Preface

Live in fragments no longer. Only connect.
Edward Morgan Forster

Welcome to the world of Python programming! Python is a powerful general-purpose programming language that is extremely effective for developing Internet and Web-based, database intensive, multi-tier, client/server systems. This book presents a great variety of leading edge computing technologies, and is our second book on open-source programming languages.[1]

As we write these words, Python 2.2 has just been released, almost to the minute! We have worked hard to incorporate the 2.2 functionality throughout the book. Appendix O presents a few additional 2.2 features.

We hope you will find *Python How to Program* educational, entertaining and challenging. It was a joy to work on this project. The team at Deitel & Associates develops programming language textbooks and e-Learning materials. We work in almost every major programming language. We noticed something special while working on this book. Our developers and writers commented on how much they like Python. They appreciate its power, its readability and its conciseness. They like its pizzazz. They like the world of open-source software development that is generating an ever-growing base of modules.

Whether you are an instructor, a student, an experienced professional programmer or a novice, this book has much to offer you. Python is an excellent first programming language and is an equally excellent language for developing industrial-strength, commercial applications. For the student and the novice programmer, the early chapters of the book establish a solid foundation in the basics of programming. We discuss many programming models including structured programming, object-based programming, object-oriented programming and event-driven programming. For the professional developer, we have employed the "heavy-duty" Python functionality to create substantial, fully implemented

---

1. *Perl How to Program* was published in January, 2001.

systems. The capstone is the case study on building an online bookstore in Chapter 23—this occupies approximately 70 pages of the text.

The standard basic topics are all here—data types, operators, control structures, arithmetic, strings, decision making, algorithm development, functions, and random numbers and simulation.

 The book features a solid treatment of data structures with an early introduction to Python's built-in structures—lists, tuples and dictionaries—and a later rigorous treatment of traditional data structures including queues, stacks, linked lists and binary trees.

The book emphasizes Internet and Web development—we feature an early introduction to CGI then use it in several chapters to build Web-based applications. We include a full-chapter treatment of PSP (Python Server Pages) where we re-engineer the message forum case study presented in Chapter 16.

The book features a detailed, three-chapter treatment of object-oriented programming covering classes, encapsulation, objects, attributes, methods, constructors, destructors, customization, operator overloading, inheritance, base classes, derived classes and polymorphism.

We include a thorough treatment of graphical user interface (GUI) programming, with **Tkinter**, using event-driven programming, labels, buttons, check buttons, radio buttons, mouse-event handling, keyboard-event handling, layout managers, and a whole range of advanced GUI capabilities for creating and manipulating menus and scrolling components.

We discuss exception handling for making programs more robust. We present a substantial treatment of Python's powerful string-manipulation capabilities and we tackle head-on the difficult—yet enormously powerful—topic of regular expressions.

We discuss file processing, sequential access files, random-access files (and the **shelve** module). We develop a transaction-processing program and consider object serialization. The discussions of file processing provide a nice lead-in to our treatment of database programming with Python's Database Application Programming Interface (DB-API). We discuss the relational database model and present an introduction to SQL (Structured Query Language).

Many people are familiar with HTML; yet few know that the World Wide Web Consortium—creator of HTML technology—has declared HTML to be a legacy technology that will undergo no further development. Nevertheless, HTML is still important, so we have provided two appendices on it. The world is evolving towards XML (eXtensible Markup Language). In the interim, contemporary Web development is using a transition technology called XHTML. We present two appendices on this important subject and use XHTML in various applications throughout the book. We present a chapter-length general introduction to XML, an absolute must subject for today's Web applications developers. We then include a full-chapter treatment of Python-specific XML processing and include a detailed case study in which we use CGI and XML to build message forums.

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, or as we prefer to say in the computer field—concurrently. We provide full chapters on process management and multithreading. These technologies give the Python applications programmer capabilities that used to be available only to systems programmers working down at the operating systems level.

We discuss networking, including the HTTP protocol of the Web, client/server networking with stream sockets, connectionless client/server interaction using datagrams, and we implement a client/server tic-tac-toe game using a multithreaded server.

We present a thorough general discussion of computer security, then deal with some Python-specific security issues. We discuss executing potentially harmful code in a restricted environment using module **Bastion**. We demonstrate encrypting text with module **rotor**.

As a capstone exercise for the book, in Chapter 23 we present a detailed case study that implements online bookstore e-business using a great many technologies discussed in the earlier chapters and the appendices. We introduce HTTP sessions and session tracking and build our bookstore as a multi-tier, client/server system able to handle a variety of clients, including a standard Web browser (using XHTML) and wireless clients (using WML and XHTML Basic).

We present a full chapter on multimedia, including an introduction to **PyOpenGL** with 3-D graphics examples and an introduction to Alice, a 3-D environment that provides objects which can be animated with Python scripts. We demonstrate **pygame** by designing a CD player, a movie player and a space-cruiser game.

Recognizing the importance of server-side development, we introduce PSP (Python Server Pages) as an alternative to CGI, and we convert the message forum case study from CGI technology to PSP.

The book is loaded with valuable appendices, including the Operator Precedence Chart, ASCII Character Set, Number Systems (binary, octal, decimal, hexadecimal), Python Development Environments, HTML, XHTML, CSS (Cascading Style Sheets), Career Opportunities (with lots of Web resources) and Unicode. We include an appendix on Accessibility (which overviews issues affecting, and resources for, people with disabilities). We close with an appendix on Additional Python 2.2 Features, including discussions on iterators, generators and nested scopes.

As you read this book, if you have an questions, just send an e-mail to **deitel@deitel.com**. We will respond promptly. Please visit with us from time to time at **www.deitel.com** and be sure to sign up there for *The DEITEL™ BUZZ* e-mail newsletter. We use the Web site and the newsletter to keep our readers up to the minute on Python and our products and services.

Features in Python How to Program

This text contains many additional features, including:

- *"Code Washing."* This is our term for the process we use to format the programs in the book so that they have a carefully commented, open layout. *Python How to Program* code appears in two colors with easy-to-read, syntax highlighting, including Python keywords, strings and comments. Program code is grouped into small, well-documented pieces. This greatly improves code readability—an especially important goal for us, considering that this book contains 14,930 lines of code.

- *Object-Oriented Programming.* Object-oriented programming is the most widely employed technique for developing robust, reusable software. Python was designed to be an object-oriented language, and this text offers a rigorous discussion

of Python's various object-oriented features. Data integrity is of particular concern in Python. All Python class data is public by default, but several techniques exist for ensuring data integrity. We discuss these and other object-oriented topics over three detailed chapters. Chapter 7, Object-Based Programming, introduces how to create classes and discusses public, "private" and *get/set* methods. Chapter 8, Customizing Classes, discusses how to create classes with customized behavior, such as operator overloading, string representation, list and dictionary behavior and methods for customizing attribute access. These concepts are extended in Chapter 9, Object-Oriented Programming: Inheritance, in which we discuss how programmers can create new classes that "absorb" the capabilities of existing classes. This chapter familiarizes the reader with the crucial concepts of polymorphism, abstract classes and concrete classes, which facilitate powerful manipulations of objects belonging to an inheritance hierarchy. The chapter concludes with a discussion of additional object-oriented capabilities available in Python 2.2, including properties.

- *Database Application Programming Interface*. Databases store vast amounts of information that individuals and organizations must access to conduct business. Database management systems (DBMS) are used by organizations and individuals to manipulate databases—Python offers the database application programming interface (DB-API) to access database management systems. Chapter 17, Database Applications Programming Interface (DB-API), details these capabilities and the Structured Query Language (SQL) to query MySQL databases.

- *XML.* Use of Extensible Markup Language (XML) is exploding in the software-development industry, and in the e-business and e-commerce communities. Because XML is a platform-independent technology for describing data and for creating markup languages, XML's data portability integrates well with Python's portable applications and services. Chapter 15, Extensible Markup Language (XML), introduces XML. In this chapter, we discuss basic XML markup and technologies such as DTDs and Schema, which are used to validate XML document contents. In Chapter 16, Python XML Processing, we explain how to manipulate XML documents using the Document Object Model (DOM™) and how to transform XML documents into other types of documents via Extensible Stylesheet Language Transformations (XSLT). This chapter also presents an alternative to DOM, called Simple API for XML (SAX), which acts as an event-based API for XML.

- *Common Gateway Interface (CGI) and Python Server Pages (PSP).* The Internet and World Wide Web are pervasive in today's world, and interactive Web sites are crucial for businesses. Chapter 6, Introduction to the Common Gateway Interface (CGI), and Chapter 25, Python Server Pages (PSP), provide server-side Web technologies with which developers create interactive Web-based applications. A detailed case study in Chapter 23, Case Study: Online Bookstore, employs MySQL, XML, XHTML, XHTML Basic, Cascading Stylesheets, XSLT, CGI and the Wireless Markup Language (WML) to construct a dynamic e-commerce application. The book demonstrates two implementations of an XML message fo-

rum in which users post messages to an online message board. The version in Chapter 16 uses CGI, and the version in Chapter 25 uses PSP.

- *Graphical User Interface (GUI)*. Python does not have built-in graphical user interface capabilities, but many modules are available that provide access to existing GUI software. Chapter 10, Graphical User Interface Components: Part 1, and Chapter 11, Graphical User Interface Components: Part 2, discuss the **Tkinter** module (included in the Python standard library) that provides a Python programmer access to the Tool Command Language/Tool Kit (Tcl/Tk), a popular GUI toolkit. Using these programming tools, a developer can create graphical programs quickly and easily. These chapters provide the reader with the ability to program the GUI portions of the programs in the rest of the text. Chapter 11 also discusses module **Pmw**, which uses **Tkinter** to provide more complex GUI components.

- *Multimedia.* Multimedia capabilities produce powerful applications using sound and color, thereby enhancing users' experiences. Several Python modules are available for creating impressive multimedia applications. Chapter 24, Multimedia, explores the capabilities of **PyOpenGL** and Alice for creating and animating 3D graphics. The chapter also discusses **pygame**, which contains modules that enable a developer to access powerful multimedia libraries. Chapter 24 uses **pygame** to create a compact-disc player, a computer game and a video-player application.

- *Multithreading and Process Management.* Computers can perform many tasks concurrently (i.e., in parallel), such as printing documents, downloading files from a network and surfing the Web. Multithreading is a technology through which programmers can develop applications that perform concurrent tasks. Python's multithreading and process management capabilities are appropriate for today's sophisticated multimedia-intensive, database-intensive, network-based, multiprocessor-based, distributed applications. Chapter 18, Process Management, discusses concurrency and interprocess communication, and Chapter 19, Multithreading, provides a detailed discussion of multithreading, including an explanation of Python's Global Interpreter Lock, which controls and manages thread execution. The chapter also introduces common thread synchronization mechanisms in the context of several examples.

- *File Processing and Serialization*. Most industrial applications read and write data to a disk. Python provides several high-level capabilities for data storage and retrieval. In Chapter 14, File Processing and Serialization, we discuss basic file objects for the storage of sequential data; **shelve** objects for the storage of random-access data and module **cPickle** for serializing entire objects to a disk.

- *XHTML.* The *World Wide Web Consortium (W3C)* has declared HTML to be a legacy technology that will undergo no further development. HTML is being replaced by the Extensible Hypertext Markup Language (XHTML)—an XML-based technology that is rapidly becoming the standard for describing Web content. We use XHTML in Chapter 6, Introduction to the Common Gateway Interface (CGI); Chapter 16, Python XML Processing; Chapter 17, Database Application Programming Interface (DB-API); Chapter 23, Case Study: Online

Bookstore and Chapter 25, Python Server Pages (PSP). We introduce the technology in Appendix I, Introduction to XHTML: Part 1, and Appendix J, Introduction to XHTML: Part 2. These appendices overview headers, images, lists, image maps and other features of this emerging markup language. (We also present a treatment of HTML in Appendices G and H, because the message forums in Chapter 16 and Chapter 25 use scripts that generate HTML).

- *Career Opportunities.* Appendix E, Career Resources, introduces career services available on the Internet. We explore online career services from both the employer's and employee's perspectives. We list many Web sites at which you can submit applications, search for jobs and review applicants (if you are interested in hiring someone). We also review services that build recruiting pages directly into e-businesses. One of our reviewers told us that he had used the Internet as a primary tool in a recent job search, and that this appendix would have helped him expand his search dramatically.

- *Unicode.* As computer systems evolved worldwide, computer vendors developed numeric representations of character sets and special symbols for the local languages spoken in different countries. In some cases, different representations were developed for the same languages. Such disparate character sets hindered communication among computer systems. Python supports the *Unicode Standard*, which is developed and maintained by a non-profit organization called the *Unicode Consortium*). This standard contains a single character set that specifies unique numeric values for characters and special symbols in most of the world's languages. Appendix F, Unicode, discusses the standard, overviews the Unicode Consortium Web site (**www.unicode.org**) and presents a Python application that displays "Welcome to Unicode!" in many languages.

- *Accessibility.* Using the Web presents many challenges to people with disabilities. Individuals with hearing and visual impairments, in particular, have difficulty accessing multimedia-rich Web sites. In an attempt to improve this situation, the World Wide Web Consortium (W3C) launched the Web Accessibility Initiative (WAI), which provides guidelines for making Web sites accessible to people with disabilities. Appendix L, Accessibility, describes these guidelines and highlights various products and services designed to improve the Web-browsing experiences of individuals with disabilities. For example, the chapter introduces VoiceXML and CallXML—two XML-based technologies for increasing the accessibility of Web-based content for people with visual impairments.

This text provides many other topics in addition to the features previously listed. For a complete list of each chapter's features, visit page 15 of Chapter 1, Tour of the Book.

Python Version 2.2 Features

This text went to publication at the same time that the final version of Python 2.2 was released. However, all the code examples in this book were tested in Python 2.2b2 (Python Version 2.2 Beta 2) and with Release Candidate 1, on both the Windows and Linux operating systems, and the text demonstrates 2.2 features and functionality in every applicable

chapter.[2] This section overviews the 2.2 capabilities presented in *Python How to Program*.

Floor Division and True Division. Python 2.2 introduces a new operator (`//`) for "floor" (integer) division. In the current and previous versions of Python, the default behavior of the `/` division operator is floor division; in future versions, the default behavior will be "true" (floating-point) division. By defining two division operators, the new versions of Python eliminates the type ambiguity that results in programs that use both integer and floating-point values for division. Section 2.6 discusses the difference between floor division and true division and explains how a program can change the default behavior of the `/` operator to perform true division.

Nested Scopes. Python 2.2 introduces nested scopes, which means that nested classes, methods and functions now have access to variables defined in their enclosing scope. This behavior is particularly helpful for writing **lambda** expressions. Chapter 4 discusses Python's basic scoping rules and provides a footnote that contains a Web resource for further information on nested scopes. The nested scoping behavior is most important to programmers who use Python in a functional-programming idiom. We also discuss nested scopes in Appendix O, Additional Python features. As this book focuses mainly on the object-oriented style of programming, we provide only a high-level motivation for nested scopes and suggest further resources where the reader can learn more about nested scopes and functional programming in Python.

More Object-Oriented Functionality. Most of the new features in Python 2.2 add more object-oriented functionality to the language. Chapters 8 and 9 introduce several of these new features. In Chapter 8, we discuss the methods that a programmer-defined class can overload to define behavior for operators, including the new `//` operator. We also introduce a dictionary method, new to 2.2, that enables programs to use **if**/**in** statements to test whether a dictionary contains a particular key. In Chapter 9, we discuss the most anticipated new feature—the ability for programmer-defined classes to inherit from built-in types. We present a substantial example that inherits from built-in type **list** to implement a programmer-defined list that contains only unique elements. We also discuss other new object-oriented features, including static methods, **__slots__** (for defining the attributes an object of a class may contain), method **__getattribute__** (that executes each time a client accesses an object's attribute) and properties (that allow classes to define *get*/*set* methods that execute when a client accesses an attribute).

Iterators. Appendix O contains additional 2.2 features that are not covered in the main text. We begin with a thorough treatment of iterators—special objects for progressing through the values of a sequence. Section O.2 contains two examples that present programmer-defined iterator classes and demonstrate a client of the class using an iterator to obtain values from a sequence. The first example illustrates how to define a class whose objects support iterators; the second example presents a computer guessing game that shows how iterators can be used to process sequences of indeterminate size. The new iterator mechanism in Python 2.2 enables the language to provide a significant performance enhancement over previous versions and improves software design by allowing programmers to separate iteration behavior from random-access behavior.

---

2. Before reading this book, download the most recent release of Python from the **python.org** Web site. As new versions are released, we will test our code and update the **www.deitel.com** Web site. Before you read each chapter, please go to our Web site for these updates.

Generators. Generators also provide new performance and design benefits. A generator is a "resumable function" that remembers its state between invocations. Often, a program writes a generator to define how to produce elements of a sequence in a simple, straightforward manner. Generators also are useful for performing recursive tasks or tasks that would require complex logic and state information to implement with "traditional" functions. Section O.2 discusses generators in the context of these issues and defines two versions of a generator that computes the Fibonacci sequence. The first version produces the next value in the sequence indefinitely; the second produces all the sequence values up through and including a user-defined *n*th value.

Some Notes to Instructors

Students Enjoy Learning a Leading-Edge Language
Students are highly motivated by the fact that they are learning a leading-edge language, Python, and a leading-edge programming paradigm (object-oriented programming) that will be immediately useful to them as they enter the business world. This increases their enthusiasm for the material—which is essential when you consider that there is much more to learn in a Python course now that students must master both the base language and substantial modules as well.

A World of Object Orientation
In the late 1990s, universities were still emphasizing procedural programming. The leading-edge courses were using object-oriented C++, but these courses generally mixed a substantial amount of procedural programming with object-oriented programming—something that C++ lets programmers do. Many instructors now are emphasizing a pure object-oriented programming approach. This book takes a predominantly object-oriented approach because of the object orientation provided in Python.

Focus of the Book
Our goal was clear: Produce a Python textbook for introductory-level university courses in computer programming aimed at students with little or no programming experience, yet offer the depth and the rigorous treatment of theory and practice demanded by both professionals and students in traditional, upper-level programming courses. To meet these objectives, we produced a comprehensive book that patiently teaches the principles of computer programming and of the Python language, including control structures, object-oriented programming, Python modules, graphical-user-interface concepts, event-driven programming and more. After mastering the material in this book, students will be well-prepared to program "industrial-strength" applications in Python.

Multimedia-Intensive Communications
People want to communicate. Sure, they have been communicating since the dawn of civilization, but the potential for information exchange has increased dramatically with the evolution of various technologies. Until recently, even computer communications were limited mostly to digits, alphabetic characters and special characters. The current wave of communication technology involves the distribution of multimedia—people enjoy using applications that transmit color pictures, animations, voices, audio clips and even

full-motion color video over the Internet. At some point, we will insist on three-dimensional, moving-image transmission.

There have been predictions that the Internet will eventually replace radio and television as we know them today. Similarly, it is not hard to imagine newspapers, magazines and books delivered to "the palm of your hand" (or even to special eyeglasses) via wireless communications. Many newspapers and magazines already offer Web-based versions, and some of these services have spread to the wireless world. When cell phones were first introduced, they were large and cumbersome. Today, they are small devices that fit in our pockets, and many are Internet-enabled. Wireless technology already enables streaming-video and graphics-intensive services, such as video conference calls and multi-player video games. Chapter 23, Case Study: Online Bookstore; and Chapter 24, Multimedia, demonstrate the possibilities of wireless Internet and multimedia applications.

Teaching Approach

Python How to Program contains a rich collection of examples, exercises and projects drawn from many fields and designed to provide students with a chance to solve interesting, real-world problems. The code examples in this text have been tested on Windows 2000 and Linux. The book concentrates on the principles of good software engineering, and stresses program clarity. We are educators who teach edge-of-the-practice topics in industry classrooms worldwide. We avoid arcane terminology and syntax specifications in favor of teaching by example. The text emphasizes good pedagogy.[3]

LIVE-CODE™ Teaching Approach

Python How to Program is loaded with numerous examples. This style exemplifies the way we teach and write about programming and is the focus of our multimedia *Cyber Classrooms* and Web-based training courses. Each new concept is presented in the context of a complete, working example that is immediately followed by one or more windows showing the program's input/output dialog or graphical display. We call this method of teaching and writing the ***LIVE-CODE™ Approach***. *We use programming languages to teach programming languages.* Reading the examples in the text is much like entering and running them on a computer.

World Wide Web Access

All of the examples for *Python How to Program* (and our other publications) are available on the Internet free for download from the following Web sites:

```
www.deitel.com
www.prenhall.com/deitel
```

Registration is quick and easy. We suggest downloading all the examples, then running each program as you read the corresponding text. Make changes to the examples and immediately see the effects of those changes—this is a great way to learn programming. We also provide instructions for installing various software used in this book (e.g., Apache

---

3. We use a different font from the text font when referring to the elements in a graphical user interface (GUI) or when referring to a location on a computer disk. Our convention is to emphasize GUI features in a sans-serif bold Helvetica font (e.g., **Edit** menu) and to emphasize programming elements in a serif bold Courier font (e.g., **def**).

Software Foundation's Apache Web Server). Additional setup instructions for other Web servers and software can be found at our Web sites with the examples. [*Note*: This is copyrighted material. Feel free to use it as you study, but you may not republish any portion of it in any form without explicit permission from Prentice Hall and the authors.]

Objectives

Each chapter begins with objectives that inform students of what to expect and give them an opportunity, after reading the chapter, to determine whether they have met the intended goals. The objectives serve as confidence builders and as a source of positive reinforcement.

Quotations

The chapter objectives are followed by sets of quotations. Some are humorous, some are philosophical and some offer interesting insights. We have found that students enjoy relating the quotations to the chapter material. Many of the quotations are worth a "second look" *after* you read each chapter.

Outline

The chapter outline enables students to approach the material in top-down fashion. Along with the chapter objectives, the outline helps students anticipate future topics and set a comfortable and effective learning pace.

Approximately 14,930 Lines of Code in 281 Example Programs (with Program Outputs)

We present Python features in the context of complete, working Python programs. The programs range in size from just a few lines of code to substantial examples containing several hundred lines of code. All examples are available on the book's CD or free for download at **www.deitel.com**.

589 Illustrations/Figures

An abundance of charts, line drawings and program outputs is included. The discussion of control structures, for example, features carefully drawn flowcharts. [*Note*: We do not teach flowcharting as a program-development tool, but we do use a brief, flowchart-oriented presentation to explain the precise operation of each Python control structure.]

412 Programming Tips

We have included programming tips to help students focus on important aspects of program development. We highlight hundreds of these tips in the form of *Good Programming Practices*, *Common Programming Errors*, *Testing and Debugging Tips, Performance Tips*, *Portability Tips*, *Software Engineering Observations* and *Look-and-Feel Observations*. These tips and practices represent the best the authors have gleaned from a combined seven decades of programming and teaching experience. One of our students—a mathematics major—told us that she feels this approach is like the highlighting of axioms, theorems and corollaries in mathematics books; it provides a foundation on which to build good software.

73 Good Programming Practices

Good Programming Practices *are tips that call attention to techniques that will help students produce better programs. When we teach introductory courses to nonprogrammers, we state that the "buzzword" for each course is "clarity," and we tell the students that we will highlight (in these Good Programming Practices) techniques for writing programs that are clearer, more understandable and more maintainable.*

125 Common Programming Errors

Students learning a language—especially in their first programming course—tend to make certain kinds of errors frequently. Pointing out these Common Programming Errors *reduces the likelihood that students will make the same mistakes. It also shortens long lines outside instructors' offices during office hours!*

31 Testing and Debugging Tips

When we first designed this "tip type," we thought the tips would contain suggestions strictly for exposing bugs and removing them from programs. In fact, many of the tips describe aspects of Python that prevent "bugs" from getting into programs in the first place, thus simplifying the testing and debugging process.

35 Performance Tips

In our experience, teaching students to write clear and understandable programs is by far the most important goal for a first programming course. But students also want to write programs that run the fastest, use the least memory, require the smallest number of keystrokes or dazzle in other ways. Students really care about performance and they want to know what they can do to produce the most efficient programs. We have included 35 Performance Tips *that highlight opportunities for improving program performance—making programs run faster or minimizing the amount of memory that they occupy.*

23 Portability Tips

We include Portability Tips *to help students write portable code and to provide insights on how Python achieves its high degree of portability.*

92 Software Engineering Observations

The object-oriented-programming paradigm necessitates a complete rethinking of the way we build software systems. Python is an effective language for achieving good software engineering. The Software Engineering Observations *highlight architectural and design issues that affect the construction of software systems, especially large-scale systems. Much of what the student learns here will be useful in upper-level courses and in industry as the student begins to work with large, complex real-world systems.*

21 Look-and-Feel Observations

We provide Look-and-Feel Observations *to highlight graphical-user-interface conventions. These observations help students design attractive, user-friendly graphical user interfaces that conform to industry norms.*

Summary (1462 Summary bullets)
Each chapter ends with additional pedagogical devices. We present a thorough, bullet-list-style summary of the chapter. On average, there are 43 summary bullets per chapter. This helps the students review and reinforce key concepts.

Terminology (2485 Terms)
We include an alphabetized list of the important terms defined in the chapter in a *Terminology* section. Again, this serves as further reinforcement. On average, there are 73 terms per chapter. Each term also appears in the index, so the student can locate terms and definitions quickly.

615 Self-Review Exercises and Answers (Count Includes Separate Parts)
Extensive self-review exercises and answers are included for self-study. These questions and answers give the student a chance to build confidence with the material and prepare for the regular exercises. Students should be encouraged to attempt all the self-review exercises and check their answers.

370 Exercises (Solutions in Instructor's Manual; Count Includes Separate Parts)
Each chapter concludes with a substantial set of exercises that involve simple recall of important terminology and concepts; writing individual Python statements; writing small portions of Python methods and classes; writing complete Python methods, classes and applications; and writing major projects. These exercises cover a wide variety of topics, enabling instructors to tailor their courses to the unique needs of their audiences and to vary course assignments each semester. Instructors can use the exercises to form homework assignments, short quizzes and major examinations. The solutions for the exercises are included in the *Instructor's Manual* and on the disks *available only to instructors* through their Prentice-Hall representatives. [*NOTE*: **Please do not write to us requesting the instructor's manual**. **Distribution of this publication is strictly limited to college professors teaching from the book. Instructors may obtain the solutions manual from their regular Prentice Hall representatives**. **We regret that we cannot provide the solutions to professionals**.] Solutions to approximately half the exercises are included on the *Python Multimedia Cyber Classroom* CD-ROM (available in April 2002 at **www.InformIT.com/cyberclassrooms**; also see the last few pages of this book or visit **www.deitel.com** for ordering instructions). Also available in April 2002 is the boxed product, *The Complete Python Training Course*, which includes both our textbook, *Python How to Program* and the *Python Multimedia Cyber Classroom*. All of our *Complete Training Course* products are available at bookstores and online booksellers, including **www.InformIT.com**.

Approximately 4212 Index Entries (with approximately 5733 Page References)
We have included an extensive Index at the back of the book. Using this resource, students can search for any term or concept by keyword. The Index is especially useful to practicing programmers who use the book as a reference. Each of the 2485 terms in the Terminology sections appears in the Index (along with many more index items from each chapter). Students can use the Index in conjunction with the Terminology sections to ensure that they have covered the key material in each chapter.

"Double Indexing" of All Python LIVE-CODE™ Examples

Python How to Program has 281 LIVE-CODE™ examples, which we have "double in-dexed." For every Python source-code program in the book, we took the file name with the `.py` extension, such as `Book.py`, and indexed it both alphabetically (in this case, under "B") and as a subindex item under "Examples." This makes it easier to find examples that are demonstrating particular features.

Software Included with Python How to Program

There are a number of Python products available for download from the Internet. We wrote *Python How to Program* using the Python Core Language 2.2 final release, and other soft-ware we have included on the CD-ROM that accompanies this text. This software includes the Apache Web Server 1.3.22, from the Apache Software Foundation and Alice99, a mul-timedia application for creating 2D and 3D graphics. The CD-ROM includes the wireless browser, Pixo 2.1, the IBM WebSphere Voice Server 1.5, a text-to-speech engine and Web-ware 0.6 for Python, which contains PSP software components for developing Web-based server-side applications. As we mentioned earlier, Python 2.2 was released almost to the minute that we released *Python How to Program* to Prentice Hall for publication. Some of the Python software packages you many want to use may not yet work with Python 2.2. If that is the case, please try running those software packages on Python 2.1, which you can download from **www.python.org**. If you still experience problems, please check the Py-thon FAQs at **www.deitel.com**. The CD-ROM contains Windows and Linux versions of the software, where possible.

Ancillary Package for Python How to Program

Python How to Program has extensive ancillary materials for instructors teaching from the book. The *Instructor's CD* contains the *Instructor's Manual* with solutions to the vast ma-jority of the end-of-chapter exercises and a *Test Item File* of multiple-choice questions (ap-proximately two per book section). In addition, we provide PowerPoint® slides containing all the code and figures in the text. You are free to customize these slides to meet your own classroom needs. Prentice Hall provides a *Companion Web Site* (**www.prenhall.com/deitel**) that includes resources for instructors and students. For instructors, the Web site has a *Syllabus Manager* for course planning, links to the PowerPoint slides and reference materials from the appendices of the book (such as operator precedence chart, character set and Web resources). For students, the Web site provides chapter objectives, additional self-review exercises and answers, chapter highlights and reference materials.

Python Multimedia Cyber Classroom and The Complete Python Training Course

We have prepared an interactive, multimedia, CD-ROM-based, software version of *Python How to Program,* called the *Python Multimedia Cyber Classroom.* This resource is loaded with e-learning features that are ideal for both education and reference. The *Cyber Class-room* is packaged with the textbook at a discount in *The Complete Python Training Course.* If you already have the book and would like to purchase the *Python Multimedia Cyber Classroom* separately, please visit **www.InformIT.com/cyberclassrooms**. The

ISBN number for the CD-ROM format of the *Python Multimedia Cyber Classroom* is 0-13-067376-5 and the Web-based training format is ISBN number 0-13-067381-1. Many Deitel™ *Cyber Classrooms* are available in CD-ROM and Web-based formats.

The CD provides an introduction in which the authors overview the *Cyber Classroom*'s features. The textbook's 281 LIVE-CODE™ example Python programs truly "come alive" in the *Cyber Classroom*. If you are viewing a program and want to execute it, you simply click the lightning-bolt icon, and the program will run. You immediately will see—and hear, when working with audio-based multimedia programs—the program's outputs. If you want to modify a program and see the effects of your changes, simply click the floppy-disk icon that causes the source code to be "lifted off" the CD and "dropped into" one of your own directories so you can edit the text, recompile the program and try out your new version. Click the audio icon, and one of the authors will discuss the program and "walk you through" the code.

The *Cyber Classroom* also provides navigational aids, including extensive hyperlinking. The *Cyber Classroom* is browser based, so it remembers sections that you have visited recently and allows you to move forward or backward among these sections. The thousands of index entries are hyperlinked to their text occurrences. Furthermore, when you enter a term using the "find" feature, the *Cyber Classroom* will locate occurrences of that term throughout the text. The Table of Contents entries are "hot," so clicking a chapter name takes you immediately to that chapter.

Students like the fact that solutions to approximately half the exercises in the book are included with the *Cyber Classroom*. Studying and running these extra programs is a great way for students to enhance their learning experience.

Students and professional users of our *Cyber Classrooms* tell us that they like the interactivity and that the *Cyber Classroom* is an effective reference due to its extensive hyperlinking and other navigational features. We received an e-mail from a person who said that he lives "in the boonies" and cannot take a live course at a university, so the *Cyber Classroom* provided an ideal solution to his educational needs.

Professors tell us that their students enjoy using the *Cyber Classroom* and spend more time on the courses and master more of the material than in textbook-only courses. For a complete list of the available and forthcoming *Cyber Classrooms* and *Complete Training Courses*, see the *Deitel™ Series* page at the beginning of this book, the product listing and ordering information at the end of this book or visit **www.deitel.com**, **www.prenhall.com/deitel** and **www.InformIT.com/deitel**.

Deitel e-Learning Initiatives

e-Books and Support for Wireless Devices
Wireless devices will play an enormous role in the future of the Internet. Given recent bandwidth enhancements and the emergence of 2.5 and 3G technologies, it is projected that, within two years, more people will access the Internet through wireless devices than through desktop computers. Deitel & Associates, Inc., is committed to wireless accessibility and has recently published *Wireless Internet & Mobile Business How to Program*. To fulfill the needs of a wide range of customers, we are developing our content both in traditional print formats and in newly developed electronic formats, such as e-books so that students and professors can

access content virtually anytime, anywhere. Visit **www.deitel.com** for periodic updates on these initiatives.

e-Matter

Deitel & Associates, Inc., is partnering with Prentice Hall's parent company, Pearson PLC, and its information technology Web site, **InformIT.com**, to launch the Deitel e-Matter series at **www.InformIT.com/deitel**. This series will provide professors, students and professionals with an additional source of information on specific programming topics. e-Matter consists of stand-alone sections taken from published texts, forthcoming texts or pieces written during the Deitel research-and-development process. Developing e-Matter based on pre-publication books allows us to offer significant amounts of the material to early adopters for use in courses. Some possible Python e-Matter titles we are considering include *Object-Oriented Programming in Python*; *Graphical User Interface Programming in Python* and *Multithreading in Python*.

Course Management Systems: WebCT, Blackboard, and CourseCompass

We are working with Prentice Hall to integrate our *How to Program Series* courseware into three Course Management Systems: WebCT, Blackboard™ and CourseCompass. These Course Management Systems enable instructors to create, manage and use sophisticated Web-based educational programs. Course Management System features include course customization (such as posting contact information, policies, syllabi, announcements, assignments, grades, performance evaluations and progress tracking), class and student management tools, a gradebook, reporting tools, communication tools (such as chat rooms), a whiteboard, document sharing, bulletin boards and more. Instructors can use these products to communicate with their students, create online quizzes and tests from questions directly linked to the text and automatically grade and track test results. For more information about these upcoming products, visit **www.deitel.com/whatsnew.html**. For demonstrations of existing WebCT, Blackboard and CourseCompass courses, visit **cms.pren_hall.com/WebCT**, **cms.prenhall.com/Blackboard** and **cms.prenhall.com/CourseCompass**, respectively.

Deitel and InformIT Newsletters

Deitel Column in the InformIT Newsletters

Deitel & Associates, Inc., contributes a weekly column to the popular *InformIT* newsletter, currently subscribed to by more than 800,000 IT professionals worldwide. For opt-in registration, visit **www.InformIT.com**.

Deitel Newsletter

Our own free, opt-in newsletter includes commentary on industry trends and developments, links to articles and resources from our published books and upcoming publications, information on future publications, product-release schedules, programming tips, errata, additional insights and more. For opt-in registration, visit **www.deitel.com**.

Acknowledgments

One of the great pleasures of writing a textbook is acknowledging the efforts of many people whose names may not appear on the cover, but whose hard work, cooperation, friendship and understanding were crucial to the production of the book.

Many other people at Deitel & Associates, Inc., devoted long hours to this project.

- Rashmi Jayaprakash, a graduate of Boston University with a degree in Computer Science, co-authored Appendices D, F and L and served as the project manager. She edited and reviewed Chapters 1–25 and the appendices and managed the review process for the book.

- Su Zhang, a graduate of McGill University with a Master's in Computer Science, co-authored Chapters 24 and 25 and contributed to Chapters 3, 5, 6, 17, 18 and 21.

- Tem R. Nieto, a graduate of the Massachusetts Institute of Technology, is Director of Product Development at Deitel & Associates, Inc. He is co-author of *Internet & World Wide Web How to Program, Second Edition; XML How to Program*; *Perl How to Program*; *Visual Basic .NET How to Program, Second Edition*; *C# How to Program* and *Wireless Internet and Mobile Business How to Program*. Tem co-authored Chapters 15–16 and edited Chapters 23–25 and Appendices D and F–L.

- Sean E. Santry, a graduate of Boston College with degrees in Computer Science and Philosophy, is Director of Software Development at Deitel & Associates, Inc., and co-author of *Advanced Java 2 Platform How to Program*. Sean contributed to Chapters 7, 12, 16–21, 23 and 25 and tested examples on Linux. Sean was instrumental in the finishing stages of the project.

- Kyle Lomelí, a graduate of Oberlin College with a degree in Computer Science and a minor in East Asian Studies, contributed to Chapters 7, 10–13, 17–19 and 24–25 and tested all the examples on Windows.

- Matthew R. Kowalewski, a graduate of Bentley College with a degree in Accounting Informations Systems, is the Director of Wireless Development at Deitel & Associates, Inc. He contributed to Chapter 16, 23 and 25 and Appendices G–N. He also reviewed Appendices A–F and edited the Index.

- Jonathan Gadzik, a graduate of the Columbia University School of Engineering and Applied Science with a major in Computer Science, contributed to Chapters 15 and 16 and reviewed Chapters 7, 12–14, 18–20 and 25.

- Betsy DuWaldt, a graduate of Metropolitan State College of Denver with a major in Technical Communications (Writing and Editing emphasis) and a minor in Computer Information Systems, is Editorial Director at Deitel & Associates, Inc. She co-authored the Preface and Chapter 1, edited the manuscript and managed the permissions process for the book.

- Laura Treibick, a graduate of the University of Colorado at Boulder with a degree in Photography and Multimedia, is Director of Multimedia at Deitel & Associates, Inc. She contributed to Chapters 18 and 24 and enhanced many of the graphics throughout the text.

- Barbara Deitel applied the copy edits to the manuscript. She did this in parallel with handling her extensive financial and administrative responsibilities at Deitel & Associates, Inc., which include serving as Chief Financial Officer. [Everyone at the company works on book content.]

- Abbey Deitel, a graduate of Carnegie Mellon University's Industrial Management Program and President of Deitel & Associates, Inc., co-authored Chapter 21, and edited the Preface and several of the book's chapters. She recruited 40 additional full-time employees and interns during 2001 and leased, equipped and furnished our second building to create the work environment from which *Python How to Program* and our other year 2001 publications were produced. She also suggested the title for the *How to Program* series.

We would also like to thank the participants in the Deitel & Associates, Inc., College Internship Program.[4]

- Christina J. Courtemarche, a senior at Boston University in Computer Science, was instrumental in the development of this manuscript. She co-authored Appendix D and contributed to Chapters 1–17, 19–22, 24, and Appendix F. She coded and tested the examples and solved the exercises for Chapters 1–14, 16, 20 and 22.

- Alex Rasin, a graduate of Brandeis University with a degree in Computer Science and a graduate student at Brown University, coded examples in Chapters 5, 9, 10, 11, 13, 21 and 24.

- Jeffrey Hamm, a sophomore at Northeastern University in Computer Science, reviewed Chapter 24.

- Christina Carney, a senior in Psychology and Business at Framingham State College, helped with the Preface and the Bibliography.

- Brian Foster, a sophomore at Northeastern University in Computer Science, contributed to ancillaries for book and helped with the Preface and Bibliography.

- Mike Preshman, a sophomore at Northeastern University with a major in Computer Science and minors in Electrical Engineering and Math, contributed to Appendices G–L, tested the code examples and helped with the Bibliography and ancillary materials.

- Matthew Rubino, a sophomore at Northeastern University in Computer Science, contributed to the ancillary materials.

- Adam Sparrow, a senior at Bentley College with a major in Computer Information Systems, contributed to the ancillary materials.

We are fortunate to have been able to work on this project with the talented and dedicated team of publishing professionals at Prentice Hall. We especially appreciate the extraordinary efforts of our Computer Science editor, Petra Recter and her boss—our mentor in publishing—Marcia Horton, Editorial Director of Prentice-Hall's Engineering and Computer Science Division. Camille Trentacoste and her boss Vince O'Brien did a marvelous job managing the production of the book. Sarah Burrows handled editorial responsibilities on the book's extensive ancillary package.

The *Python Multimedia Cyber Classroom* was developed in parallel with *Python How to Program*. We sincerely appreciate the "new media" insight, savvy and technical expertise of our electronic-media editors, Mark Taub and Karen McLean. They and project manager Mike Ruel did a wonderful job bringing the *Python Multimedia Cyber Classroom* and *The Complete Python Training Course* to publication.

We owe special thanks to the creativity of Tamara Newnam (`smart_art@earth-link.net`), who produced the art work for our programming-tip icons and for the cover. She created the delightful creature who shares with you the book's programming tips. Barbara Deitel and Harvey Deitel contributed the bugs' names for the front cover.

We wish to acknowledge the efforts of our first- and second-round reviewers and to thank Crissy Statuto of Prentice Hall, who recruited the reviewers and managed the review process. Adhering to a tight time schedule, these reviewers scrutinized the text and the programs, providing countless suggestions for improving the accuracy and completeness of the presentation.

Python How to Program reviewers:

Guido Van Rossum (Creator of Python)
Mike Fletcher (PyOpenGL Project)
Jeremy Hylton (Pythonlabs at Digital Creations)
Andreas Jung (Zope Corporation)
Alex Martelli (Senior Software Consultant, think3, Inc.)
Russell Nelson (Vice President, Open Source Initiative)
Uche Ogbuji (CEO, Fourthought, Inc.)
Pete Shinners (Pygame Maintainer)
Aahz (Writer & Trainer)
Rob Andrews (Webmaster, Useless Python)
Ian Bicking (Colorstudy Web Design)
Carl Burnham (Southpoint)
Nathan Clegg (Geerbox)
Luis Cortes (Association for Computing Machinery, IEEE)
David Currie (Netdecisions)
Kevin Dorff (Honeywell)
Cam Farnell (Consultant)
Doug Fort (Downright Software LLC)
Charles Fry (thesundancekid.org)
Robert Fulkerson (University of Nebraska at Omaha)
Gnanavel Gnana Arun Ganesh (Arun Microsystems)
Alan Gauld (Author of *Learn to Program Using Python*)
Dean Goodmanson (Renaissance Learning, Inc.)

Chris Gonnerman (Owner, New Century Computers)
David den Harring (Consultant)
Michael Hudson (University of Bristol)
Wayne Izatt (Independent Software Developer)
Curtis Jensen (University of California, San Diego)
Nicolas Kauer (University of Wisconsin)
Tim Keating (QA Engineer, Origin Systems)
Andredi Kulakov (Python Programmer)
Cameron Laird (Phasit, Inc.)
Tripp Lilley (Webware)
Andrew Markebo (Telelogic AB)
Rick McGowen (Unicode Consortium)
Sean McGrath (Propylon)
Zan Ouyang (Core Technology)
Todd Pitts (Consultant)
Brandon Rogers (Washington State University student)
Richard H.C. Seabrook (Anne Arundel Community College)
Christine Shannon (Centre College)
John W. Shipman (New Mexico Tech Computer Center)
David R. Sopha (Sopha Consulting, Inc.)
Ken Stone (Tellium)
Terry Sullivan (pantos.org)
Geoff Talvola (Webware)
Vladimir Toncar (Tiny Software)
Howard Whitston (Lawrence Technological University)
Collin Williams (Consultant)
Jesse Wilkins (Metalinear Media)
Jody Winston (xprt Computer Consulting Inc.)
Joerg Volelke (FeLis)
Kirby Urner (Curriculum Writer, 4D Solutions)

We would like to thank the Python Software Foundation for permission to use their software (Python, IDLE and **Tkinter**) in our illustrations to demonstrate many Python concepts. Copyright © 2001, Python Software Foundation; Copyright © 2000 BeOpen.com; Copyright © 1995–2001 Corporation for National Research Initiatives; Copyright © 1991-1995 Stichting Mathematisch Centrum, Amsterdam.

Modules and their Copyrights
The nature of Python is that members of the Python community create additions to the language called modules. Like Python, many of these modules are created as open source and are available for anyone to use in their Python programs. Here, we present a list of the modules used in this book and their creators.

- **Pmw** (**pmw.sourceforge.net**). Copyright © 1997, 1998, 1999, 2000, 2001 Telstra Corporation Limited, Australia (ACN 051 775 556).
  PyXML (**pyxml.sourceforge.net**). Author: PyXML Special Interest Group.

- **4Suite** (`4suite.org/4SuiteIndex.html`). Copyright © 2000–2001 Fourthought, Inc.

- **MySQLdb** (`sourceforge.net/projects/mysql-python`). Author: Andy Dustman.

- **PyOpenGL** (`pyopengl.sourceforge.net`). Copyright © 1997–1998 by James Hugunin, Cambridge MA, USA, Thomas Schwaller, Munich, Germany and David Ascher, San Francisco CA, USA. PyOpenGL 1.5.6 Copyright © 1997–1998, 2000–2001.

- **Pygame** (`www.pygame.org`). Author: Pete Shinners. Copyright © Pygame, 2001 Python Game Development.

- **Webware** (`webware.sourceforge.net`). Copyright © 1999–2001 by Chuck Esterbrook.

We would sincerely appreciate your comments, criticisms, corrections and suggestions for improving the text. Please address all correspondence to:

`deitel@deitel.com`

We will respond promptly.

Well, that's it for now. Welcome to the exciting world of Python programming. We hope you enjoy this look at leading-edge computer applications development. Good luck!

Dr. Harvey M. Deitel
Paul J. Deitel
Jonathan Liperi
Ben Wiedermann

About the Authors

Dr. Harvey M. Deitel, CEO and Chairman of Deitel & Associates, Inc., has 41 years experience in the computing field, including extensive industry and academic experience. Dr. Deitel earned B.S. and M.S. degrees from the Massachusetts Institute of Technology and a Ph.D. from Boston University. He worked on the pioneering virtual-memory operating-systems projects at IBM and MIT that developed techniques now widely implemented in systems such as UNIX, Linux and Windows NT. He has 20 years of college teaching experience, including earning tenure and serving as the Chairman of the Computer Science Department at Boston College before founding Deitel & Associates, Inc., with his son, Paul J. Deitel. He is the author or co-author with Paul Deitel of several dozen books and multimedia packages and is writing many more. With translations published in Japanese, Russian, Spanish, Traditional Chinese, Simplified Chinese, Korean, French, Polish, Italian and Portuguese, Dr. Deitel's texts have earned international recognition. Dr. Deitel has delivered hundreds of professional seminars to major corporations, government organizations and various branches of the military.

Paul J. Deitel, Executive Vice President and Chief Technical Officer of Deitel & Associates, Inc., is a graduate of the Massachusetts Institute of Technology's Sloan School of Management, where he studied Information Technology. Through Deitel & Associates, Inc., he has delivered Java, C, C++ and Internet and World Wide Web courses to industry

clients including Compaq, Sun Microsystems, White Sands Missile Range, Rogue Wave Software, Boeing, Dell, Stratus, Fidelity, Cambridge Technology Partners, Open Environment Corporation, One Wave, Hyperion Software, Lucent Technologies, Adra Systems, Entergy, CableData Systems, NASA at the Kennedy Space Center, the National Severe Storm Laboratory, IBM and many other organizations. He has lectured on C++ and Java for the Boston Chapter of the Association for Computing Machinery and has taught satellite-based Java courses through a cooperative venture of Deitel & Associates, Inc., Prentice Hall and the Technology Education Network. He and his father, Dr. Harvey M. Deitel, are the world's best-selling Computer Science textbook authors.

Jonathan Liperi is a senior at Boston University where he has been accepted into the Computer Science department's BA/MA program. He will earn his Master's degree in Computer Science in May 2003. His coursework has included advanced algorithms, queueing theory, computer architecture, computer networks, artificial intelligence, computer graphics, database systems, software engineering and various programming courses (C, C++, Python and Java).

Ben Wiedermann graduated from Boston University *magna cum laude* with a degree in Computer Science and a minor in Theater Arts. Ben plans to pursue post-graduate work in programming-language theory. Other Deitel publications to which he has contributed include *Java How to Program, Fourth Edition*; *C++ How to Program, Third Edition*; *Perl How to Program*; *Internet and World Wide Web How to Program, Second Edition*; *XML How to Program*; *e-Business & e-Commerce How to Program* and *C How to Program, Third Edition*.

About Deitel & Associates, Inc.

Deitel & Associates, Inc., is an internationally recognized corporate training and content-creation organization specializing in Internet/World Wide Web software technology, e-business/e-commerce software technology, object technology and computer programming languages education. The company provides courses on Internet and World Wide Web programming, wireless Internet programming, object technology and major programming platforms and languages, such as Visual Basic .NET, C#, Java, advanced Java, C, C++, XML, Perl, Python and more. The founders of Deitel & Associates, Inc., are Dr. Harvey M. Deitel and Paul J. Deitel. The company's clients include many of the world's largest computer companies, government agencies, branches of the military and business organizations. Through its 25-year publishing partnership with Prentice Hall, Deitel & Associates, Inc., publishes leading-edge programming textbooks, professional books, interactive CD-ROM-based multimedia *Cyber Classrooms*, *Complete Training Courses*, e-books, e-matter, Web-based training courses and course management systems e-content. Deitel & Associates, Inc., and the authors can be reached via e-mail at:

```
deitel@deitel.com
```

To register for *THE DEITEL™ BUZZ* e-mail newsletter and to learn more about Deitel & Associates, Inc., its publications and its worldwide corporate on-site curriculum, see the last few pages of this book or visit:

```
www.deitel.com
```

Individuals wishing to purchase Deitel books, *Cyber Classrooms*, *Complete Training Courses* and Web-based training courses can do so through bookstores, online booksellers and:

```
www.deitel.com
www.prenhall.com/deitel
www.InformIT.com/deitel
www.InformIT.com/cyberclassrooms
```

Bulk orders by corporations and academic institutions should be placed directly with Prentice Hall. See the last few pages of this book for worldwide ordering details. To register for the InformIT e-mail newsletter which contains a weekly column by the Deitels, visit **www.InformIT.com**.

The World Wide Web Consortium (W3C)

Deitel & Associates, Inc., is a member of the *World Wide Web Consortium (W3C)*. The W3C was founded in 1994 "to develop common protocols for the evolution of the World Wide Web." As a W3C member, Deitel & Associates, Inc., holds a seat on the W3C Advisory Committee (the company's representative is our Chief Technology Officer, Paul Deitel). Advisory Committee members help provide "strategic direction" to the W3C through meetings held around the world. Member organizations also help develop standards recommendations for Web technologies (such as XHTML, XML and many others) through participation in W3C activities and groups. Membership in the W3C is intended for companies and large organizations. To obtain information on becoming a member of the W3C visit **www.w3.org/Consortium/Prospectus/Joining**.