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 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.

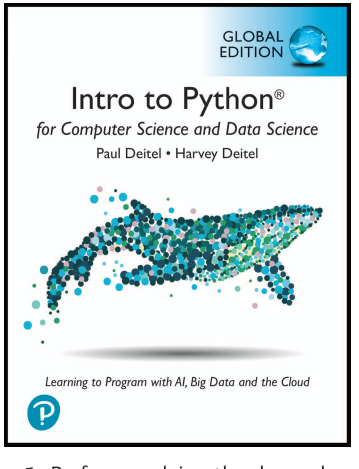
PART 3
CS: Python High-End Topics

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

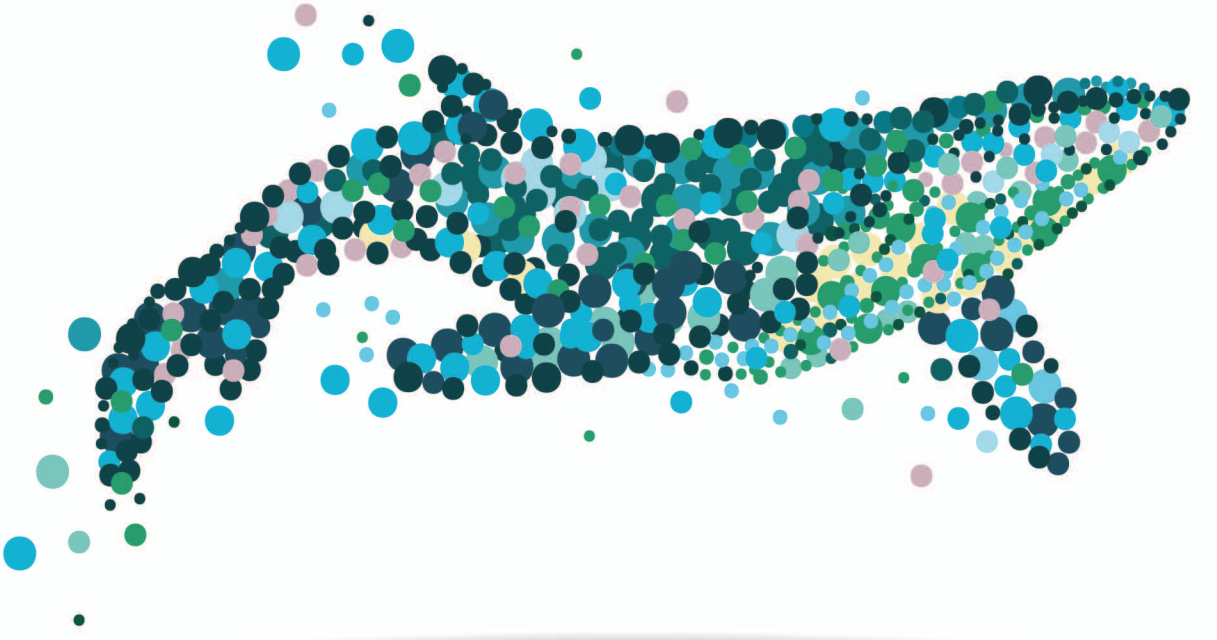
7. **CS courses** may cover more of the Python chapters and less of the DS content. Vice versa for **Data Science courses**.
8. We put Chapter 5 in Part 1. It's also a natural fit with Part 2.

Questions? `deitel@deitel.com`

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.

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.*

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

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 *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/`.

Transparency

- The data science curriculum proposal mentions data transparency. One aspect of data transparency is the availability of data. Many governments and other organization 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 Files[19]

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.

Part 3: Python High-End Topics

The following summarizes inter-chapter dependencies for Python Chapters 10–11 and assumes that you've read Chapters 1–5.

- **Chapter 10, Object-Oriented Programming**—The Intro to Data Science requires pandas `DataFrame` features from the Intro to Data Science Section 7.14. Instructors wanting to cover only **classes and objects** can present Sections 10.1–10.6. Instructors wanting to cover more advanced topics like **inheritance, polymorphism and duck typing**, can present Sections 10.7–10.9. Sections 10.10–10.15 provide additional advanced perspectives.
- **Chapter 11, Computer Science Thinking: Recursion, Searching, Sorting and Big O**—Requires creating and accessing the elements of arrays (Chapter 7), the `%timeit` magic (Section 7.6), string method `join` (Section 8.9) and Matplotlib `FuncAnimation` from Section 6.4's Intro to Data Science.

Part 4: AI, Cloud and Big Data Case Studies

The following summary of inter-chapter dependencies for Chapters 12–17 assumes that you've read Chapters 1–5. Most of Chapters 12–17 also require dictionary fundamentals from Section 6.2.

- **Chapter 12, Natural Language Processing (NLP)**, uses pandas `DataFrame` features from Section 7.14's Intro to Data Science.
- **Chapter 13, Data Mining Twitter**, uses pandas `DataFrame` features from Section 7.14's Intro to Data Science, string method `join` (Section 8.9), JSON fundamentals (Section 9.5), `TextBlob` (Section 12.2) and Word clouds (Section 12.3). Several examples require defining a class via inheritance (Chapter 10), but readers can simply mimic the class definitions we provide without reading Chapter 10.
- **Chapter 14, IBM Watson and Cognitive Computing**, uses built-in function `open` and the `with` statement (Section 9.3).
- **Chapter 15, Machine Learning: Classification, Regression and Clustering**, uses NumPy array fundamentals and method `unique` (Chapter 7), pandas `DataFrame` features from Section 7.14's Intro to Data Science and Matplotlib function `subplots` (Section 10.6).
- **Chapter 16, Deep Learning**, requires NumPy array fundamentals (Chapter 7), string method `join` (Section 8.9), general machine-learning concepts from Chapter 15 and features from Chapter 15's Case Study: Classification with k-Nearest Neighbors and the Digits Dataset.
- **Chapter 17, Big Data: Hadoop, Spark, NoSQL and IoT**, uses string method `split` (Section 6.2.7), Matplotlib `FuncAnimation` from Section 6.4's Intro to Data Science, pandas `Series` and `DataFrame` features from Section 7.14's Intro to Data Science, string method `join` (Section 8.9), the `json` module (Section 9.5), NLTK stop words (Section 12.2.13) and from Chapter 13 Twitter authentication, Tweepy's `StreamListener` class for streaming tweets, and the `geopy` and `folium` libraries. A few examples require defining a class via inheritance (Chapter 10), but readers can simply mimic the class definitions we provide without reading Chapter 10.

Computing and Data Science Curricula

We read the following ACM/IEEE CS-and-related curriculum documents in preparation for writing this book:

- Computer Science Curricula 2013,[20]
- CC2020: A Vision on Computing Curricula,[21]
- Information Technology Curricula 2017,[22]
- Cybersecurity Curricula 2017,[23]

and the 2016 data science initiative "**Curriculum Guidelines for Undergraduate Programs in Data Science**"[24] from the faculty group sponsored by the NSF and the Institute for Advanced Study.

Computing Curricula

- According to "CC2020: A Vision on Computing Curricula," the curriculum "needs to be reviewed and updated to include the new and emerging areas of computing such as **cybersecurity** and **data science**."[25]
- Data science includes key topics (besides general-purpose programming) such as machine learning, deep learning, natural language processing, speech synthesis and recognition and others that are classic artificial intelligence (AI)—and hence CS topics as well.

Data Science Curriculum

- Graduate-level data science is well established and the undergraduate level is growing rapidly to meet strong industry demand. Our hands-on, nonmathematical, project-oriented, programming-intensive approach facilitates moving data science into the undergraduate curriculum, based on the proposed new curriculum.
- There already are lots of undergraduate data science and data analytics programs, but they're not uniform. That was part of the motivation for the 25 faculty members on the data science curriculum committee to get together in 2016 and develop the proposed 10-course undergraduate major in data science, "Curriculum Guidelines for Undergraduate Programs in Data Science."
- The curriculum committee says that "many of the courses traditionally found in computer science, statistics, and mathematics offerings should be redesigned for

---

20. ACM/IEEE (Assoc. Comput. Mach./Inst. Electr. Electron. Eng.). 2013. *Computer Science Curricula 2013: Curriculum Guidelines for Undergraduate Degree Programs in Computer Science* (New York: ACM), http://ai.stanford.edu/users/sahami/CS2013/final-draft/CS2013-final-report.pdf.
21. A. Clear, A. Parrish, G. van der Veer and M. Zhang "CC2020: A Vision on Computing Curricula," https://dl.acm.org/citation.cfm?id= 3017690.
22. *Information Technology Curricula 2017*, http://www.acm.org/binaries/content/assets/education/it2017.pdf.
23. *Cybersecurity Curricula 2017,* https://cybered.hosting.acm.org/wp-content/uploads/2018/02/newcover_csec2017.pdf.
24. "Curriculum Guidelines for Undergraduate Programs in Data Science," http://www.annualreviews.org/doi/full/10.1146/annurev-statistics-060116-053930.
25. http://delivery.acm.org/10.1145/3020000/3017690/p647-clear.pdf.

the data science major in the interests of efficiency and the potential synergy that integrated courses would offer."[26]

- The committee recommends integrating these areas with computational and statistical thinking in all courses, and indicates that ***new textbooks will be essential***[27]**—this book is designed with the committee's recommendations in mind**.
- Python has rapidly become one of the world's most popular general-purpose programming languages. For schools that want to **cover only one language** in their data science major, it's reasonable that Python be that language.

Data Science Overlaps with Computer Science[28]

The undergraduate data science curriculum proposal includes algorithm development, programming, computational thinking, data structures, database, mathematics, statistical thinking, machine learning, data science and more—a significant overlap with computer science, especially given that the data science courses include some key AI topics. Even though ours is a Python programming textbook, it touches each of these areas (except for heavy mathematics) from the recommended data science 10-course curriculum, as we efficiently work data science into various examples, exercises, projects and full-implementation case studies.

Key Points from the Data Science Curriculum Proposal

In this section, we call out some key points from the data science undergraduate curriculum proposal[29] or its detailed course descriptions appendix.[30] We worked hard to incorporate these and many other objectives:

- learn **programming fundamentals** commonly presented in **computer science** courses, including working with **data structures**.
- be able to **solve problems** by **creating algorithms**.
- work with **procedural**, **functional** and **object-oriented programming**.
- receive an integrated presentation of **computational and statistical thinking**, including **exploring concepts via simulations**.
- use **development environments** (we use **IPython** and **Jupyter Notebooks**).
- work with **real-world data** in **practical case studies** and **projects in every course**.
- **obtain**, **explore** and **transform** (**wrangle**) **data** for analysis.
- create **static**, **dynamic** and **interactive data visualizations**.

26. "Curriculum Guidelines for Undergraduate Programs in Data Science," `http://www.annualreviews.org/doi/full/10.1146/annurev-statistics-060116-053930` (pp. 16–17).
27. "Curriculum Guidelines for Undergraduate Programs in Data Science," `http://www.annualreviews.org/doi/full/10.1146/annurev-statistics-060116-053930` (pp. 16–17).
28. This section is intended primarily for data science instructors. Given that the emerging 2020 Computing Curricula for computer science and related disciplines is likely to include some key data science topics, this section includes important information for computer science instructors as well.
29. "Curriculum Guidelines for Undergraduate Programs in Data Science," `http://www.annualreviews.org/doi/full/10.1146/annurev-statistics-060116-053930`.
30. "Appendix—Detailed Courses for a Proposed Data Science Major," `http://www.annualreviews.org/doi/suppl/10.1146/annurev-statistics-060116-053930/suppl_file/st04_de_veaux_supmat.pdf`.

- communicate **reproducible results**.
- work with **existing software** and **cloud-based tools**.
- work with **statistical and machine-learning models**.
- work with **high-performance tools** (**Hadoop**, **Spark**, **MapReduce** and **NoSQL**).
- focus on **data's ethics**, **security**, **privacy**, **reproducibility** and **transparency** issues.

Jobs Requiring Data Science Skills

In 2011, McKinsey Global Institute produced their report, "Big data: The next frontier for innovation, competition and productivity." In it, they said, "The United States alone faces a shortage of 140,000 to 190,000 people with deep analytical skills as well as 1.5 million managers and analysts to analyze big data and make decisions based on their findings."[31] This continues to be the case. The August 2018 "LinkedIn Workforce Report" says the United States has a shortage of over 150,000 people with data science skills.[32] A 2017 report from IBM, Burning Glass Technologies and the Business-Higher Education Forum, says that by 2020 in the United States there will be hundreds of thousands of new jobs requiring data science skills.[33]

Jupyter Notebooks

For your convenience, we provide the book's examples in **Python source code** (`.py`) **files** for use with the command-line IPython interpreter *and* as **Jupyter Notebooks** (`.ipynb`) **files** that you can load into your web browser and execute. You can use whichever method of executing code examples you prefer.

Jupyter Notebooks is a free, open-source project that enables authors to combine text, graphics, audio, video, and interactive coding functionality for entering, editing, executing, debugging, and modifying code quickly and conveniently in a web browser. According to the article, "What Is Jupyter?":

> *Jupyter has become a standard for scientific research and data analysis. It packages computation and argument together, letting you build "computational narratives"; … and it simplifies the problem of distributing working software to teammates and associates.*[34]

In our experience, it's a wonderful learning environment and **rapid prototyping tool** for novices and experienced developers alike. For this reason, we use **Jupyter Notebooks** rather than a traditional **integrated development environment** (**IDE**), such as **Eclipse**, **Visual Studio**, **PyCharm** or **Spyder**. Academics and professionals already use Jupyter extensively for sharing research results. Jupyter Notebooks support is provided through the traditional open-source community mechanisms[35] (see "Getting Your Questions Answered" later in this Preface).

---

31. `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 3).
32. `https://economicgraph.linkedin.com/resources/linkedin-workforce-report-august-2018`.
33. `https://www.burning-glass.com/wp-content/uploads/The_Quant_Crunch.pdf` (page 3).
34. `https://www.oreilly.com/ideas/what-is-jupyter`.
35. `https://jupyter.org/community`.

We believe Jupyter Notebooks are a compelling way to teach and learn Python and that most instructors will choose to use Jupyter. The notebooks include:

- examples,
- Self Check exercises,
- all end-of-chapter exercises containing code, such as "**What does this code do?**" and "**What's wrong with this code?" exercises**.
- **Visualizations and animations**, which are a crucial part of the book's pedagogy. We provide the code in Jupyter Notebooks so students can conveniently **reproduce our results**.

See the Before You Begin section that follows this Preface for software installation details and see the test-drives in Section 1.10 for information on running the book's examples.

Collaboration and Sharing Results

Working in teams and **communicating research results** are both emphasized in the proposed undergraduate data science curriculum[36] and are important for students moving into data-analytics positions in industry, government or academia:

- The notebooks you create are **easy to share** among team members simply by copying the files or via **GitHub**.
- Research results, including code and insights, can be shared as static web pages via tools like **nbviewer** (`https://nbviewer.jupyter.org`) and **GitHub**—both automatically render notebooks as web pages.

Reproducibility: A Strong Case for Jupyter Notebooks

In data science, and in the sciences in general, experiments and studies should be **reproducible**. This has been written about in the literature for many years, including

- Donald Knuth's 1992 computer science publication—*Literate Programming*.[37]
- The article "Language-Agnostic Reproducible Data Analysis Using Literate Programming,"[38] which says, "Lir (literate, reproducible computing) is based on the idea of literate programming as proposed by Donald Knuth."

Essentially, reproducibility captures the complete environment used to produce results—hardware, software, communications, algorithms (especially code), data and the **data's provenance** (origin and lineage).

The undergraduate data science curriculum proposal mentions reproducibility as a goal in four places. The article "50 Years of Data Science" says, "teaching students to work reproducibly enables easier and deeper evaluation of their work; having them reproduce parts of analyses by others allows them to learn skills like **Exploratory Data Analysis** that are commonly practiced but not yet systematically taught; and training them to work reproducibly will make their post-graduation work more reliable."[39]

---

36. "Curriculum Guidelines for Undergraduate Programs in Data Science," `http://www.annualreviews.org/doi/full/10.1146/annurev-statistics-060116-053930` (pp. 18–19).
37. Knuth, D., "Literate Programming" (PDF), *The Computer Journal*, British Computer Society, 1992.
38. `http://journals.plos.org/plosone/article?id=10.1371/journal.pone.0164023`.

Docker

In Chapter 17, we'll introduce **Docker**—a tool for packaging software into *containers* that bundle *everything* required to execute that software conveniently, reproducibly and portably across platforms. Some software packages we use in Chapter 17 require complicated setup and configuration. For many of these, you can download free preexisting **Docker containers**. These enable you to avoid complex installation issues and execute software locally on your desktop or notebook computers, making Docker a great way to help you get started with new technologies quickly and conveniently.

Docker also helps with **reproducibility**. You can create custom Docker containers that are configured with the versions of every piece of software and every library you used in your study. This would enable others to recreate the environment you used, then reproduce your work, and will help you reproduce your own results. In Chapter 17, you'll use Docker to download and execute a container that's preconfigured for you to code and run big data Spark applications using Jupyter Notebooks.

Class Tested

While the book was under development, one of our academic reviewers—Dr. Alison Sanchez, Assistant Professor in Economics, University of San Diego—class tested it in a new course, "Business Analytics Strategy." She commented: "Wonderful for first-time Python learners from all educational backgrounds and majors. My business analytics students had little to no coding experience when they began the course. In addition to loving the material, it was easy for them to follow along with the example exercises and by the end of the course were able to mine and analyze Twitter data using techniques learned from the book. The chapters are clearly written with detailed explanations of the example code—which makes it easy for students without a computer science background to understand. The modular structure, wide range of contemporary data science topics, and companion Jupyter notebooks make this a fantastic resource for instructors and students of a variety of Data Science, Business Analytics and Computer Science courses."

"Flipped Classroom"

Many instructors are now using "flipped classrooms."[40,41] Students learn the content on their own before coming to class (typically via video lectures), and class time is used for tasks such as hands-on coding, working in groups and discussions. Our book and supplements are appropriate for flipped classrooms:

- We provide extensive VideoNotes in which co-author Paul Deitel teaches the concepts in the core Python chapters. See "Student and Instructor Supplements" later in this Preface for details on accessing the videos.
- Some students learn best by —and video is *not* hands-on. **One of the most compelling features of the book** is its interactive approach with **538 Python code**

39. "50 Years of Data Science," `http://courses.csail.mit.edu/18.337/2015/docs/50YearsDataScience.pdf`, p. 33.
40. `https://en.wikipedia.org/wiki/Flipped_classroom`.
41. `https://www.edsurge.com/news/2018-05-24-a-case-for-flipping-learning-without-videos`.

examples—many with just one or a few snippets—and 557 **Self Check exercises with answers**. These enable students to learn in small pieces with immediate feedback—perfect for active self-paced learning. Students can easily modify the "hot" code and see the effects of their changes.

- Our **Jupyter Notebooks supplements** provide a convenient mechanism for students to work with the code.
- We provide 471 exercises and projects, which students can work on at home and/or in class. Many of these are appropriate for group projects.
- We provide lots of probing questions on ethics, privacy, security and more in the exercises and projects. These are appropriate for in-class discussions and group work.

Special Feature: IBM Watson Analytics and Cognitive Computing

Early in our research for this book, we recognized the rapidly growing interest in **IBM's Watson**. We investigated competitive services and found Watson's "no credit card required" policy for its "free tiers" to be among the most friendly for our readers.

IBM Watson is a **cognitive-computing** platform being employed across a wide range of real-world scenarios. Cognitive-computing systems simulate the **pattern-recognition** and **decision-making** capabilities of the human brain to "learn" as they consume more data.[42,43,44] We include a significant hands-on Watson treatment. We use the free **Watson Developer Cloud: Python SDK**, which provides application programming interfaces (APIs) that enable you to interact with Watson's services programmatically. Watson is fun to use and a great platform for letting your creative juices flow. You'll demo or use the following Watson APIs: **Conversation**, **Discovery**, **Language Translator**, **Natural Language Classifier**, **Natural Language Understanding**, **Personality Insights**, **Speech to Text**, **Text to Speech**, **Tone Analyzer** and **Visual Recognition**.

Watson's Lite Tier Services and Watson Case Study

IBM encourages learning and experimentation by providing *free lite tiers* for many of their APIs.[45] In Chapter 14, you'll try demos of many Watson services.[46] Then, you'll use the lite tiers of Watson's **Text to Speech**, **Speech to Text** and **Translate** services to implement a **"traveler's assistant" translation app**. You'll speak a question in English, then the app will transcribe your speech to English text, translate the text to Spanish and speak the Spanish text. Next, you'll speak a Spanish response (in case you don't speak Spanish, we provide an audio file you can use). Then, the app will quickly transcribe the speech to Spanish text, translate the text to English and speak the English response. Cool stuff!

42. http://whatis.techtarget.com/definition/cognitive-computing.
43. https://en.wikipedia.org/wiki/Cognitive_computing.
44. https://www.forbes.com/sites/bernardmarr/2016/03/23/what-everyone-should-know-about-cognitive-computing.
45. Always check the latest terms on IBM's website, as the terms and services may change.
46. https://console.bluemix.net/catalog/.

Teaching Approach

Intro to Python for Computer Science and Data Science contains a rich collection of examples, exercises and projects drawn from many fields. Students solve interesting, real-world problems working with **real-world datasets**. The book concentrates on the principles of good **software engineering** and stresses **program clarity**.

Using Fonts for Emphasis

We place the key terms and the index's page reference for each defining occurrence in **bold** text for easier reference. We place on-screen components in the **bold Helvetica** font (for example, the **File** menu) and use the `Lucida` font for Python code (for example, `x = 5`).

Syntax Coloring

The book is in full color. For readability, we syntax color all the code. Our syntax-coloring conventions are as follows:

```
comments appear in green
keywords appear in dark blue
constants and literal values appear in light blue
errors appear in red
all other code appears in black
```

Objectives and Outline

Each chapter begins with objectives that tell you what to expect and give you an opportunity, after reading the chapter, to determine whether it has met the intended goals. The chapter outline enables students to approach the material in top-down fashion.

538 Examples

The book's **538 examples** contain approximately **4000 lines of code**. This is a relatively small amount of code for a book this size and is due to the fact that Python is such an expressive language. Also, our coding style is to use powerful class libraries to do most of the work wherever possible.

160 Tables/Illustrations/Visualizations

Abundant tables, line drawings, and visualizations are included. The visualizations are in color and include some 2D and 3D, some static and dynamic and some interactive.

Programming Wisdom

We integrate into the discussions programming wisdom from the authors' combined nine decades of programming and teaching experience, including:

- **Good programming practices** and preferred Python idioms that help you produce clearer, more understandable and more maintainable programs.
- **Common programming errors** to reduce the likelihood that you'll make them.
- **Error-prevention tips** with suggestions for exposing bugs and removing them from your programs. Many of these tips describe techniques for preventing bugs from getting into your programs in the first place.
- **Performance tips** that highlight opportunities to make your programs run faster or minimize the amount of memory they occupy.

- **Software engineering observations** that highlight architectural and design issues for proper software construction, especially for larger systems.

Wrap-Up
Chapters 2–17 end with Wrap-Up sections summarizing what you've learned.

Index
We have included an extensive index. The defining occurrences of key terms are highlighted with a **bold** page number.

Software Used in the Book
All the software you'll need for this book is available for Windows, macOS and Linux and is free for download from the Internet. We wrote the book's examples using the free **Anaconda Python distribution**. It includes most of the Python, visualization and data science libraries you'll need, as well as Python, the IPython interpreter, Jupyter Notebooks and Spyder, considered one of the best Python data science integrated development environments (IDEs)—we use only IPython and Jupyter Notebooks for program development in the book. The Before You Begin section discusses installing Anaconda and other items you'll need for working with our examples.

Python Documentation
You'll find the following documentation especially helpful as you work through the book:

- The Python Standard Library:
  `https://docs.python.org/3/library/index.html`
- The Python Language Reference:
  `https://docs.python.org/3/reference/index.html`
- Python documentation list:
  `https://docs.python.org/3/`

Getting Your Questions Answered
Online forums enable you to interact with other Python programmers and get your Python questions answered. Popular Python and general programming forums include:

- `python-forum.io`
- `StackOverflow.com`
- `https://www.dreamincode.net/forums/forum/29-python/`

Also, many vendors provide forums for their tools and libraries. Most of the libraries you'll use in this book are managed and maintained at `github.com`. Some library maintainers provide support through the **Issues** tab on a given library's GitHub page. If you cannot find an answer to your questions online, please see our web page for the book at

`http://www.deitel.com`[47]

47. Our website is undergoing a major upgrade. If you do not find something you need, please write to us directly at `deitel@deitel.com`.

Getting Jupyter Help

Jupyter Notebooks support is provided through:

- Project Jupyter Google Group:
  `https://groups.google.com/forum/#!forum/jupyter`
- Jupyter real-time chat room:
  `https://gitter.im/jupyter/jupyter`
- GitHub
  `https://github.com/jupyter/help`
- StackOverflow:
  `https://stackoverflow.com/questions/tagged/jupyter`
- Jupyter for Education Google Group (for instructors teaching with Jupyter):
  `https://groups.google.com/forum/#!forum/jupyter-education`

Student and Instructor Supplements

The following supplements are available to students and instructors.

Code Examples, Videos Notes and Companion Website

To get the most out of the presentation, you should execute each code example in parallel with reading the corresponding discussion. The book's **Companion Website** at

`https://www.pearsonglobaleditions.com`

contains:

- **Downloadable Python source code** (`.py` files) and **Jupyter Notebooks** (`.ipynb` files) for the book's **code examples**, for code-based **Self-Check Exercises** and for **end-of-chapter exercises** that have code as part of the exercise description.
- **VideoNotes,** in which co-author Paul Deitel explains most of the examples in the book's core Python chapters.

For download instructions, see the *Before You Begin* section that follows this Preface. New copies of this book come with a **Companion Website access code** on the book's inside front cover. If the access code is already visible or there isn't one, you purchased a used book. Access codes are also available on the inside front cover if you are using the eBook or the eText.

Instructor Supplements on Pearson's Instructor Resource Center

The following supplements are available to qualified instructors only through Pearson Education's IRC (Instructor Resource Center) at `https://www.pearsonglobaleditions.com`:

- **PowerPoint slides**.
- **Instructor Solutions Manual** with solutions to many of the exercises. Solutions are *not* provided for "project" and "research" exercises—many of which are substantial and appropriate for term projects, directed-study projects, capstone-course

projects and thesis topics. **Before assigning a particular exercise for homework, instructors should check the IRC to be sure the solution is available.**

- **Test Item File** with multiple-choice, short-answer questions and answers. These are easy to use in automated assessment tools.

Please do not write to us requesting access to the Pearson Instructor's Resource Center which contains the book's instructor supplements, including exercise solutions. Access is strictly limited to college instructors teaching from the book. Instructors may obtain access through their Pearson representatives.

Keeping in Touch with the Authors

For answers to questions, syllabus assistance or to report an error, send an e-mail to us at

```
deitel@deitel.com
```

or interact with us via **social media**:

- **Facebook®** (`http://www.deitel.com/deitelfan`)
- **Twitter®** (`@deitel`)
- **LinkedIn®** (`http://linkedin.com/company/deitel-&-associates`)
- **YouTube®** (`http://youtube.com/DeitelTV`)

Acknowledgments

We'd like to thank Barbara Deitel for long hours devoted to Internet research on this project. We're fortunate to have worked with the dedicated team of publishing professionals at Pearson. We appreciate the guidance, wisdom and energy of Tracy Johnson (Executive Portfolio Manager, Higher Ed Courseware, Computer Science)—she challenged us at every step of the process to "get it right." Carole Snyder managed the book's production and interacted with Pearson's permissions team, promptly clearing our graphics and citations.

We wish to acknowledge the efforts of our academic and professional reviewers. Meghan Jacoby and Patricia Byron-Kimball recruited the reviewers and managed the review process.

Adhering to a tight schedule, the reviewers scrutinized our work, providing countless suggestions for improving the accuracy, completeness and timeliness of the presentation.

Reviewers

Proposal Reviewers

Dr. Irene Bruno, Associate Professor in the Department of Information Sciences and Technology, George Mason University
Lance Bryant, Associate Professor, Department of Mathematics, Shippensburg University
Daniel Chen, Data Scientist, Lander Analytics
Garrett Dancik, Associate Professor of Computer Science/Bioinformatics Department of Computer Science, Eastern Connecticut State University
Dr. Marsha Davis, Department Chair of Mathematical Sciences, Eastern Connecticut State University
Roland DePratti, Adjunct Professor of Computer Science, Eastern Connecticut State University
Shyamal Mitra, Senior Lecturer, Computer Science, University of Texas at Austin
Dr. Mark Pauley, Senior Research Fellow, Bioinformatics, School of Interdisciplinary Informatics, University of Nebraska at Omaha
Sean Raleigh, Associate Professor of Mathematics, Chair of Data Science, Westminster College
Alison Sanchez, Assistant Professor in Economics, University of San Diego
Dr. Harvey Siy, Associate Professor of Computer Science, Information Science and Technology, University of Nebraska at Omaha
Jamie Whitacre, Independent Data Science Consultant

Book Reviewers

Daniel Chen, Data Scientist, Lander Analytics
Garrett Dancik, Associate Professor of Computer Science/Bioinformatics, Eastern Connecticut State University
Pranshu Gupta, Assistant Professor, Computer Science, DeSales University
David Koop, Assistant Professor, Data Science Program Co-Director, U-Mass Dartmouth
Ramon Mata-Toledo, Professor, Computer Science, James Madison University
Shyamal Mitra, Senior Lecturer, Computer Science, University of Texas at Austin
Alison Sanchez, Assistant Professor in Economics, University of San Diego
José Antonio González Seco, IT Consultant
Jamie Whitacre, Independent Data Science Consultant
Elizabeth Wickes, Lecturer, School of Information Sciences, University of Illinois

A Special Thank You

Our thanks to Prof. Alison Sanchez for class-testing the book prepublication in her new "Business Analytics Strategy" class at the University of San Diego. She reviewed the lengthy proposal, adopting the book sight unseen and signed on as a full-book reviewer in parallel with using the book in her class. Her guidance (and courage) throughout the entire book-development process are sincerely appreciated.

Well, there you have it! As you read the book, we'd appreciate your comments, criticisms, corrections and suggestions for improvement. Please send all correspondence to:

```
deitel@deitel.com
```

We'll respond promptly.

Welcome again to the exciting open-source world of Python programming. We hope you enjoy this look at leading-edge computer-applications development with Python, IPython, Jupyter Notebooks, AI, big data and the cloud. We wish you great success!

Paul and Harvey Deitel

Acknowledgments for the Global Edition

Pearson would like to acknowledge and thank the following for their work on the Global Edition.

Contributor
Greet Baldewijns, KU Leuven

Reviewers
Ritesh Ajoodha, The University of the Witwatersrand
David Merand, The University of the Witwatersrand
Faruk Tokdemir, Middle East Technical University

About the Authors

Paul J. Deitel, CEO and Chief Technical Officer of Deitel & Associates, Inc., is an MIT graduate with 38 years of experience in computing. Paul is one of the world's most experienced programming-languages trainers, having taught professional courses to software developers since 1992. He has delivered hundreds of programming courses to industry clients internationally, including Cisco, IBM, Siemens, Sun Microsystems (now Oracle), Dell, Fidelity, NASA at the Kennedy Space Center, the National Severe Storm Laboratory, White Sands Missile Range, Rogue Wave Software, Boeing, Nortel Networks, Puma, iRobot and many more. He and his co-author, Dr. Harvey M. Deitel, are the world's best-selling programming-language textbook/professional book/video authors.

Dr. Harvey M. Deitel, Chairman and Chief Strategy Officer of Deitel & Associates, Inc., has 58 years of experience in computing. Dr. Deitel earned B.S. and M.S. degrees in Electrical Engineering from MIT and a Ph.D. in Mathematics from Boston University—he studied computing in each of these programs before they spun off Computer Science programs. He has extensive college teaching experience, including earning tenure and serving as the Chairman of the Computer Science Department at Boston College before founding Deitel & Associates, Inc., in 1991 with his son, Paul. The Deitels' publications have earned international recognition, with more than 100 translations published in Japanese, German, Russian, Spanish, French, Polish, Italian, Simplified Chinese, Traditional Chinese, Korean, Portuguese, Greek, Urdu and Turkish. Dr. Deitel has delivered hundreds of programming courses to academic, corporate, government and military clients.

About Deitel® & Associates, Inc.

Deitel & Associates, Inc., founded by Paul Deitel and Harvey Deitel, is an internationally recognized authoring and corporate training organization, specializing in computer programming languages, object technology, mobile app development and Internet and web software technology. The company's training clients include some of the world's largest companies, government agencies, branches of the military, and academic institutions. The company offers instructor-led training courses delivered at client sites worldwide on major programming languages and platforms.

Through its 44-year publishing partnership with Pearson/Prentice Hall, Deitel & Associates, Inc., publishes leading-edge programming textbooks and professional books in print and e-book formats, **LiveLessons** video courses, Safari-Live online seminars and **Revel™** interactive multimedia courses. To contact Deitel & Associates, Inc. and the authors, or to request a proposal on-site, instructor-led training, write to:

```
deitel@deitel.com
```

To learn more about Deitel on-site corporate training, visit

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

Before You Begin

This section contains information you should review before using this book.

Font and Naming Conventions

We show Python code and commands and file and folder names in a `sans-serif font`, and on-screen components, such as menu names, in a **bold sans-serif font**. We use *italics for emphasis* and **bold occasionally for strong emphasis**.

Getting the Code Examples

You can download the `examples.zip` file containing the book's examples from Pearson's Companion Website for the book at:

```
https://www.pearsonglobaleditions.com
```

When the download completes, locate it on your system, and extract its `examples` folder into your user account's `Documents` folder:

- Windows: `C:\Users\`*YourAccount*`\Documents\examples`
- macOS or Linux: `~/Documents/examples`

Most operating systems have a built-in extraction tool. You also may use an archive tool such as 7-Zip (`www.7-zip.org`) or WinZip (`www.winzip.com`).

Structure of the `examples` Folder

You'll execute three kinds of examples in this book:

- Individual code snippets in the IPython interactive environment.
- Complete applications, which are known as scripts.
- Jupyter Notebooks—a convenient interactive, web-browser-based environment in which you can write and execute code and intermix the code with text, images and video.

We demonstrate each in Section 1.10's test drives.

The `examples` folder contains one subfolder per chapter. These are named `ch##`, where `##` is the two-digit chapter number `01` to `17`—for example, `ch01`. Except for Chapters 14, 16 and 17, each chapter's folder contains the following items:

- `snippets_ipynb`—A folder containing the chapter's Jupyter Notebook files.
- `snippets_py`—A folder containing Python source code files in which each code snippet we present is separated from the next by a blank line. You can copy and paste these snippets into IPython or into new Jupyter Notebooks that you create.
- Script files and their supporting files.

Chapter 14 contains one application. Chapters 16 and 17 explain where to find the files you need in the `ch16` and `ch17` folders, respectively.

Installing Anaconda

We use the easy-to-install Anaconda Python distribution with this book. It comes with almost everything you'll need to work with our examples, including:

- the IPython interpreter,
- most of the Python and data science libraries we use,
- a local Jupyter Notebooks server so you can load and execute our notebooks, and
- various other software packages, such as the Spyder Integrated Development Environment (IDE)—we use only IPython and Jupyter Notebooks in this book.

Download the Python 3.x Anaconda installer for Windows, macOS or Linux from:

```
https://www.anaconda.com/download/
```

When the download completes, run the installer and follow the on-screen instructions. To ensure that Anaconda runs correctly, do not move its files after you install it.

Updating Anaconda

Next, ensure that Anaconda is up to date. Open a command-line window on your system as follows:

- On macOS, open a **Terminal** from the **Applications** folder's **Utilities** subfolder.
- On Windows, open the **Anaconda Prompt** from the start menu. When doing this to update Anaconda (as you'll do here) or to install new packages (discussed momentarily), execute the **Anaconda Prompt** as an *administrator* by right-clicking, then selecting **More > Run as administrator**. (If you cannot find the Anaconda Prompt in the start menu, simply search for it in the **Type here to search** field at the bottom of your screen.)
- On Linux, open your system's **Terminal** or shell (this varies by Linux distribution).

In your system's command-line window, execute the following commands to update Anaconda's installed packages to their latest versions:

1. `conda update conda`
2. `conda update --all`

Package Managers

The `conda` command used above invokes the **conda package manager**—one of the two key Python package managers you'll use in this book. The other is **`pip`**. Packages contain the files required to install a given Python library or tool. Throughout the book, you'll use `conda` to install additional packages, unless those packages are not available through `conda`, in which case you'll use `pip`. Some people prefer to use `pip` exclusively as it currently supports more packages. If you ever have trouble installing a package with `conda`, try `pip` instead.

Installing the Prospector Static Code Analysis Tool

In the book's exercises, we ask you to analyze Python code using the Prospector analysis tool, which checks your Python code for common errors and helps you improve your code. To install Prospector and the Python libraries it uses, run the following command in the command-line window:

```
pip install prospector
```

Installing jupyter-matplotlib

We implement several animations using a visualization library called Matplotlib. To use them in Jupyter Notebooks, you must install a tool called `ipympl`. In the **Terminal**, **Anaconda Command Prompt** or shell you opened previously, execute the following commands[1] one at a time:

```
conda install -c conda-forge ipympl
conda install nodejs
jupyter labextension install @jupyter-widgets/jupyterlab-manager
jupyter labextension install jupyter-matplotlib
```

Installing the Other Packages

Anaconda comes with approximately 300 popular Python and data science packages for you, such as NumPy, Matplotlib, pandas, Regex, BeautifulSoup, requests, Bokeh, SciPy, SciKit-Learn, Seaborn, Spacy, sqlite, statsmodels and many more. The number of additional packages you'll need to install throughout the book will be small and we'll provide installation instructions as necessary. As you discover new packages, their documentation will explain how to install them.

Get a Twitter Developer Account

If you intend to use our "Data Mining Twitter" chapter and any Twitter-based examples in subsequent chapters, apply for a Twitter developer account. Twitter now requires registration for access to their APIs. To apply, fill out and submit the application at

```
https://developer.twitter.com/en/apply-for-access
```

Twitter reviews every application. At the time of this writing, personal developer accounts were being approved immediately and company-account applications were taking from several days to several weeks. Approval is not guaranteed.

Internet Connection Required in Some Chapters

While using this book, you'll need an Internet connection to install various additional Python libraries. In some chapters, you'll register for accounts with cloud-based services, mostly to use their free tiers. Some services require credit cards to verify your identity. In a few cases, you'll use services that are not free. In these cases, you'll take advantage of monetary credits provided by the vendors so you can try their services without incurring charges. **Caution: Some cloud-based services incur costs once you set them up. When**

1. `https://github.com/matplotlib/jupyter-matplotlib`.

you complete our case studies using such services, be sure to promptly delete the resources you allocated.

Slight Differences in Program Outputs

When you execute our examples, you might notice some differences between the results we show and your own results:

- Due to differences in how calculations are performed with floating-point numbers (like –123.45, 7.5 or 0.0236937) across operating systems, you might see minor variations in outputs—especially in digits far to the right of the decimal point.
- When we show outputs that appear in separate windows, we crop the windows to remove their borders.

Getting Your Questions Answered

Online forums enable you to interact with other Python programmers and get your Python questions answered. Popular Python and general programming forums include:

- `python-forum.io`
- `StackOverflow.com`
- `https://www.dreamincode.net/forums/forum/29-python/`

Also, many vendors provide forums for their tools and libraries. Most of the libraries you'll use in this book are managed and maintained at `github.com`. Some library maintainers provide support through the **Issues** tab on a given library's GitHub page. If you cannot find an answer to your questions online, please see our web page for the book at

`http://www.deitel.com`[1]

You're now ready to begin reading *Intro to Python for Computer Science and Data Sciences: Learning to Program with AI, Big Data and the Cloud*. We hope you enjoy the book!

1. Our website is undergoing a major upgrade. If you do not find something you need, please write to us directly at `deitel@deitel.com`.