing a Python Program Using the IPython Interpreter	71
1.10.3 Writing and Executing Code in a Jupyter Notebook	72
1.11 Internet and World Wide Web	77
1.11.1 Internet: A Network of Networks	77
1.11.2 World Wide Web: Making the Internet User-Friendly	78
1.11.3 The Cloud	78
1.11.4 Internet of Things	79
1.12 Software Technologies	80
1.13 How Big Is Big Data?	81
1.13.1 Big Data Analytics	86
1.13.2 Data Science and Big Data Are Making a Difference: Use Cases	87
1.14 Case Study—A Big-Data Mobile Application	88
1.15 Intro to Data Science: Artificial Intelligence—at the Intersection of CS and Data Science	90

2	Introduction to Python Programming	97
2.1	Introduction	98
2.2	Variables and Assignment Statements	98
2.3	Arithmetic	100
2.4	Function <code>print</code> and an Intro to Single- and Double-Quoted Strings	104
2.5	Triple-Quoted Strings	106
2.6	Getting Input from the User	107
2.7	Decision Making: The <code>if</code> Statement and Comparison Operators	109
2.8	Objects and Dynamic Typing	114
2.9	Intro to Data Science: Basic Descriptive Statistics	116
2.10	Wrap-Up	118
3	Control Statements and Program Development	121
3.1	Introduction	122
3.2	Algorithms	122
3.3	Pseudocode	123
3.4	Control Statements	123
3.5	<code>if</code> Statement	126
3.6	<code>if...else</code> and <code>if...elif...else</code> Statements	128
3.7	<code>while</code> Statement	133
3.8	<code>for</code> Statement	134
	3.8.1 Iterables, Lists and Iterators	136
	3.8.2 Built-In <code>range</code> Function	136
3.9	Augmented Assignments	137
3.10	Program Development: Sequence-Controlled Repetition	138
	3.10.1 Requirements Statement	138
	3.10.2 Pseudocode for the Algorithm	138
	3.10.3 Coding the Algorithm in Python	139
	3.10.4 Introduction to Formatted Strings	140
3.11	Program Development: Sentinel-Controlled Repetition	141
3.12	Program Development: Nested Control Statements	145
3.13	Built-In Function <code>range</code> : A Deeper Look	149
3.14	Using Type <code>Decimal</code> for Monetary Amounts	150
3.15	<code>break</code> and <code>continue</code> Statements	153
3.16	Boolean Operators <code>and</code> , <code>or</code> and <code>not</code>	154
3.17	Intro to Data Science: Measures of Central Tendency— Mean, Median and Mode	157
3.18	Wrap-Up	159
4	Functions	167
4.1	Introduction	168
4.2	Defining Functions	168
4.3	Functions with Multiple Parameters	171
4.4	Random-Number Generation	173

4.5	Case Study: A Game of Chance	176
4.6	Python Standard Library	179
4.7	math Module Functions	180
4.8	Using IPython Tab Completion for Discovery	181
4.9	Default Parameter Values	183
4.10	Keyword Arguments	184
4.11	Arbitrary Argument Lists	184
4.12	Methods: Functions That Belong to Objects	186
4.13	Scope Rules	186
4.14	import: A Deeper Look	188
4.15	Passing Arguments to Functions: A Deeper Look	190
4.16	Function-Call Stack	193
4.17	Functional-Style Programming	194
4.18	Intro to Data Science: Measures of Dispersion	196
4.19	Wrap-Up	198

5 Sequences: Lists and Tuples 203

5.1	Introduction	204
5.2	Lists	204
5.3	Tuples	209
5.4	Unpacking Sequences	211
5.5	Sequence Slicing	214
5.6	del Statement	217
5.7	Passing Lists to Functions	219
5.8	Sorting Lists	220
5.9	Searching Sequences	222
5.10	Other List Methods	224
5.11	Simulating Stacks with Lists	226
5.12	List Comprehensions	227
5.13	Generator Expressions	229
5.14	Filter, Map and Reduce	230
5.15	Other Sequence Processing Functions	233
5.16	Two-Dimensional Lists	235
5.17	Intro to Data Science: Simulation and Static Visualizations	239
	5.17.1 Sample Graphs for 600, 60,000 and 6,000,000 Die Rolls	239
	5.17.2 Visualizing Die-Roll Frequencies and Percentages	241
5.18	Wrap-Up	247

6 Dictionaries and Sets 257

6.1	Introduction	258
6.2	Dictionaries	258
	6.2.1 Creating a Dictionary	258
	6.2.2 Iterating through a Dictionary	260
	6.2.3 Basic Dictionary Operations	260

6.2.4	Dictionary Methods keys and values	262
6.2.5	Dictionary Comparisons	264
6.2.6	Example: Dictionary of Student Grades	265
6.2.7	Example: Word Counts	266
6.2.8	Dictionary Method update	268
6.2.9	Dictionary Comprehensions	268
6.3	Sets	269
6.3.1	Comparing Sets	271
6.3.2	Mathematical Set Operations	273
6.3.3	Mutable Set Operators and Methods	274
6.3.4	Set Comprehensions	276
6.4	Intro to Data Science: Dynamic Visualizations	276
6.4.1	How Dynamic Visualization Works	276
6.4.2	Implementing a Dynamic Visualization	279
6.5	Wrap-Up	282

7 Array-Oriented Programming with NumPy **287**

7.1	Introduction	288
7.2	Creating arrays from Existing Data	289
7.3	array Attributes	290
7.4	Filling arrays with Specific Values	292
7.5	Creating arrays from Ranges	292
7.6	List vs. array Performance: Introducing <code>%timeit</code>	294
7.7	array Operators	296
7.8	NumPy Calculation Methods	298
7.9	Universal Functions	300
7.10	Indexing and Slicing	302
7.11	Views: Shallow Copies	304
7.12	Deep Copies	306
7.13	Reshaping and Transposing	307
7.14	Intro to Data Science: pandas Series and DataFrames	310
	7.14.1 pandas Series	310
	7.14.2 DataFrames	315
7.15	Wrap-Up	323

8 Strings: A Deeper Look **331**

8.1	Introduction	332
8.2	Formatting Strings	333
	8.2.1 Presentation Types	333
	8.2.2 Field Widths and Alignment	334
	8.2.3 Numeric Formatting	335
	8.2.4 String's format Method	336
8.3	Concatenating and Repeating Strings	337
8.4	Stripping Whitespace from Strings	338

8.5	Changing Character Case	339
8.6	Comparison Operators for Strings	340
8.7	Searching for Substrings	340
8.8	Replacing Substrings	342
8.9	Splitting and Joining Strings	342
8.10	Characters and Character-Testing Methods	345
8.11	Raw Strings	346
8.12	Introduction to Regular Expressions	347
	8.12.1 re Module and Function <code>fullmatch</code>	348
	8.12.2 Replacing Substrings and Splitting Strings	351
	8.12.3 Other Search Functions; Accessing Matches	352
8.13	Intro to Data Science: Pandas, Regular Expressions and Data Munging	355
8.14	Wrap-Up	360

9 Files and Exceptions 367

9.1	Introduction	368
9.2	Files	369
9.3	Text-File Processing	369
	9.3.1 Writing to a Text File: Introducing the <code>with</code> Statement	370
	9.3.2 Reading Data from a Text File	371
9.4	Updating Text Files	373
9.5	Serialization with JSON	375
9.6	Focus on Security: <code>pickle</code> Serialization and Deserialization	378
9.7	Additional Notes Regarding Files	378
9.8	Handling Exceptions	379
	9.8.1 Division by Zero and Invalid Input	380
	9.8.2 <code>try</code> Statements	380
	9.8.3 Catching Multiple Exceptions in One <code>except</code> Clause	383
	9.8.4 What Exceptions Does a Function or Method Raise?	384
	9.8.5 What Code Should Be Placed in a <code>try</code> Suite?	384
9.9	<code>finally</code> Clause	384
9.10	Explicitly Raising an Exception	387
9.11	(Optional) Stack Unwinding and Tracebacks	387
9.12	Intro to Data Science: Working with CSV Files	390
	9.12.1 Python Standard Library Module <code>csv</code>	390
	9.12.2 Reading CSV Files into Pandas <code>DataFrames</code>	392
	9.12.3 Reading the Titanic Disaster Dataset	394
	9.12.4 Simple Data Analysis with the Titanic Disaster Dataset	395
	9.12.5 Passenger Age Histogram	396
9.13	Wrap-Up	397

10 Object-Oriented Programming 403

10.1	Introduction	404
10.2	Custom Class Account	406

12 Contents

10.2.1	Test-Driving Class Account	406
10.2.2	Account Class Definition	408
10.2.3	Composition: Object References as Members of Classes	409
10.3	Controlling Access to Attributes	411
10.4	Properties for Data Access	412
10.4.1	Test-Driving Class Time	412
10.4.2	Class Time Definition	414
10.4.3	Class Time Definition Design Notes	418
10.5	Simulating “Private” Attributes	419
10.6	Case Study: Card Shuffling and Dealing Simulation	421
10.6.1	Test-Driving Classes Card and DeckOfCards	421
10.6.2	Class Card—Introducing Class Attributes	423
10.6.3	Class DeckOfCards	425
10.6.4	Displaying Card Images with Matplotlib	426
10.7	Inheritance: Base Classes and Subclasses	430
10.8	Building an Inheritance Hierarchy; Introducing Polymorphism	432
10.8.1	Base Class CommissionEmployee	432
10.8.2	Subclass SalariedCommissionEmployee	435
10.8.3	Processing CommissionEmployees and SalariedCommissionEmployees Polymorphically	439
10.8.4	A Note About Object-Based and Object-Oriented Programming	439
10.9	Duck Typing and Polymorphism	440
10.10	Operator Overloading	441
10.10.1	Test-Driving Class Complex	442
10.10.2	Class Complex Definition	443
10.11	Exception Class Hierarchy and Custom Exceptions	445
10.12	Named Tuples	447
10.13	A Brief Intro to Python 3.7’s New Data Classes	448
10.13.1	Creating a Card Data Class	449
10.13.2	Using the Card Data Class	451
10.13.3	Data Class Advantages over Named Tuples	453
10.13.4	Data Class Advantages over Traditional Classes	454
10.14	Unit Testing with Docstrings and doctest	454
10.15	Namespaces and Scopes	459
10.16	Intro to Data Science: Time Series and Simple Linear Regression	462
10.17	Wrap-Up	471

11 Computer Science Thinking: Recursion, Searching, Sorting and Big O 479

11.1	Introduction	480
11.2	Factorials	481
11.3	Recursive Factorial Example	481
11.4	Recursive Fibonacci Series Example	484
11.5	Recursion vs. Iteration	487
11.6	Searching and Sorting	488

11.7	Linear Search	488
11.8	Efficiency of Algorithms: Big O	490
11.9	Binary Search	492
	11.9.1 Binary Search Implementation	493
	11.9.2 Big O of the Binary Search	495
11.10	Sorting Algorithms	496
11.11	Selection Sort	496
	11.11.1 Selection Sort Implementation	497
	11.11.2 Utility Function <code>print_pass</code>	498
	11.11.3 Big O of the Selection Sort	499
11.12	Insertion Sort	499
	11.12.1 Insertion Sort Implementation	500
	11.12.2 Big O of the Insertion Sort	501
11.13	Merge Sort	502
	11.13.1 Merge Sort Implementation	502
	11.13.2 Big O of the Merge Sort	507
11.14	Big O Summary for This Chapter's Searching and Sorting Algorithms	507
11.15	Visualizing Algorithms	508
	11.15.1 Generator Functions	510
	11.15.2 Implementing the Selection Sort Animation	511
11.16	Wrap-Up	516

12 Natural Language Processing (NLP) 525

12.1	Introduction	526
12.2	TextBlob	527
	12.2.1 Create a TextBlob	529
	12.2.2 Tokenizing Text into Sentences and Words	530
	12.2.3 Parts-of-Speech Tagging	530
	12.2.4 Extracting Noun Phrases	531
	12.2.5 Sentiment Analysis with TextBlob's Default Sentiment Analyzer	532
	12.2.6 Sentiment Analysis with the <code>NaiveBayesAnalyzer</code>	534
	12.2.7 Language Detection and Translation	535
	12.2.8 Inflection: Pluralization and Singularization	537
	12.2.9 Spell Checking and Correction	537
	12.2.10 Normalization: Stemming and Lemmatization	538
	12.2.11 Word Frequencies	539
	12.2.12 Getting Definitions, Synonyms and Antonyms from WordNet	540
	12.2.13 Deleting Stop Words	542
	12.2.14 n-grams	544
12.3	Visualizing Word Frequencies with Bar Charts and Word Clouds	545
	12.3.1 Visualizing Word Frequencies with Pandas	545
	12.3.2 Visualizing Word Frequencies with Word Clouds	548
12.4	Readability Assessment with Textastic	551
12.5	Named Entity Recognition with spaCy	553
12.6	Similarity Detection with spaCy	555

12.7	Other NLP Libraries and Tools	557
12.8	Machine Learning and Deep Learning Natural Language Applications	557
12.9	Natural Language Datasets	558
12.10	Wrap-Up	558

13 Data Mining Twitter **563**

13.1	Introduction	564
13.2	Overview of the Twitter APIs	566
13.3	Creating a Twitter Account	567
13.4	Getting Twitter Credentials—Creating an App	568
13.5	What’s in a Tweet?	569
13.6	Tweepy	573
13.7	Authenticating with Twitter Via Tweepy	573
13.8	Getting Information About a Twitter Account	575
13.9	Introduction to Tweepy Cursors: Getting an Account’s Followers and Friends	577
13.9.1	Determining an Account’s Followers	577
13.9.2	Determining Whom an Account Follows	580
13.9.3	Getting a User’s Recent Tweets	580
13.10	Searching Recent Tweets	582
13.11	Spotting Trends: Twitter Trends API	584
13.11.1	Places with Trending Topics	584
13.11.2	Getting a List of Trending Topics	585
13.11.3	Create a Word Cloud from Trending Topics	587
13.12	Cleaning/Preprocessing Tweets for Analysis	589
13.13	Twitter Streaming API	590
13.13.1	Creating a Subclass of <code>StreamListener</code>	591
13.13.2	Initiating Stream Processing	593
13.14	Tweet Sentiment Analysis	595
13.15	Geocoding and Mapping	599
13.15.1	Getting and Mapping the Tweets	600
13.15.2	Utility Functions in <code>tweetutilities.py</code>	604
13.15.3	Class <code>LocationListener</code>	606
13.16	Ways to Store Tweets	607
13.17	Twitter and Time Series	608
13.18	Wrap-Up	608

14 IBM Watson and Cognitive Computing **613**

14.1	Introduction: IBM Watson and Cognitive Computing	614
14.2	IBM Cloud Account and Cloud Console	616
14.3	Watson Services	616
14.4	Additional Services and Tools	620
14.5	Watson Developer Cloud Python SDK	621
14.6	Case Study: Traveler’s Companion Translation App	622

14.6.1	Before You Run the App	623
14.6.2	Test-Driving the App	624
14.6.3	SimpleLanguageTranslator.py Script Walkthrough	625
14.7	Watson Resources	635
14.8	Wrap-Up	637

15 Machine Learning: Classification, Regression and Clustering 641

15.1	Introduction to Machine Learning	642
15.1.1	Scikit-Learn	643
15.1.2	Types of Machine Learning	644
15.1.3	Datasets Bundled with Scikit-Learn	646
15.1.4	Steps in a Typical Data Science Study	647
15.2	Case Study: Classification with k-Nearest Neighbors and the Digits Dataset, Part 1	647
15.2.1	k-Nearest Neighbors Algorithm	649
15.2.2	Loading the Dataset	650
15.2.3	Visualizing the Data	654
15.2.4	Splitting the Data for Training and Testing	656
15.2.5	Creating the Model	657
15.2.6	Training the Model	658
15.2.7	Predicting Digit Classes	658
15.3	Case Study: Classification with k-Nearest Neighbors and the Digits Dataset, Part 2	660
15.3.1	Metrics for Model Accuracy	660
15.3.2	K-Fold Cross-Validation	664
15.3.3	Running Multiple Models to Find the Best One	665
15.3.4	Hyperparameter Tuning	667
15.4	Case Study: Time Series and Simple Linear Regression	668
15.5	Case Study: Multiple Linear Regression with the California Housing Dataset	673
15.5.1	Loading the Dataset	674
15.5.2	Exploring the Data with Pandas	676
15.5.3	Visualizing the Features	678
15.5.4	Splitting the Data for Training and Testing	682
15.5.5	Training the Model	682
15.5.6	Testing the Model	683
15.5.7	Visualizing the Expected vs. Predicted Prices	684
15.5.8	Regression Model Metrics	685
15.5.9	Choosing the Best Model	686
15.6	Case Study: Unsupervised Machine Learning, Part 1—Dimensionality Reduction	687
15.7	Case Study: Unsupervised Machine Learning, Part 2—k-Means Clustering	690
15.7.1	Loading the Iris Dataset	692

16 Contents

15.7.2	Exploring the Iris Dataset: Descriptive Statistics with Pandas	694
15.7.3	Visualizing the Dataset with a Seaborn pairplot	695
15.7.4	Using a KMeans Estimator	698
15.7.5	Dimensionality Reduction with Principal Component Analysis	700
15.7.6	Choosing the Best Clustering Estimator	703
15.8	Wrap-Up	704

16 Deep Learning 713

16.1	Introduction	714
16.1.1	Deep Learning Applications	716
16.1.2	Deep Learning Demos	717
16.1.3	Keras Resources	717
16.2	Keras Built-In Datasets	717
16.3	Custom Anaconda Environments	718
16.4	Neural Networks	720
16.5	Tensors	722
16.6	Convolutional Neural Networks for Vision; Multi-Classification with the MNIST Dataset	724
16.6.1	Loading the MNIST Dataset	725
16.6.2	Data Exploration	726
16.6.3	Data Preparation	728
16.6.4	Creating the Neural Network	730
16.6.5	Training and Evaluating the Model	739
16.6.6	Saving and Loading a Model	744
16.7	Visualizing Neural Network Training with TensorBoard	745
16.8	ConvnetJS: Browser-Based Deep-Learning Training and Visualization	748
16.9	Recurrent Neural Networks for Sequences; Sentiment Analysis with the IMDb Dataset	749
16.9.1	Loading the IMDb Movie Reviews Dataset	750
16.9.2	Data Exploration	751
16.9.3	Data Preparation	753
16.9.4	Creating the Neural Network	754
16.9.5	Training and Evaluating the Model	757
16.10	Tuning Deep Learning Models	758
16.11	Convnet Models Pretrained on ImageNet	759
16.12	Reinforcement Learning	760
16.12.1	Deep Q-Learning	761
16.12.2	OpenAI Gym	761
16.13	Wrap-Up	762

17 Big Data: Hadoop, Spark, NoSQL and IoT 771

17.1	Introduction	772
17.2	Relational Databases and Structured Query Language (SQL)	776
17.2.1	A books Database	778

17.2.2	SELECT Queries	782
17.2.3	WHERE Clause	782
17.2.4	ORDER BY Clause	784
17.2.5	Merging Data from Multiple Tables: INNER JOIN	785
17.2.6	INSERT INTO Statement	786
17.2.7	UPDATE Statement	787
17.2.8	DELETE FROM Statement	787
17.3	NoSQL and NewSQL Big-Data Databases: A Brief Tour	789
17.3.1	NoSQL Key-Value Databases	789
17.3.2	NoSQL Document Databases	790
17.3.3	NoSQL Columnar Databases	790
17.3.4	NoSQL Graph Databases	791
17.3.5	NewSQL Databases	791
17.4	Case Study: A MongoDB JSON Document Database	792
17.4.1	Creating the MongoDB Atlas Cluster	793
17.4.2	Streaming Tweets into MongoDB	794
17.5	Hadoop	803
17.5.1	Hadoop Overview	803
17.5.2	Summarizing Word Lengths in <i>Romeo and Juliet</i> via MapReduce	806
17.5.3	Creating an Apache Hadoop Cluster in Microsoft Azure HDInsight	806
17.5.4	Hadoop Streaming	808
17.5.5	Implementing the Mapper	808
17.5.6	Implementing the Reducer	809
17.5.7	Preparing to Run the MapReduce Example	810
17.5.8	Running the MapReduce Job	811
17.6	Spark	814
17.6.1	Spark Overview	814
17.6.2	Docker and the Jupyter Docker Stacks	815
17.6.3	Word Count with Spark	818
17.6.4	Spark Word Count on Microsoft Azure	821
17.7	Spark Streaming: Counting Twitter Hashtags Using the pyspark-notebook Docker Stack	825
17.7.1	Streaming Tweets to a Socket	825
17.7.2	Summarizing Tweet Hashtags; Introducing Spark SQL	828
17.8	Internet of Things and Dashboards	834
17.8.1	Publish and Subscribe	836
17.8.2	Visualizing a PubNub Sample Live Stream with a Freeboard Dashboard	836
17.8.3	Simulating an Internet-Connected Thermostat in Python	838
17.8.4	Creating the Dashboard with Freeboard.io	840
17.8.5	Creating a Python PubNub Subscriber	842
17.9	Wrap-Up	846

This page is intentionally left blank



Preface

“There’s gold in them thar hills!”¹

For many decades, some powerful trends have been in place. Computer hardware has rapidly been getting faster, cheaper and smaller. Internet bandwidth (that is, its information carrying capacity) has rapidly been getting larger and cheaper. And quality computer software has become ever more abundant and essentially free or nearly free through the “open source” movement. Soon, the “Internet of Things” will connect tens of billions of devices of every imaginable type. These will generate enormous volumes of data at rapidly increasing speeds and quantities.

Not so many years ago, if people had told us that we’d write a college-level introductory programming textbook with words like “Big Data” and “Cloud” in the title and a graphic of a multicolored whale (emblematic of “big”) on the cover, our reaction might have been, “Huh?” And, if they’d told us we’d include AI (for artificial intelligence) in the title, we might have said, “Really? Isn’t that pretty advanced stuff for novice programmers?”

If people had said, we’d include “Data Science” in the title, we might have responded, “Isn’t data already included in the domain of ‘Computer Science’? Why would we need a separate academic discipline for it?” Well, in programming today, the latest innovations are “all about the data”—*data science*, *data analytics*, *big data*, relational *databases* (SQL), and NoSQL and NewSQL *databases*.

So, here we are! Welcome to *Intro to Python for Computer Science and Data Science: Learning to Program with AI, Big Data and the Cloud*.

In this book, you’ll learn hands-on with today’s most compelling, leading-edge computing technologies—and, as you’ll see, with an easily tunable mix of computer science and data science appropriate for introductory courses in those and related disciplines. And, you’ll program in Python—one of the world’s most popular languages and the fastest growing among them. In this Preface, we present the “soul of the book.”

Professional programmers often quickly discover that they like Python. They appreciate its expressive power, readability, conciseness and interactivity. They like the world of open-source software development that’s generating an ever-growing base of reusable software for an enormous range of application areas.

Whether you’re an instructor, a novice student or an experienced professional programmer, this book has much to offer you. Python is an excellent first programming language for novices and is equally appropriate for developing industrial-strength applications. For the novice, the early chapters establish a solid programming foundation.

We hope you’ll find *Intro to Python for Computer Science and Data Science* educational, entertaining and challenging. It has been a joy to work on this project.

1. Source unknown, frequently misattributed to Mark Twain.

Python for Computer Science and Data Science Education

Many top U.S. universities have switched to Python as their language of choice for teaching introductory computer science, with “eight of the top 10 CS departments (80%), and 27 of the top 39 (69%)” using Python.² It’s now particularly popular for educational and scientific computing,³ and it recently surpassed R as the most popular data science programming language.^{4,5,6}

Modular Architecture

We anticipate that the computer science undergraduate curriculum will evolve to include a data science component—this book is designed to facilitate that and to meet the needs of introductory data science courses with a Python programming component.

The book’s **modular architecture** (please see the **Table of Contents graphic** on the book’s first page) helps us meet the diverse needs of computer science, data science and related audiences. Instructors can adapt it conveniently to a wide range of courses offered to **students drawn from many majors**.

Chapters 1–11 cover traditional introductory computer science programming topics. Chapters 1–10 each include an *optional* brief **Intro to Data Science** section introducing artificial intelligence, basic descriptive statistics, measures of central tendency and dispersion, simulation, static and dynamic visualization, working with CSV files, pandas for data exploration and data wrangling, time series and simple linear regression. These help you prepare for the data science, AI, big data and cloud case studies in Chapters 12–17, which present opportunities for you to use **real-world datasets** in complete case studies.

After covering Python Chapters 1–5 and a few key parts of Chapters 6–7, you’ll be able to handle significant portions of the **data science, AI and big data case studies** in Chapters 12–17, which are appropriate for all contemporary programming courses:

- Computer science courses will likely work through more of Chapters 1–11 and fewer of the Intro to Data Science sections in Chapters 1–10. CS instructors will want to cover some or all of the case-study Chapters 12–17.
- Data science courses will likely work through fewer of Chapters 1–11, most or all of the Intro to Data Science sections in Chapters 1–10, and most or all of the case-study Chapters 12–17.

The “Chapter Dependencies” section of this Preface will help instructors plan their syllabi in the context of the book’s unique architecture.

Chapters 12–17 are loaded with cool, powerful, contemporary content. They present hands-on implementation case studies on topics such as supervised machine learning, unsupervised machine learning, deep learning, reinforcement learning (in the exercises), natural

2. Guo, Philip, “Python Is Now the Most Popular Introductory Teaching Language at Top U.S. Universities,” ACM, July 07, 2014, <https://cacm.acm.org/blogs/blog-cacm/176450-python-is-now-the-most-popular-introductory-teaching-language-at-top-u-s-universities/fulltext>.

3. <https://www.oreilly.com/ideas/5-things-to-watch-in-python-in-2017>.

4. <https://www.kdnuggets.com/2017/08/python-overtakes-r-leader-analytics-data-science.html>.

5. <https://www.r-bloggers.com/data-science-job-report-2017-r-passes-sas-but-python-leaves-them-both-behind/>.

6. <https://www.oreilly.com/ideas/5-things-to-watch-in-python-in-2017>.

language processing, data mining Twitter, cognitive computing with IBM’s Watson, big data and more. Along the way, you’ll acquire a **broad literacy** of data science terms and concepts, ranging from briefly defining terms to using concepts in small, medium and large programs. Browsing the book’s detailed index will give you a sense of the breadth of coverage.

Audiences for the Book

The modular architecture makes this book appropriate for several audiences:

- **All standard Python computer science and related majors.** First and foremost, our book is a solid contemporary Python CS 1 entry. The computing curriculum recommendations from the ACM/IEEE list five types of computing programs: Computer Engineering, Computer Science, Information Systems, Information Technology and Software Engineering.⁷ The book is appropriate for each of these.
- **Undergraduate courses for data science majors**—Our book is useful in many data science courses. It follows the curriculum recommendations for **integration of all the key areas in all courses**, as appropriate for intro courses. In the proposed data science curriculum, the book can be the primary textbook for the first computer science course or the first data science course, then be used as a Python reference throughout the upper curriculum.
- **Service courses** for students who are not computer science or data science majors.
- **Graduate courses in data science**—The book can be used as the primary textbook in the first course, then as a Python reference in other graduate-level data science courses.
- **Two-year colleges**—These schools will increasingly offer courses that prepare students for data science programs in the four-year colleges—the book is an appropriate option for that purpose.
- **High schools**—Just as they began teaching computer classes in response to strong interest, many are already teaching Python programming and data science classes.⁸ According to a recent article on LinkedIn, “data science should be taught in high school,” where the “curriculum should mirror the types of careers that our children will go into, focused directly on where jobs and technology are going.”⁹ We believe that data science could soon become a popular college advanced-placement course and that eventually there will be a data science AP exam.
- **Professional industry training courses.**

Key Features

KIS (Keep It Simple), KIS (Keep it Small), KIT (Keep it Topical)

- **Keep it simple**—In every aspect of the book and its instructor and student supplements, we strive for **simplicity and clarity**. For example, when we present nat-

7. <https://www.acm.org/education/curricula-recommendations>.

8. <http://datascience.la/introduction-to-data-science-for-high-school-students/>.

9. <https://www.linkedin.com/pulse/data-science-should-taught-high-school-rebecca-croucher/>.

ural language processing, we use the simple and intuitive TextBlob library rather than the more complex NLTK. In general, when multiple libraries could be used to perform similar tasks, we use the simplest one.

- **Keep it small**—Most of the book’s 538 examples are small—often just a few lines of code, with immediate interactive IPython feedback. We use large examples as appropriate in approximately 40 larger scripts and complete case studies.
- **Keep it topical**—We read scores of recent Python-programming and data science textbooks and professional books. In all we browsed, read or watched about 15,000 current articles, research papers, white papers, videos, blog posts, forum posts and documentation pieces. This enabled us to “take the pulse” of the Python, computer science, data science, AI, big data and cloud communities to create 1566 up-to-the-minute examples, exercises and projects (EEPs).

IPython’s Immediate-Feedback, Explore, Discover and Experiment Pedagogy

- The ideal way to learn from this book is to read it and run the code examples in parallel. Throughout the book, we use the **IPython interpreter**, which provides a friendly, immediate-feedback, interactive mode for quickly exploring, discovering and experimenting with Python and its extensive libraries.
- Most of the code is presented in **small, interactive IPython sessions** (which we call **IIs**). For each code snippet you write, IPython immediately reads it, evaluates it and prints the results. This **instant feedback** keeps your attention, boosts learning, facilitates rapid prototyping and speeds the software-development process.
- Our books always emphasize the **live-code teaching approach**, focusing on *complete, working programs with sample inputs and outputs*. IPython’s “magic” is that it turns snippets into live code that “comes alive” as you enter each line. This promotes learning and encourages experimentation.
- IPython is a great way to learn the error messages associated with common errors. We’ll intentionally make errors to show you what happens. When we say something is an error, try it to see what happens.
- We use this same immediate-feedback philosophy in the book’s 557 **Self-Check Exercises** (ideal for “flipped classrooms”—we’ll soon say more about that phenomenon) and many of the 471 end-of-chapter exercises and projects.

Python Programming Fundamentals

- First and foremost, this is an introductory Python textbook. We provide rich coverage of Python and general programming fundamentals.
- We discuss Python’s programming models—**procedural programming**, **functional-style programming** and **object-oriented programming**.
- We emphasize **problem-solving** and **algorithm development**.
- We use best practices to **prepare students for industry**.
- **Functional-style programming** is used throughout the book as appropriate. A chart in Chapter 4 lists most of Python’s key functional-style programming capabilities and the chapters in which we initially cover many of them.

538 Examples, and 471 Exercises and Projects (EEPs)

- Students use a hands-on applied approach to learn from a broad selection of **real-world examples, exercises and projects (EEPs)** drawn from computer science, data science and many other fields.
- The **538 examples** range from individual code snippets to complete computer science, data science, artificial intelligence and big data case studies.
- The **471 exercises and projects** naturally extend the chapter examples. Each chapter concludes with a substantial set of exercises covering a wide variety of topics. This helps instructors tailor their courses to the unique requirements of their audiences and to vary course assignments each semester.
- The EEPs give you an engaging, challenging and entertaining introduction to Python programming, including hands-on AI, computer science and data science.
- Students attack exciting and entertaining challenges with **AI, big data and cloud technologies** like **natural language processing, data mining Twitter, machine learning, deep learning, Hadoop, MapReduce, Spark, IBM Watson**, key data science libraries (**NumPy, pandas, SciPy, NLTK, TextBlob, spaCy, BeautifulSoup, Textastic, Tweepy, Scikit-learn, Keras**), key visualization libraries (**Matplotlib, Seaborn, Folium**) and more.
- Our EEPs encourage you to think into the future. We had the following idea as we wrote this Preface—although it’s not in the text, many similar thought-provoking projects are: With **deep learning**, the **Internet of Things** and large numbers of TV cameras trained on sporting events, it will become possible to keep *automatic statistics*, review the details of every play and resolve instant-replay reviews immediately. So, fans won’t have to endure the bad calls and delays common in today’s sporting events. Here’s a thought—we can use these technologies to eliminate referees. Why not? We’re increasingly entrusting our lives to other deep-learning-based technologies like **robotic surgeons** and **self-driving cars**!
- The **project exercises** encourage you to go deeper into what you’ve learned and research technologies we have not covered. Projects are often larger in scope and may require significant Internet research and implementation effort.
- In the **instructor supplements**, we provide solutions to many exercises, including most in the core Python Chapters 1–11. **Solutions are available only to instructors**—see the section “Instructor Supplements on Pearson’s Instructor Resource Center” later in this Preface for details. **We do not provide solutions to the project and research exercises.**
- We encourage you to look at lots of **demos** and free **open-source** code examples (available on sites such as **GitHub**) for inspiration on additional **class projects, term projects, directed-study projects, capstone-course projects** and **thesis research**.

557 Self-Check Exercises and Answers

- Most sections end with an average of three **Self-Check Exercises**.
- **Fill-in-the-blank, true/false and discussion Self Checks** enable you to test your understanding of the concepts you just studied.

- **IPython interactive Self Checks** give you a chance to try out and reinforce the programming techniques you just learned.
- For rapid learning, answers immediately follow all Self-Check Exercises.

Avoid Heavy Math in Favor of English Explanations

- Data science topics can be highly mathematical. This book will be used in first computer science and data science courses where students may not have deep mathematical backgrounds, so we avoid heavy math, leaving it to upper-level courses.
- We capture the conceptual essence of the mathematics and put it to work in our examples, exercises and projects. We do this by using **Python libraries** such as **statistics**, **NumPy**, **SciPy**, **pandas** and many others, which hide the mathematical complexity. So, it's straightforward for students to get many of the benefits of mathematical techniques like **linear regression** without having to know the mathematics behind them. In the **machine-learning** and **deep-learning** examples, we focus on creating objects that do the math for you “behind the scenes.” This is one of the keys to **object-based programming**. It's like driving a car safely to your destination without knowing all the math, engineering and science that goes into building engines, transmissions, power steering and anti-skid braking systems.

Visualizations

- **67 full-color static, dynamic, animated and interactive two-dimensional and three-dimensional visualizations** (charts, graphs, pictures, animations etc.) help you understand concepts.
- We focus on high-level visualizations produced by **Matplotlib**, **Seaborn**, **pandas** and **Folium** (for **interactive maps**).
- We use visualizations as a pedagogic tool. For example, we make the **law of large numbers** “come alive” in a dynamic **die-rolling simulation** and bar chart. As the number of rolls increases, you'll see each face's percentage of the total rolls gradually approach 16.667% (1/6th) and the sizes of the bars representing the percentages equalize.
- You need to get to know your data. One way is simply to look at the raw data. For even modest amounts of data, you could rapidly get lost in the detail. Visualizations are especially crucial in big data for **data exploration** and **communicating reproducible research results**, where the data items can number in the millions, billions or more. A common saying is that a picture is worth a thousand words¹⁰—in **big data**, a visualization could be worth billions or more items in a database.
- Sometimes, you need to “fly 40,000 feet above the data” to see it “in the large.” **Descriptive statistics** help but can be misleading. Anscombe's quartet, which you'll investigate in the exercises, demonstrates through visualizations that significantly *different* datasets can have *nearly identical* descriptive statistics.
- We show the visualization and animation code so you can implement your own. We also provide the animations in source-code files and as Jupyter Notebooks, so

10. https://en.wikipedia.org/wiki/A_picture_is_worth_a_thousand_words.

you can conveniently customize the code and animation parameters, re-execute the animations and see the effects of the changes.

- Many exercises ask you to create your own visualizations.

Data Experiences

- The undergraduate data science curriculum proposal says “**Data experiences** need to play a central role in all courses.”¹¹
- In the book’s examples, exercises and projects (EEPs), you’ll work with many **real-world datasets and data sources**. There’s a wide variety of **free open datasets** available online for you to experiment with. Some of the sites we reference list hundreds or thousands of datasets. We encourage you to explore these.
- We collected hundreds of syllabi, tracked down **instructor dataset preferences** and researched the most popular datasets for **supervised machine learning, unsupervised machine learning and deep learning** studies. Many of the libraries you’ll use come bundled with popular datasets for experimentation.
- You’ll learn the steps required to obtain data and prepare it for analysis, analyze that data using many techniques, tune your models and communicate your results effectively, especially through visualization.

Thinking Like a Developer

- You’ll work with a **developer focus**, using such popular sites as **GitHub** and **StackOverflow**, and doing lots of Internet research. Our **Intro to Data Science sections** and case studies in Chapters 12–17 provide rich data experiences.
- **GitHub** is an excellent venue for **finding open-source code** to incorporate into your projects (and to contribute your code to the **open-source community**). It’s also a crucial element of the software developer’s arsenal with **version control tools** that help teams of developers manage open-source (and private) projects.
- We encourage you to study developers’ code on sites like GitHub.
- To get ready for career work in computer science and data science, you’ll use an extraordinary range of free and open-source Python and data science **libraries**, free and open **real-world datasets** from government, industry and academia, and **free, free-trial and freemium** offerings of software and cloud services.

Hands-On Cloud Computing

- Much of big data analytics occurs in the cloud, where it’s easy to scale *dynamically* the amount of hardware and software your applications need. You’ll work with various cloud-based services (some directly and some indirectly), including Twitter, Google Translate, IBM Watson, Microsoft Azure, OpenMapQuest, geopy, Dweet.io and PubNub. You’ll explore more in the exercises and projects.
- We encourage you to use free, free trial or freemium services from various cloud vendors. We prefer those that don’t require a credit card because you don’t want

11. “Curriculum Guidelines for Undergraduate Programs in Data Science,” <http://www.annualreviews.org/doi/full/10.1146/annurev-statistics-060116-053930> (p. 18).

to risk accidentally running up big bills. If you decide to use a service that requires a credit card, ensure that the tier you're using for free will not automatically jump to a paid tier.

Database, Big Data and Big Data Infrastructure

- According to IBM (Nov. 2016), 90% of the world's data was created in the last two years.¹² Evidence indicates that the speed of data creation is accelerating.
- According to a March 2016 *AnalyticsWeek* article, within five years there will be over 50 billion devices connected to the Internet and by 2020 we'll be producing 1.7 megabytes of new data every second *for every person on the planet*.¹³
- We include an optional treatment of **relational databases** and SQL with SQLite.
- Databases are critical big data infrastructure for storing and manipulating the massive amounts of data you'll process. Relational databases process *structured data*—they're not geared to the *unstructured* and *semi-structured data* in big data applications. So, as big data evolved, NoSQL and NewSQL databases were created to handle such data efficiently. We include a NoSQL and NewSQL overview and a hands-on case study with a MongoDB JSON document database.
- We include a solid treatment of **big data hardware and software infrastructure** in Chapter 17, “Big Data: Hadoop, Spark, NoSQL and IoT (Internet of Things).”

Artificial Intelligence Case Studies

- Why doesn't this book have an artificial intelligence chapter? After all, AI is on the cover. In the case study Chapters 12–16, we present **artificial intelligence** topics (a key intersection between computer science and data science), including **natural language processing**, **data mining Twitter to perform sentiment analysis**, **cognitive computing with IBM Watson**, **supervised machine learning**, **unsupervised machine learning**, **deep learning** and **reinforcement learning** (in the exercises). Chapter 17 presents the big data hardware and software infrastructure that enables computer scientists and data scientists to implement leading-edge AI-based solutions.

Computer Science

- The Python fundamentals treatment in Chapters 1–10 will get you thinking like a computer scientist. Chapter 11, “Computer Science Thinking: Recursion, Searching, Sorting and Big O,” gives you a more advanced perspective—these are classic computer science topics. Chapter 11 emphasizes performance issues.

Built-In Collections: Lists, Tuples, Sets, Dictionaries

- There's little reason today for most application developers to build *custom* data structures. This is a subject for CS2 courses—our scope is *strictly* CS1 and the corresponding data science course(s). The book features a solid **two-chapter**

12. <https://public.dhe.ibm.com/common/ssi/ecm/wr/en/wr112345usen/watson-customer-engagement-watson-marketing-wr-other-papers-and-reports-wr112345usen-20170719.pdf>.
 13. <https://analyticsweek.com/content/big-data-facts/>.

treatment of Python’s built-in data structures—lists, tuples, dictionaries and sets—with which most data-structuring tasks can be accomplished.

Array-Oriented Programming with NumPy Arrays and Pandas Series/DataFrames

- We take an innovative approach in this book by focusing on three key data structures from open-source libraries—NumPy arrays, pandas Series and pandas DataFrames. These libraries are used extensively in data science, computer science, artificial intelligence and big data. NumPy offers as much as two orders of magnitude higher performance than built-in Python lists.
- We include in Chapter 7 a rich treatment of NumPy arrays. Many libraries, such as pandas, are built on NumPy. The **Intro to Data Science sections** in Chapters 7–9 introduce pandas Series and DataFrames, which along with NumPy arrays are then used throughout the remaining chapters.

File Processing and Serialization

- Chapter 9 presents **text-file processing**, then demonstrates how to serialize objects using the popular **JSON (JavaScript Object Notation)** format. JSON is a commonly used data-interchange format that you’ll frequently see used in the data science chapters—often with libraries that hide the JSON details for simplicity.
- Many data science libraries provide built-in file-processing capabilities for loading datasets into your Python programs. In addition to plain text files, we process files in the popular **CSV (comma-separated values) format** using the Python Standard Library’s `csv` module and capabilities of the pandas data science library.

Object-Based Programming

- In all the Python code we studied during our research for this book, we rarely encountered *custom classes*. These are common in the powerful libraries *used* by Python programmers.
- We emphasize using the enormous number of valuable classes that the **Python open-source community** has packaged into industry standard class libraries. You’ll focus on knowing what libraries are out there, choosing the ones you’ll need for your app, creating objects from existing classes (usually in one or two lines of code) and making them “jump, dance and sing.” This is called **object-based programming**—it enables you to **build impressive applications concisely**, which is a significant part of Python’s appeal.
- With this approach, you’ll be able to use machine learning, deep learning, reinforcement learning (in the exercises) and other AI technologies to solve a wide range of intriguing problems, including **cognitive computing** challenges like **speech recognition** and **computer vision**. In the past, with just an introductory programming course, you never would have been able to tackle such tasks.

Object-Oriented Programming

- For computer science students, developing *custom* classes is a crucial **object-oriented programming** skill, along with inheritance, polymorphism and duck typing. We discuss these in Chapter 10.

- The object-oriented programming treatment is modular, so instructors can present basic or intermediate coverage.
- Chapter 10 includes a discussion of unit testing with `doctest` and a fun card-shuffling-and-dealing simulation.
- The six data science, AI, big data and cloud chapters require only a few straightforward custom class definitions. Instructors who do not wish to cover Chapter 10 can have students simply mimic our class definitions.

Privacy

- In the exercises, you'll research ever-stricter privacy laws such as **HIPAA (Health Insurance Portability and Accountability Act)** in the United States and **GDPR (General Data Protection Regulation)** for the European Union. A key aspect of privacy is protecting users' **personally identifiable information (PII)**, and a key challenge with big data is that it's easy to cross-reference facts about individuals among databases. We mention privacy issues in several places throughout the book.

Security

- Security is crucial to privacy. We deal with some Python-specific security issues.
- AI and big data present unique privacy, security and ethical challenges. In the exercises, students will research the **OWASP Python Security Project** (<http://www.pythonsecurity.org/>), **anomaly detection**, **blockchain** (the technology behind cryptocurrencies like BitCoin and Ethereum) and more.

Ethics

- Ethics conundrum: Suppose big data analytics with AI predicts that a person with no criminal record has a significant chance of committing a serious crime. Should that person be arrested? In the exercises, you'll research this and other ethical issues, including *deep fakes* (AI-generated images and videos that appear to be real), *bias* in machine learning and *CRISPR gene editing*. Students also investigate privacy and ethical issues surrounding AIs and **intelligent assistants**, such as **IBM Watson**, **Amazon Alexa**, **Apple Siri**, **Google Assistant** and **Microsoft Cortana**. For example, just recently, a judge ordered Amazon to turn over Alexa recordings for use in a criminal case.¹⁴

Reproducibility

- In the sciences in general, and data science in particular, there's a need to reproduce the results of experiments and studies, and to communicate those results effectively. **Jupyter Notebooks** are a preferred means for doing this.
- We provide you with a Jupyter Notebooks experience to help meet the reproducibility recommendations of the data science undergraduate curriculum proposal.
- We discuss *reproducibility* throughout the book in the context of programming techniques and software such as Jupyter Notebooks and **Docker**.

14. <https://techcrunch.com/2018/11/14/amazon-echo-recordings-judge-murder-case/>.

Transparency

- The data science curriculum proposal mentions data transparency. One aspect of data transparency is the availability of data. Many governments and other organizations now adhere to **open-data** principles, enabling anyone to access their data.¹⁵ We point you to a wide range of datasets that are made available by such entities.
- Other aspects of data transparency include determining that data is correct and knowing its origin (think, for example, of “fake news”). Many of the datasets we use are bundled with key libraries we present, such as **Scikit-learn** for machine learning and **Keras** for deep learning. We also point you to various curated **dataset repositories** such as the **University of California Irvine (UCI) Machine Learning Repository** (with 450+ datasets)¹⁶ and **Carnegie Mellon University’s StatLib Datasets Archive** (with 100+ datasets).¹⁷

Performance

- We use the **timeit profiling tool** in several examples and exercises to compare the performance of different approaches to performing the same tasks. Other performance-related discussions include generator expressions, NumPy arrays vs. Python lists, performance of machine-learning and deep-learning models, and Hadoop and Spark distributed-computing performance.

Big Data and Parallelism

- Computer applications have generally been good at doing one thing at a time. Today’s more sophisticated applications need to be able to do many things in parallel. The human brain is believed to have the equivalent of 100 billion parallel processors.¹⁸ For years we’ve written about parallelism at the program level, which is complex and error-prone.
- In this book, rather than writing your own parallelization code, you’ll let libraries like Keras running over TensorFlow, and big data tools like Hadoop and Spark parallelize operations for you. In this big data/AI era, the sheer processing requirements of massive data apps demand taking advantage of true parallelism provided by **multicore processors**, **graphics processing units (GPUs)**, **tensor processing units (TPUs)** and huge **clusters of computers in the cloud**. Some big data tasks could have thousands of processors working in parallel to analyze massive amounts of data in reasonable time. Sequentializing such processing is typically not an option, because it would take too long.

15. https://www.mckinsey.com/~media/McKinsey/Business%20Functions/McKinsey%20Digital/Our%20Insights/Big%20data%20The%20next%20frontier%20for%20innovation/MGI_big_data_full_report.ashx (page 56).

16. <https://archive.ics.uci.edu/ml/datasets.html>.

17. <http://lib.stat.cmu.edu/datasets/>.

18. <https://www.technologyreview.com/s/532291/fmri-data-reveals-the-number-of-parallel-processes-running-in-the-brain/>.

Chapter Dependencies

If you're an instructor planning your course syllabus or a professional deciding which chapters to read, this section will help you make the best decisions. Please read the one-page **Table of Contents** on the first page of the book—this will quickly familiarize you with the book's unique architecture. Teaching or reading the chapters in order is easiest. However, much of the content in the Intro to Data Science sections at the ends of Chapters 1–10 and the case studies in Chapters 12–17 requires only Chapters 1–5 and small portions of Chapters 6–10 as discussed below.

Part 1: Python Fundamentals Quickstart

We recommend that all courses cover Python Chapters 1–5:

- **Chapter 1, Introduction to Computers and Python**, introduces concepts that lay the groundwork for the Python programming in Chapters 2–11 and the big data, artificial-intelligence and cloud-based case studies in Chapters 12–17. The chapter also includes **test-drives** of **IPython** and **Jupyter Notebooks**.
- **Chapter 2, Introduction to Python Programming**, presents Python programming fundamentals with code examples illustrating key language features.
- **Chapter 3, Control Statements and Program Development**, presents Python's control statements, focuses on **problem-solving and algorithm development**, and introduces **basic list processing**.
- **Chapter 4, Functions**, introduces program construction using existing functions and custom functions as building blocks, presents **simulation techniques** with **random-number generation** and introduces **tuple fundamentals**.
- **Chapter 5, Sequences: Lists and Tuples**, presents Python's built-in list and tuple collections in more detail and begins our introduction to **functional-style programming**.

Part 2: Python Data Structures, Strings and Files¹⁹

The following summarizes inter-chapter dependencies for Python Chapters 6–9 and assumes that you've read Chapters 1–5.

- **Chapter 6, Dictionaries and Sets**—The Intro to Data Science section is not dependent on Chapter 6's contents.
- **Chapter 7, Array-Oriented Programming with NumPy**—The Intro to Data Science section requires dictionaries (Chapter 6) and arrays (Chapter 7).
- **Chapter 8, Strings: A Deeper Look**—The Intro to Data Science section requires raw strings and regular expressions (Sections 8.11–8.12), and pandas `Series` and `DataFrame` features from Section 7.14's Intro to Data Science.
- **Chapter 9, Files and Exceptions**—For **JSON serialization**, it's useful to understand dictionary fundamentals (Section 6.2). Also, the Intro to Data Science section requires the built-in `open` function and the `with` statement (Section 9.3), and pandas `DataFrame` features from Section 7.14's Intro to Data Science.

19. We could have included Chapter 5 in Part 2. We placed it in Part 1 because that's the group of chapters all courses should cover.