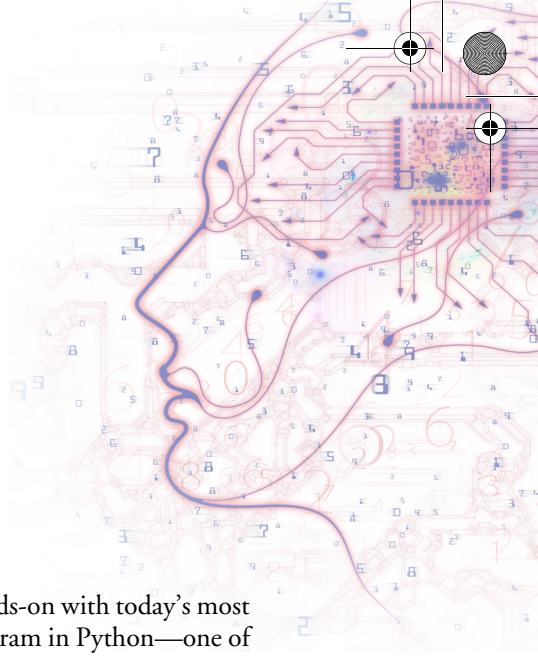

Preface

"*There's gold in them thar hills!*"[1]

Welcome to *Python for Programmers*! In this book, you'll learn hands-on with today's most compelling, leading-edge computing technologies, and you'll program in Python—one of the world's most popular languages and the fastest growing among them.

Developers 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 a rapidly growing base of reusable software for an enormous range of application areas.

For many decades, some powerful trends have been in place. Computer hardware has rapidly been getting faster, cheaper and smaller. Internet bandwidth 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.

In computing 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, each of which we address along with an innovative treatment of Python programming.

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."[2] 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.[3] 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.[4]

---

1. Source unknown, frequently misattributed to Mark Twain.
2. `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).
3. `https://economicgraph.linkedin.com/resources/linkedin-workforce-report-august-2018`.
4. `https://www.burning-glass.com/wp-content/uploads/The_Quant_Crunch.pdf` (page 3).

Modular Architecture

The book's **modular architecture** (please see the **Table of Contents graphic** on the book's inside front cover) helps us meet the diverse needs of various professional audiences.

Chapters 1–10 cover Python programming. These chapters each include a 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 11–16, 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 case studies in Chapters 11–16. The "Chapter Dependencies" section of this Preface will help trainers plan their professional courses in the context of the book's unique architecture.

Chapters 11–16 are loaded with cool, powerful, contemporary examples. They present hands-on implementation case studies on topics such as **natural language processing**, **data mining Twitter**, **cognitive computing with IBM's Watson**, **supervised machine learning with classification and regression**, **unsupervised machine learning with clustering**, **deep learning with convolutional neural networks**, **deep learning with recurrent neural networks**, **big data with Hadoop, Spark and NoSQL databases**, the **Internet of Things** and more. Along the way, you'll acquire a **broad literacy** of data science terms and concepts, ranging from brief definitions to using concepts in small, medium and large programs. Browsing the book's detailed **Table of Contents** and Index will give you a sense of the breadth of coverage.

Key Features

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

- **Keep it simple**—In every aspect of the book, we strive for **simplicity and clarity**. For example, when we present natural language processing, we use the simple and intuitive **TextBlob** library rather than the more complex NLTK. In our deep learning presentation, we prefer **Keras** to TensorFlow. 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 also include 40 larger scripts and in-depth case studies.
- **Keep it topical**—We read scores of recent Python-programming and data science books, and 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.

Immediate-Feedback: Exploring, Discovering and Experimenting with IPython

- 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**. 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 approach**, focusing on complete, working programs with live inputs and outputs. IPython's "magic" is that it turns even snippets into code that "comes alive" as you enter each line. This promotes learning and encourages experimentation.

Python Programming Fundamentals

- First and foremost, this book provides rich Python coverage.
- We discuss Python's programming models—**procedural programming**, **functional-style programming** and **object-oriented programming**.
- We use best practices, emphasizing current idiom.
- **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 most of them.

538 Code Examples

- You'll get an engaging, challenging and entertaining introduction to Python with **538 real-world examples** ranging from individual snippets to substantial computer science, data science, artificial intelligence and big data case studies.
- You'll attack significant tasks 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**, **Textatistic**, **Tweepy**, **Scikit-learn**, **Keras**), key visualization libraries (**Matplotlib**, **Seaborn**, **Folium**) and more.

Avoid Heavy Math in Favor of English Explanations

- We capture the conceptual essence of the mathematics and put it to work in our examples. We do this by using libraries such as **statistics**, **NumPy**, **SciPy**, **pandas** and many others, which hide the mathematical complexity. So, it's straightforward for you 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."

Visualizations

- 67 **static, dynamic, animated and interactive visualizations** (charts, graphs, pictures, animations etc.) help you understand concepts.

- Rather than including a treatment of low-level graphics programming, 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.
- Visualizations are 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[5]—in **big data**, a visualization could be worth billions, trillions or even more items in a database. Visualizations enable you to "fly 40,000 feet above the data" to see it "in the large" and to get to know your data. **Descriptive statistics** help but can be misleading. For example, Anscombe's quartet[6] 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 you can conveniently customize the code and animation parameters, re-execute the animations and see the effects of the changes.

Data Experiences

- Our **Intro to Data Science sections** and case studies in Chapters 11–16 provide rich data experiences.
- You'll work with many **real-world datasets and data sources**. There's an enormous variety of **free open datasets** available online for you to experiment with. Some of the sites we reference list hundreds or thousands of datasets.
- Many 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.

GitHub

- **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.
- You'll use an extraordinary range of free and open-source Python and data science **libraries**, and **free**, **free-trial** and **freemium** offerings of software and cloud services. Many of the libraries are hosted on GitHub.

---

5. `https://en.wikipedia.org/wiki/A_picture_is_worth_a_thousand_words`.
6. `https://en.wikipedia.org/wiki/Anscombe%27s_quartet`.

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**.
- We encourage you to use free, free trial or freemium cloud services. We prefer those that don't require a credit card because you don't want 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.[7] Evidence indicates that the speed of data creation is rapidly 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*![8]
- We include a 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**. MongoDB is the most popular NoSQL database.
- We discuss **big data hardware and software infrastructure** in Chapter 16, "Big Data: Hadoop, Spark, NoSQL and IoT (Internet of Things)."

Artificial Intelligence Case Studies

- In case study Chapters 11–15, we present **artificial intelligence** topics, including **natural language processing**, **data mining Twitter to perform sentiment analysis**, **cognitive computing with IBM Watson**, **supervised machine learning**, **unsupervised machine learning** and **deep learning**. Chapter 16 presents the big data hardware and software infrastructure that enables computer scientists and data scientists to implement leading-edge AI-based solutions.

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

- There's little reason today for most application developers to build *custom* data structures. The book features a rich **two-chapter treatment of Python's built-in data structures—lists**, **tuples**, **dictionaries** and **sets—**with which most data-structuring tasks can be accomplished.

7. `https://public.dhe.ibm.com/common/ssi/ecm/wr/en/wrl12345usen/watson-customer-engagement-watson-marketing-wr-other-papers-and-reports-wrl12345usen-20170719.pdf`.
8. `https://analyticsweek.com/content/big-data-facts/`.

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

- We also focus on three key data structures from open-source libraries—**NumPy arrays**, **pandas `Series`** and **pandas `DataFrames`**. These 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 used frequently in the data science chapters.
- 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

- We emphasize using the huge 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 apps, creating objects from existing classes (usually in one or two lines of code) and making them "jump, dance and sing." This **object-*based* programming** enables you to build impressive applications quickly and concisely, which is a significant part of Python's appeal.
- With this approach, you'll be able to use machine learning, deep learning and other AI technologies to quickly solve a wide range of intriguing problems, including **cognitive computing** challenges like **speech recognition** and **computer vision**.

Object-Oriented Programming

- Developing *custom* classes is a crucial **object-oriented programming** skill, along with inheritance, polymorphism and duck typing. We discuss these in Chapter 10.
- Chapter 10 includes a discussion of **unit testing with `doctest`** and a fun card-shuffling-and-dealing simulation.
- Chapters 11–16 require only a few straightforward custom class definitions. In Python, you'll probably use more of an object-based programming approach than full-out object-oriented programming.

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 discuss reproducibility throughout the book in the context of programming techniques and software such as Jupyter Notebooks and **Docker**.

Performance

- We use the **`%timeit` profiling tool** in several examples 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

- 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 applications 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 expeditiously.

Chapter Dependencies

If you're a trainer planning your syllabus for a professional training course or a developer deciding which chapters to read, this section will help you make the best decisions. Please read the one-page color **Table of Contents** on the book's inside front cover—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 11–16 requires only Chapters 1–5 and small portions of Chapters 6–10 as discussed below.

Part 1: Python Fundamentals Quickstart

We recommend that you read all the chapters in order:

- **Chapter 1, Introduction to Computers and Python**, introduces concepts that lay the groundwork for the Python programming in Chapters 2–10 and the big data, artificial-intelligence and cloud-based case studies in Chapters 11–16. The chapter also includes **test-drives** of the **IPython** interpreter and **Jupyter Notebooks**.
- **Chapter 2, Introduction to Python Programming**, presents Python programming fundamentals with code examples illustrating key language features.
- **Chapter 3, Control Statements**, presents Python's **control statements** and introduces **basic list processing**.
- **Chapter 4, Functions**, introduces custom functions, 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 introducing **functional-style programming**.

Part 2: Python Data Structures, Strings and Files

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 in this chapter is not dependent on the chapter'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.

Part 3: Python High-End Topics

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

- **Chapter 10, Object-Oriented Programming**—The Intro to Data Science section requires pandas `DataFrame` features from Intro to Data Science Section 7.14. Trainers wanting to cover only **classes and objects** can present Sections 10.1–10.6. Trainers 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.

Part 4: AI, Cloud and Big Data Case Studies

The following summary of inter-chapter dependencies for Chapters 11–16 assumes that you've read Chapters 1–5. Most of Chapters 11–16 also require dictionary fundamentals from Section 6.2.

- **Chapter 11, Natural Language Processing (NLP)**, uses pandas `DataFrame` features from Section 7.14's Intro to Data Science.
- **Chapter 12, 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 11.2) and Word clouds (Section 11.3). Several examples require defining a class via inheritance (Chapter 10).
- **Chapter 13, IBM Watson and Cognitive Computing**, uses built-in function `open` and the `with` statement (Section 9.3).
- **Chapter 14, 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 15, Deep Learning**, requires NumPy array fundamentals (Chapter 7), string method `join` (Section 8.9), general machine-learning concepts from

Chapter 14 and features from Chapter 14's Case Study: Classification with k-Nearest Neighbors and the Digits Dataset.

- **Chapter 16, 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 11.2.13) and from Chapter 12, 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 you can simply mimic the class definitions we provide without reading Chapter 10.

Jupyter Notebooks

For your convenience, we provide the book's code 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.

Jupyter Notebooks is a free, open-source project that enables you 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.*[9]

In our experience, it's a wonderful learning environment and **rapid prototyping tool**. For this reason, we use **Jupyter Notebooks** rather than a traditional 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[10] (see "Getting Jupyter Help" later in this Preface). See the Before You Begin section that follows this Preface for software installation details and see the test-drives in Section 1.5 for information on running the book's examples.

Collaboration and Sharing Results

Working in teams and communicating research results are both important for developers in or 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.

9. `https://www.oreilly.com/ideas/what-is-jupyter`.
10. `https://jupyter.org/community`.

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*.[11]
- The article "Language-Agnostic Reproducible Data Analysis Using Literate Programming,"[12] 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).

Docker

In Chapter 16, we'll use **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 16 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 other developers to recreate the environment you used, then reproduce your work, and will help you reproduce your own results. In Chapter 16, 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.

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.[13,14,15] We include a significant hands-on Watson treatment. We use the free **Watson Developer Cloud: Python SDK**, which provides 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**

11. Knuth, D., "Literate Programming" (PDF), *The Computer Journal*, British Computer Society, 1992.
12. `http://journals.plos.org/plosone/article?id=10.1371/journal.pone.0164023`.
13. `http://whatis.techtarget.com/definition/cognitive-computing`.
14. `https://en.wikipedia.org/wiki/Cognitive_computing`.
15. `https://www.forbes.com/sites/bernardmarr/2016/03/23/what-everyone-should-know-about-cognitive-computing`.

Understanding, **Personality Insights**, **Speech to Text**, **Text to Speech**, **Tone Analyzer** and **Visual Recognition**.

Watson's Lite Tier Services and a Cool Watson Case Study

IBM encourages learning and experimentation by providing *free lite tiers* for many of its APIs.[16] In Chapter 13, you'll try demos of many Watson services.[17] 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!

Teaching Approach

Python for Programmers contains a rich collection of examples drawn from many fields. You'll work through interesting, real-world examples using **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 refer to 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

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
```

538 Code Examples

The book's **538 examples** contain approximately **4000 lines of code**. This is a relatively small amount 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

We include abundant tables, line drawings, and static, dynamic and interactive visualizations.

16. Always check the latest terms on IBM's website, as the terms and services may change.
17. `https://console.bluemix.net/catalog/`.

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.

Software Used in the Book

The software we use 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 the IPython interpreter, Jupyter Notebooks and Spyder, considered one of the best Python data science IDEs. We use only IPython and Jupyter Notebooks for program development in the book. The Before You Begin section following this Preface discusses installing Anaconda and a few 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 Language Reference:
  `https://docs.python.org/3/reference/index.html`
- The Python Standard Library:
  `https://docs.python.org/3/library/index.html`
- Python documentation list:
  `https://docs.python.org/3/`

Getting Your Questions Answered

Popular Python and general programming online forums include:

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

Also, many vendors provide forums for their tools and libraries. Many of the libraries you'll use in this book are managed and maintained at `github.com`. Some library main-

tainers 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`[18]

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`

Supplements

To get the most out of the presentation, you should execute each code example in parallel with reading the corresponding discussion in the book. On the book's web page at

`http://www.deitel.com`

we provide:

- **Downloadable Python source code** (`.py` files) and **Jupyter Notebooks** (`.ipynb` files) for the book's **code examples**.
- **Getting Started videos** showing how to use the code examples with IPython and Jupyter Notebooks. We also introduce these tools in Section 1.5.
- **Blog posts** and **book updates**.

For download instructions, see the Before You Begin section that follows this Preface.

Keeping in Touch with the Authors

For answers to questions 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`)

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

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 efforts and 25-year mentorship of our friend and colleague Mark L. Taub, Vice President of the Pearson IT Professional Group. Mark and his team publish our professional books, LiveLessons video products and Learning Paths in the Safari service (`https://learning.oreilly.com/`). They also sponsor our Safari live online training seminars. Julie Nahil managed the book's production. We selected the cover art and Chuti Prasertsith designed the cover.

We wish to acknowledge the efforts of our reviewers. Patricia Byron-Kimball and Meghan Jacoby 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

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

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, 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

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, data science, AI, big data and the cloud. We wish you great success!

Paul and Harvey Deitel

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.

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 (available for purchase at `https://www.informit.com`), **Learning Paths** and live online training seminars in the Safari service (`https://learning.oreilly.com`) 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
```

Individuals wishing to purchase Deitel books can do so at

```
https://www.amazon.com
```

Bulk orders by corporations, the government, the military and academic institutions should be placed directly with Pearson. For more information, visit

```
https://www.informit.com/store/sales.aspx
```