Preface

The sources and forms of data, along with the demand for acquiring, processing, and analyzing that data are increasing at a prodigious rate. From a curricular perspective, majors, programs, and courses are being designed to enable interested students to understand these topics and prepare themselves for the practice of these associated data skills. As the sources and forms of data increase, with a commensurate increase in the data providers and interfaces for accessing data, our interpretation of what constitutes a data system must keep pace. An understanding of these forms and sources of data is exactly what is needed in a broad introduction to data systems.

Data systems encompasses the study of forms and sources of data, an increasingly important topic for both computer scientists and data scientists. This book covers data acquisition, wrangling, normalization, and curation, requiring only basic prior exposure to Python. The book includes a detailed treatment of tidy data, relational data, and hierarchical data, laying a conceptual basis for the structure, operations, and constraints of each data model, while simultaneously providing hands-on skills in Python, SQL, and XPath. The sources of data studied encompass local files, text applications and regular expressions, database servers, HTTP requests, REST APIs, and web scraping.

Who Is This Book for?

As university curricula expand to include content on data systems, the student audience for such curricular efforts is broadening as well. We identify three student constituencies at the undergraduate level:

- *Computer Science*: students recognizing the value of being facile while working with data and seeing the synergy with the systems and algorithmic problem-solving aspects of the computer science discipline.

- *Data Science/Data Analytics*: students pursuing the emerging undergraduate majors, who desire to build algorithmic and practical skills with an eye toward enabling data exploration, visualization, statistical analysis, and application of machine learning techniques for use in coursework as well as in practicums and internships.
- *Multidisciplinary/domain-centric*: students whose focus is on practical aspects of acquiring and transforming data for use in their own disciplines. These include students in social sciences as well as natural sciences, and often as a prelude to a "methods" course specific to their discipline.

Within computer science education, the union of these audiences has been termed the "data-centric" audience. We have carefully identified a set of essential data-related topics, and a desired level of understanding, that these student constituencies will need to be successful in their future endeavors.

This book is ideal for first- or second-year undergraduate students, either for a classroom setting or for self-directed learning, and does not require prerequisites of data structures, algorithms, or other courses beyond a first course in Python programming. This book equips students with understanding and skills that can be applied in computer science, data science/data analytics, and information technology programs as well as for internships and research experiences. By drawing together content normally spread across a set of upper-level computer science courses, it offers a single source providing the essentials for data science practitioners, an accessible and foundational second course for computer science majors, a potential second course for data science majors, and content that can supplement introductory courses seeking more exposure to real-world data. In our increasingly data-centric world, students from all domains both want and need the *data-aptitude* built by the material in this book.

Philosophy of This Book

We see two primary dimensions to data-aptitude, corresponding to the *sources* and *forms* of data. Data sources range from local files, downloaded or given to the student or analyst, to a relational database system, to a wide assortment of network-based data providers. Data forms/formats include comma-separated value or other delimited flat files, tables within a relational database, XML and JSON used for data interchange, as well as unstructured data and HTML from which data can be extracted. Data models give a framework for understanding the structure, operations, and constraints for these various forms of data. A third dimension of data-aptitude relates to the protection and privacy of the data. Some data is explicitly open and freely available. Other data may be limited to use by particular applications. Still other data may be comprised of protected resources of one or more resource owners, and the acquisition and use of such data must be appropriately authorized.

An Introduction to Data Systems, as advanced by this book, takes the perspective that students can, early in their academic careers, learn these dimensions of data-aptitude. Students need, in a structured way, to learn about the forms and sources of data in modern data systems, defined broadly. As *users* of data systems, students need to be able to build applications to acquire data and, given the form of the data acquired, be able to extract and transform the data into a normalized form, where subsequent statistics and analysis may be easily performed on the data.

This book uses the framework of a set of data models, from a simple tabular model with appropriate constraints, to the relational data model, and to a hierarchical data model, giving structure to the various data forms seen in practice. In the data sources dimension of data-aptitude, the course follows a progression of data sources, starting the students with local files, moving on to MySQL database server and SQLite databases as representative of relational databases, and then covering more advanced client–server interaction over HTTP using provider APIs, and also extracting data from HTML. In this way, we feel we can cover the topics of data-aptitude in sufficient depth, while keeping material broad and applicable to the wide range of data-centric students.

Since specific programming languages and packages are bound to change over the course of the reader's lifetime, we stress a conceptual understanding over an exhaustive coverage of packages. For each topic, we begin with a high-level discussion, situating the topic within its disciplinary context. We then discuss *how* to work with the topic, with an emphasis on problem solving and algorithms. Lastly, we illustrate using tools such as Python, `pandas`, SQL, XPath, `curl`, etc. In this way, we ensure that readers understand the principles underlying these tools, rather than only how to use them as a black box. We select tools that are powerful enough to illustrate the concepts, and simultaneously as easy as possible for readers to learn. Readers are free to focus their energy on the essential content, rather than struggling to learn the tools. We include detailed references that can guide interested readers to further exploration.

To avoid overwhelming readers with too many different data situations, we center our exposition on three data sets—based on economic data, sociological data, and educational data—that we carry with us throughout the book. These illustrate our foundational material, the three data models, and the process of data acquisition. This allows us to center our discussion on compelling real-world problems, rather than on the tools used to solve these problems. This approach has been shown to be effective for a diverse array of student learners, and to increase retention in the discipline. When we develop data wrangling solutions on these data sets, we illustrate good software engineering principles, guiding the reader through the process of developing incrementally, testing their code as they go, and error handling. We include a large number of exercises where students can sharpen their skills.

Web Resources

Accompanying this book is a website supported by the authors, containing files and supplementary content that will aid the reader:

http://datasystems.denison.edu.

This website hosts data files used in Parts I and II of the book and is used to illustrate HTTP, HTML, and web scraping in Part III. In addition, the authors have taught many iterations of a course using this book and have curated a repository of hundreds of exercises, reading questions, and hands-on activities engaging students in the material. These resources are contained within Jupyter notebooks that use the freely available nbgrader system, so that worksheets may be automatically graded. The repository also contains multiple in-depth projects guiding students to work with real-world data sets, normalize the data into the relevant data model, generate interesting questions, and answer these questions with visualizations. Via these projects, students can create a portfolio to showcase what they have learned to potential future employers and graduate programs. We host several sample projects on the book website, and we intend to add more as we continue to teach courses using this book. In this way, projects can be updated if real-world data sources change where they host data, what data they make available, or the form in which the data is provided.

To Students

This text was written with both concrete and abstract goals in mind. A large part of our motivation was to serve "data-focused" students and provide a bridge for you to take the learning from the classroom and use these skills in *concrete, real-world settings*. The focus of the book is giving you the skills you need to acquire data from a multitude of sources, mutate it into a form suitable for analysis, and access it programmatically to answer interesting questions. This includes data stored in files, on database servers, provided by Application Programming Interfaces (APIs), and data obtained via web scraping. The projects (hosted on the book web page) will guide you to applying your new skills in the real world: after learning about data models, you will be able to use that knowledge to work with real-world data sets, normalize the data into the model, and then generate interesting questions and visualizations to help answer those questions. Similarly, after learning about obtaining data over the Internet, REST APIs, and authorization for protected resources, the API projects allow you to bring it all together, and acquire data from providers such as LinkedIn, Reddit, various Google APIs, Lyft, and many others. This will entail using the material from the entire arc of the book.

We surveyed students to see how the material in this book benefited them after the course was over. The vast majority of respondents reported using material from

some or all of the chapters of this book in their subsequent jobs, internships, and research projects. Furthermore, if done correctly, the projects you complete can become part of a *portfolio* that you show to potential employers, which, coupled with your knowledge of the terms and concepts covered in this book, can help you secure the kind of data-focused job you are interested in.

Equally important to these concrete learning goals are our abstract goals: to sharpen your analytic thinking, problem solving, coding, writing, and technical reading skills. For each technique in this book, we first describe the approach in general terms, then carefully work through multiple examples, and finally provide numerous exercises (building on the examples) for you to achieve mastery. In our examples, we model an *incremental approach*, where we develop a partial solution, test it, and then develop a bit more, repeating this process until we have a general solution. We encourage you to follow the same steps when you solve exercises, including the use of try/except blocks and assert statements to ensure that your code behaves as expected.

This book is written to only assume you have prior exposure to computer science principles (and Python) at the level of an introductory course. Normally, the material in this book is spread across several electives that only junior and senior computer science majors take. By condensing the material, and taking an elementary approach to it, we aim to give you the concrete and abstract skills early in your college career. However, the trade-off is that this book contains a great deal that will probably be new to you. We do not expect that you will be an expert at reading computer science books. Some parts may be difficult to understand or may require you to read multiple times. As with anything you read, if you come across a term that is new to you, we encourage you to look that term up and understand it before proceeding. It may be that the term was defined earlier in the book, in which case the index at the end of the book can tell you where the term was first defined.

Most sections begin with an abstract approach and introduce examples later. It may make sense to peek ahead at the examples if you struggle with the abstract part, or to reread it after finishing the chapter to better structure what you have learned. To help you identify the most important parts of each section, and to guide you through the types of activities that will help you to understand the material (e.g., relating the abstract concepts to experiences you may have already had in the real world), we include *reading questions* at the end of each section. These may be assigned by your instructor, to make sure you attempt the reading before class. Even if they are not assigned, we recommend working through the questions as you read each section, as they will often clarify the meaning of new terms introduced, highlight potential pitfalls, and emphasize which pieces of the reading are most essential. You can and should use Python when answering reading questions that reference code, modules, and methods.

Data systems is a rapidly evolving field, and this book is the first of its kind. Previously, students who wanted to learn this material would need to do so by reading online tutorials that often treated data science tools as a black box. By emphasizing the concepts and programming underlying the tools, we aim to give you a deeper level of understanding. This understanding, coupled with the technical

reading skills you will develop, will aid you if, in future years, you find yourself needing to read tutorials and manuals in order to learn new technologies that did not exist when you were a student. For this reason, reading questions sometimes guide you to online self-learning materials to get you comfortable with this kind of exploration. This approach is especially apparent in Part III, where we learn how to acquire data from web scraping and APIs, which, necessarily, differ between different data providers. Our approach teaches you the general steps that will guide you to learn the particulars of APIs and HTML web pages, so that you can develop software to obtain data from a diverse array of sources.

As you read the text, in addition to trying to answer the reading questions, it is essential to familiarize yourself with the blocks of code provided. You should expect to practice with the code every day, as you would if you were learning a new language or musical instrument. Fundamental to the process of learning is to make connections between the new material you are seeing and your past life experience. We encourage you to be curious and playful as you explore the content: try plugging in different values for the parameters of functions, adding print statements to see how a block of code computes, and coming up with questions beyond the examples in the book.

To Instructors

This book is designed with minimal prerequisites, requiring only that students have prior exposure to Python programming at the level of a first course in computer science. This makes it suitable for any of the three student constituencies identified above, or as a supplementary textbook to go along with an introductory computer science course and add a more data-driven focus. As discussed above, we emphasize conceptual understanding, an abstract framework, concrete examples based on real-world data, and good software engineering principles. This emphasis gives readers a deeper understanding than would be obtained from a "pick it up as you go" approach, or an approach rooted in technical manuals. The structure provided in this book empowers readers to learn new technologies throughout their lives, as part of a unifying framework.

Broadly speaking, the book breaks down into six units, which we summarize in Table 1.

There is a rich set of possible orderings for covering the material in the book for a given course, and we have successfully used several different orderings. The book chapters themselves progress by covering all the data models prior to the treatment of the data sources. But, a course that wanted to interleave the two parts could do so, with arguable benefit for student engagement. For instance, after learning the tabular model, the course could cover networking and HTTP to access data sets from web servers and immediately apply the data model in client software. Similarly, after coverage of the hierarchical model, the course could access XML from web servers or non-authenticated APIs to again apply the data model.

Table 1 Textbook contents

Unit	Name	Description
00	Foundations	After an overview of data systems, we begin building a foundation in Python to be used in the remainder of the book. We review introductory basic topics typically found in intro courses. We then progress to useful and possibly new topics like list comprehensions, lambda functions, and regular expressions
01	Tabular data model	The focus is on single, rectangular table data sets. We work from native Python structures to `pandas` data frames. We conclude with the constraints required for rectangular data to be considered "tidy data"
02	Relational data model	This unit explores the data model associated with relational databases and the operations and organization involved in sound design of the set of tables for a database. We develop skills in querying existing databases and creating databases using the Structured Query Language (SQL) through both direct commands and Python programming with `sqlalchemy`
03	Hierarchical data model	The focus shifts to hierarchical data, such as that found in XML and JSON. We explore both declarative XPath and programmatic means to process the data and wrangle it into data frames. We also cover schemas for constraining data in the model, such as XML Schema and JSON Schema
04	Networking, HTTP, HTML	In this unit, our goals are to understand and program client applications over the network. We start with foundational networking concepts and the network protocol stack and progress to learning HTTP and making requests for static data in files or as HyperText Markup Language (HTML)
05	APIs	The book culminates with acquisition of data from API providers. This unit is a synthesis of the full set of data models and provider sources. This unit covers APIs and associated authentication (including OAuth) and illustrates the framework with real-world providers

We often begin with Chaps. 1–3, because this material is foundational and can identify any gaps in student background. Chapter 4 can be substantially delayed if one wishes to get more quickly to data models. While the tabular data model and relational data model can be taught in either order, we generally teach the tabular model first, as it only depends on local files instead of forming connections to database servers. Our study of hierarchical operations depends on Chap. 6, because the examples wrangle hierarchical data into tabular form, but otherwise our treatment of hierarchical models is logically independent from our treatment of the other two data models.

We note that the relational model can be taught as a stand-alone sequence consisting of Chaps. 10–14, which may be valuable to give students the basics of databases and SQL in a three-week module or attached to another course. One could similarly teach Chaps. 2, 6, 15, 16, and 17 as a stand-alone sequence

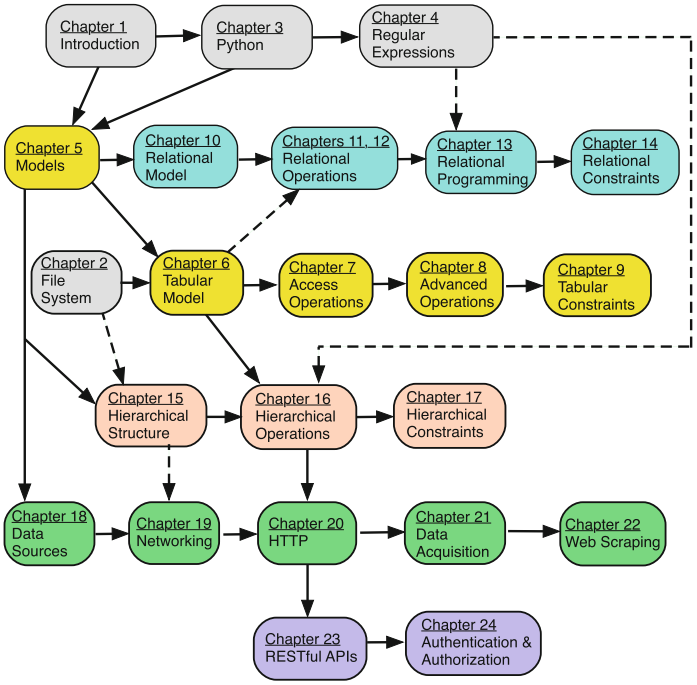

Fig. 1 Chapter dependencies

on hierarchical data or could teach Chaps. 18–20 as a stand-alone sequence on the basics of networking. If a reader is primarily interested in web scraping and APIs, the tabular and relational model could be skipped entirely (except for the introduction of `pandas` in Chap. 6), and the minimal set of chapters to cover in this situation would be Chaps. 1–6, 15–16, and 20–22 (or 20, 23, and 24 for APIs).

Figure 1 shows the dependency structure between the chapters. We have color coded the chapters to help visualize the units described above. Dotted arrows represent suggested, but optional, orderings. The dotted lines from regular expressions to hierarchical operations and to relational programming support some more advanced exercises in those downstream chapters.

For many topics, we employ a spiral approach: returning to the topic in increasingly complex ways as the book progresses. For example, regular expressions are introduced in Chap. 4, then reinforced in Chaps. 13 and 16, and then can be used heavily in Chap. 22. Similarly, the JSON format is introduced in Chap. 2, then reinforced in Chap. 13, and then used heavily in Chaps. 15, 16, and 22–23. We also introduce non-local (over the network) data sources gradually. Concepts of protocol, connection, and user identity (authentication and authorization) are introduced concurrently with the relational data model by using databases provided by a database server. The next source of data, web servers, allows us to expand the basics of networking, the network protocol stack, and then the fundamentals of the

HyperText Transfer Protocol (HTTP). Understanding the aspects of this protocol allows the facility for constructing and issuing requests and being able to retrieve and understand the constituent parts of the response. This type of data source can provide delimited flat files, HTML files, or even XML/JSON files. We combine this data source with the knowledge from the hierarchical data model to understand web scraping as an application of previously introduced ideas.

Software Assumptions

To run the code in this book, you will need to have Python 3.4 (or later) on your machine, as well as a number of packages and libraries, including `pandas`, `os`, `sys`, `io`, `json`, `re`, `sqlalchemy`, `sqlite3`, `lxml`, `jsonschema`, `requests`, `base64` and a custom module discussed in the Appendix and hosted on the book website. At the time of writing, this version of Python, along with all of these packages and libraries, was included with a standard installation of the freely available Anaconda distribution from Continuum Analytics.

https://store.continuum.io/cshop/anaconda

Online Corrigenda

This book was written in `bookdown`, using facilities of Python and SQL code as part of the textbook composition. In this facility, every example piece of Python and SQL is executed as part of the rendering process, ensuring that there are no errors in the code presented to the reader. Furthermore, for examples that reference Internet data sources (e.g., in the web scraping and API chapters), we attempted to select sources that seemed most likely to remain stable in the years to come. Nevertheless, we are aware of the possibility that the reader may discover typos, or that code referencing online data sources may cease to work properly. As errors are identified, they will be corrected at

https://www.datasystems.denison.edu/errata.

If you find an error not documented at the link above, please report it using the link below:

https://www.datasystems.denison.edu/feedback.

Acknowledgments

The authors are grateful to the Denison University for hosting the web resources associated with this book and providing a wonderful working environment where it was written. We also thank Dick De Veaux for facilitating the Park City Mathematics

Institute Undergraduate Faculty Program in 2016 where this book began, and Deborah Nolan and Duncan Temple-Lang for sharing an early draft of their book *Data Technologies and Computational Reasoning*. We are indebted too, to the students at Denison. We especially thank Gavin Thomas and Paul Rubenstein for their pedagogical research work related to this book (which produced a paper published in SIGCSE in 2019). Students from the Denison CS-181 and DA-210 courses have given feedback and helped tremendously. Emma Steinman was particularly helpful and dedicated in giving feedback, and Caileigh Marshall in helping with materials on SQL and OAuth2. In the Spring of 2020, a number of additional students gave significant thought and feedback, and so we sincerely thank Paul Bass, Thomas Luong, Ben Rahal, Jill Reiner, Dan Seely, Jay Dickson, Matthew Bartlett, Brandon Novak, and Dang Pham.

Granville, OH, USA — Tom Bressoud
Granville, OH, USA — David White
June 2020