Intro to Python®

for Computer Science and Data Science

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 1
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.

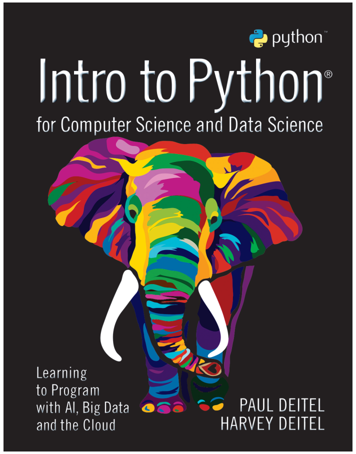
PART 3
CS: Python High-End Topics

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`

Deitel® Series Page

How To Program Series

Android™ How to Program, 3/E
C++ How to Program, 10/E
C How to Program, 8/E
Java™ How to Program, Early Objects Version, 11/E
Java™ How to Program, Late Objects Version, 11/E
Internet & World Wide Web How to Program, 5/E
Visual Basic® 2012 How to Program, 6/E
Visual C#® How to Program, 6/E

REVEL™ Interactive Multimedia

REVEL™ for Deitel Java™

VitalSource Web Books

`http://bit.ly/DeitelOnVitalSource`

Android™ How to Program, 2/E and 3/E
C++ How to Program, 9/E and 10/E
Java™ How to Program, 10/E and 11/E
Simply C++: An App-Driven Tutorial Approach
Simply Visual Basic® 2010: An App-Driven Approach, 4/E
Visual Basic® 2012 How to Program, 6/E
Visual C#® How to Program, 6/E
Visual C#® 2012 How to Program, 5/E

Deitel® Developer Series

Android™ 6 for Programmers: An App-Driven Approach, 3/E
C for Programmers with an Introduction to C11
C++11 for Programmers
C# 6 for Programmers
Java™ for Programmers, 4/E
JavaScript for Programmers
Swift™ for Programmers

LiveLessons Video Training

`http://deitel.com/books/LiveLessons/`

Android™ 6 App Development Fundamentals, 3/E
C++ Fundamentals
Java SE 8™ Fundamentals, 2/E
Java SE 9™ Fundamentals, 3/E
C# 6 Fundamentals
C# 2012 Fundamentals
JavaScript Fundamentals
Swift™ Fundamentals

To receive updates on Deitel publications, Resource Centers, training courses, partner offers and more, please join the Deitel communities on

- Facebook®—`http://facebook.com/DeitelFan`
- Twitter®—`@deitel`
- LinkedIn®—`http://linkedin.com/company/deitel-&-associates`
- YouTube™—`http://youtube.com/DeitelTV`
- Instagram®—`http://instagram.com/DeitelFan`

and register for the free *Deitel® Buzz Online* e-mail newsletter at:

`http://www.deitel.com/newsletter/subscribe.html`

To communicate with the authors, send e-mail to:

`deitel@deitel.com`

For information on programming-languages corporate training seminars offered by Deitel & Associates, Inc. worldwide, write to `deitel@deitel.com` or visit:

`http://www.deitel.com/training/`

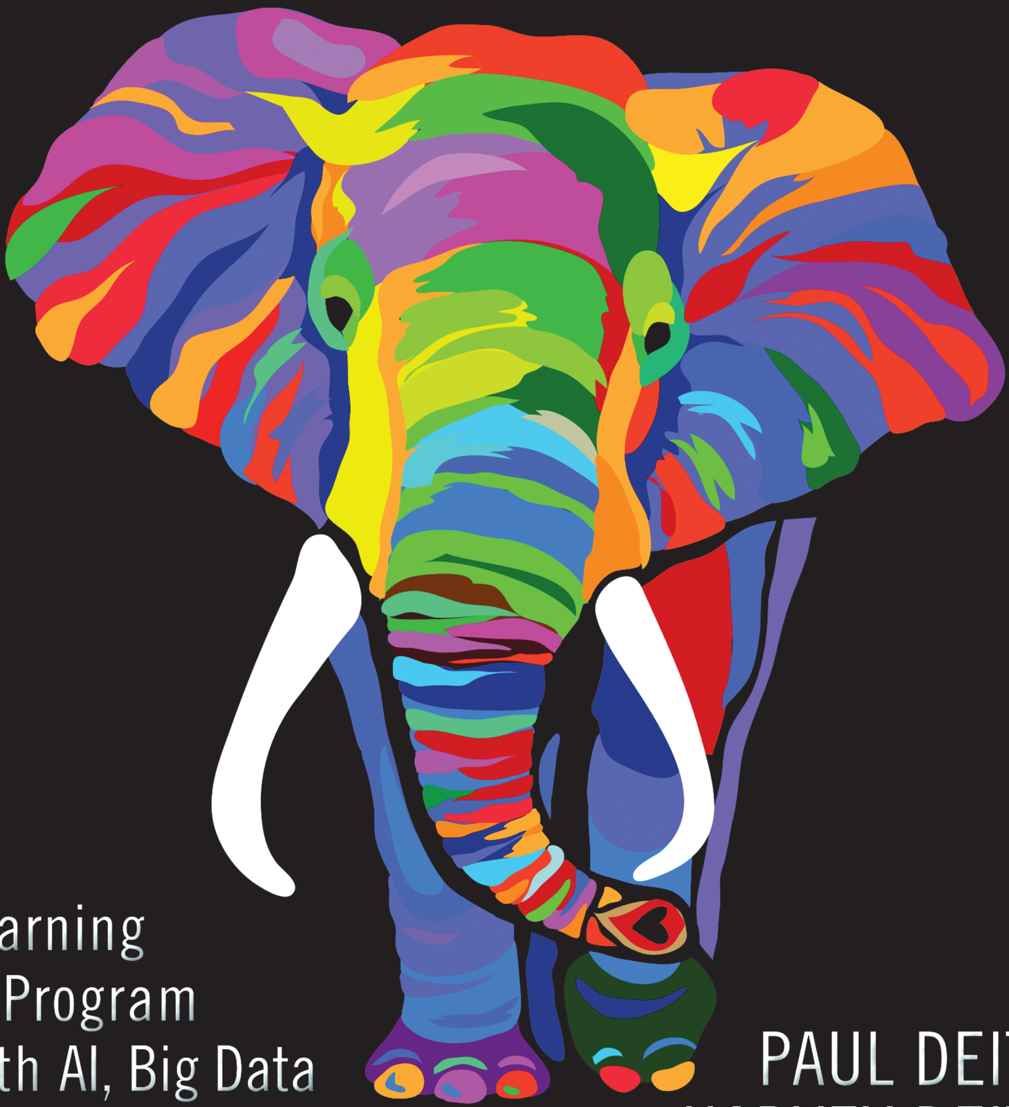
For continuing updates on Pearson/Deitel publications visit:

`http://www.deitel.com`
`http://www.pearson.com/deitel`

python™

Intro to Python®

for Computer Science and Data Science



Learning
to Program
with AI, Big Data
and the Cloud

PAUL DEITEL
HARVEY DEITEL

Senior Vice President, Courseware Portfolio Management: ***Marcia J. Horton***
Director, Portfolio Management: Engineering, Computer Science & Global Editions: ***Julian Partridge***
Executive Higher Ed Portfolio Management: ***Tracy Johnson (Dunkelberger)***
Portfolio Management Assistant: ***Meghan Jacoby***
Managing Content Producer: ***Scott Disanno***
Content Producer: ***Carole Snyder***
Rights and Permissions Manager: ***Ben Ferrini***
Inventory Manager: ***Bruce Boundy***
Product Marketing Manager: ***Yvonne Vannatta***
Field Marketing Manager: ***Demetrius Hall***
Marketing Assistant: ***Jon Bryant***
Cover Designer: ***Paul Deitel, Harvey Deitel, Chuti Prasertsith***
Cover Art: ©***Denel/Shutterstock***

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

The authors and publisher of this book have used their best efforts in preparing this book. These efforts include the development, research, and testing of the theories and programs to determine their effectiveness. The authors and publisher make no warranty of any kind, expressed or implied, with regard to these programs or to the documentation contained in this book. The authors and publisher shall not be liable in any event for incidental or consequential damages in connection with, or arising out of, the furnishing, performance, or use of these programs.

Deitel and the double-thumbs-up bug are registered trademarks of Deitel and Associates, Inc.

Library of Congress Cataloging-in-Publication Data
On file

ISBN-10: 0-13-540467-3
ISBN-13: 978-0-13-540467-6

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

15 Machine Learning: Classification, Regression and Clustering 593

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 elephant (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 *data*bases (SQL), and NoSQL and NewSQL *data*bases.

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**, **Textatistic**, **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 words[10]—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/wrl12345usen/watson-customer-engagement-watson-marketing-wr-other-papers-and-reports-wrl12345usen-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/`.