

LEARNING MADE EASY



3rd Edition

# Beginning Programming with Python®

for  
**dummies**®  
A Wiley Brand



Get started with your  
first Python application

Troubleshoot and resolve  
errors in your code

Code anywhere and use  
magic functions to do it

**John Paul Mueller**

Dreams about teaching real pythons  
to write code



# Beginning Programming with Python\*

3rd Edition

by John Paul Mueller

**for  
dummies**<sup>®</sup>  
A Wiley Brand

# **Beginning Programming with Python® For Dummies®, 3rd Edition**

Published by: **John Wiley & Sons, Inc.**, 111 River Street, Hoboken, NJ 07030-5774, [www.wiley.com](http://www.wiley.com)

Copyright © 2023 by John Wiley & Sons, Inc., Hoboken, New Jersey

Media and software compilation copyright © 2023 by John Wiley & Sons, Inc. All rights reserved.

Published simultaneously in Canada

No part of this publication may be reproduced, stored in a retrieval system or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, scanning or otherwise, except as permitted under Sections 107 or 108 of the 1976 United States Copyright Act, without the prior written permission of the Publisher. Requests to the Publisher for permission should be addressed to the Permissions Department, John Wiley & Sons, Inc., 111 River Street, Hoboken, NJ 07030, (201) 748-6011, fax (201) 748-6008, or online at <http://www.wiley.com/go/permissions>.

**Trademarks:** Wiley, For Dummies, the Dummies Man logo, Dummies.com, Making Everything Easier, and related trade dress are trademarks or registered trademarks of John Wiley & Sons, Inc. and may not be used without written permission. Python is a registered trademark of Python Software Foundation Corporation. All other trademarks are the property of their respective owners. John Wiley & Sons, Inc. is not associated with any product or vendor mentioned in this book.

LIMIT OF LIABILITY/DISCLAIMER OF WARRANTY: WHILE THE PUBLISHER AND AUTHORS HAVE USED THEIR BEST EFFORTS IN PREPARING THIS WORK,
---

THEY MAKE NO REPRESENTATIONS OR WARRANTIES WITH RESPECT TO THE ACCURACY OR COMPLETENESS OF THE CONTENTS OF THIS WORK AND SPECIFICALLY DISCLAIM ALL WARRANTIES, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. NO WARRANTY MAY BE CREATED OR EXTENDED BY SALES REPRESENTATIVES, WRITTEN SALES MATERIALS OR PROMOTIONAL STATEMENTS FOR THIS WORK. THE FACT THAT AN ORGANIZATION, WEBSITE, OR PRODUCT IS REFERRED TO IN THIS WORK AS A CITATION AND/OR POTENTIAL SOURCE OF FURTHER INFORMATION DOES NOT MEAN THAT THE PUBLISHER AND AUTHORS ENDORSE THE INFORMATION OR SERVICES THE ORGANIZATION, WEBSITE, OR PRODUCT MAY PROVIDE OR RECOMMENDATIONS IT MAY MAKE. THIS WORK IS SOLD WITH THE UNDERSTANDING THAT THE PUBLISHER IS NOT ENGAGED IN RENDERING PROFESSIONAL SERVICES. THE ADVICE AND STRATEGIES CONTAINED HEREIN MAY NOT BE SUITABLE FOR YOUR SITUATION. YOU SHOULD CONSULT WITH A SPECIALIST WHERE APPROPRIATE. FURTHER, READERS SHOULD BE AWARE THAT WEBSITES LISTED IN THIS WORK MAY HAVE CHANGED OR DISAPPEARED BETWEEN WHEN THIS WORK WAS WRITTEN AND WHEN IT IS READ. NEITHER THE PUBLISHER NOR AUTHORS SHALL BE LIABLE FOR ANY LOSS OF PROFIT OR ANY OTHER COMMERCIAL DAMAGES, INCLUDING BUT NOT LIMITED TO SPECIAL, INCIDENTAL, CONSEQUENTIAL, OR OTHER DAMAGES.

For general information on our other products and services, please contact our Customer Care Department

within the U.S. at 877-762-2974, outside the U.S. at 317-572-3993, or fax 317-572-4002. For technical support, please visit <https://hub.wiley.com/community/support/dummies>.

Wiley publishes in a variety of print and electronic formats and by print-on-demand. Some material included with standard print versions of this book may not be included in e-books or in print-on-demand. If this book refers to media such as a CD or DVD that is not included in the version you purchased, you may download this material at <http://booksupport.wiley.com>. For more information about Wiley products, visit [www.wiley.com](http://www.wiley.com).

Library of Congress Control Number: 2022948492

ISBN: 978-1-119-91377-1 (pbk);

ISBN 978-1-119-91378-8 (ebk); ISBN 978-1-119-91379-5 (ebk)

# Beginning Programming with Python® For Dummies®

To view this book's Cheat Sheet, simply go to [www.dummies.com](http://www.dummies.com) and search for “Beginning Programming with Python For Dummies Cheat Sheet” in the Search box.

## Table of Contents

[Cover](#)

[Title Page](#)

[Copyright](#)

[Introduction](#)

[About This Book](#)

[Foolish Assumptions](#)

[Icons Used in This Book](#)

[Beyond the Book](#)

[Where to Go from Here](#)

[Part 1: Getting Started with Python](#)

[Chapter 1: Talking to Your Computer](#)

[Understanding Why You Want to Talk to Your Computer](#)

[Knowing that an Application Is a Form of Communication](#)

[Defining What an Application Is](#)

[Understanding Why Python Is So Cool](#)

## **Chapter 2: Working with Google Colab**

[Defining Google Colab](#)

[Working with Notebooks](#)

[Performing Common Tasks](#)

[Using Hardware Acceleration](#)

[Executing the Code](#)

[Getting Help](#)

## **Chapter 3: Interacting with Python**

[Typing a Command](#)

[Getting Python's Help](#)

[Finding Out More about Functions and Objects](#)

[Playing the Part of Inspector](#)

## **Chapter 4: Writing Your First Application**

[Understanding Why IDEs Are Important](#)

[Creating the Application](#)

[Running the Application](#)

[Understanding the Use of Indentation](#)

[Adding Comments](#)

[Making Your Notebook Informative, Descriptive, and Pretty](#)

[Closing and Halting a Notepad](#)

## **Chapter 5: Performing Magic**

[Understanding the Concept of a Magic Command](#)

[What Kind of Magic Do You Want to Perform?](#)

[Learning the Magic Commands](#)

## **Part 2: Talking the Talk**

### **Chapter 6: Storing and Modifying Information**

[Storing Information](#)

[Defining the Essential Python Data Types](#)

[Working with Dates and Times](#)

### **Chapter 7: Managing Information**

[Controlling How Python Views Data](#)

[Working with Operators](#)

[Creating and Using Functions](#)

[Getting User Input](#)

## **Chapter 8: Making Decisions**

[Making Simple Decisions by Using the if Statement](#)

[Choosing Alternatives by Using the if...else Statement](#)

[Using Nested Decision Statements](#)

## **Chapter 9: Performing Repetitive Tasks**

[Processing Data Using the for Statement](#)

[Processing Data by Using the while Statement](#)

[Nesting Loop Statements](#)

## **Chapter 10: Dealing with Errors**

[Knowing Why Python Doesn't Understand You](#)

[Considering the Sources of Errors](#)

[Catching Exceptions](#)

[Raising Exceptions](#)

[Deciding to Say “Oops” in Your Own Way: Custom Exceptions](#)

[Using the finally Clause](#)

## **Part 3: Performing Common Tasks**

### **Chapter 11: Interacting with Packages**

[Creating Code Groupings](#)

[Importing Packages](#)

[Finding Packages](#)

[Downloading Packages from Other Sources](#)

[Viewing the Package Content](#)

[Viewing Package Documentation](#)

### **Chapter 12: Working with Strings**

[Understanding That Strings Are Different](#)

[Creating Stings with Special Characters](#)

[Selecting Individual Characters](#)



[Slicing and Dicing Strings](#)

[Locating a Value in a String](#)

[Using String Interpolation](#)

## **Chapter 13: Managing Lists**

[Organizing Information in an Application](#)

[Creating Lists](#)

[Accessing Lists](#)

[Looping through Lists](#)

[Modifying Lists](#)

[Searching Lists](#)

[Sorting Lists](#)

[Printing Lists](#)

[Working with the Counter Object](#)

## **Chapter 14: Collecting All Sorts of Data**

[Understanding Collections](#)

[Working with Tuples](#)

[Working with Dictionaries](#)

[Creating Stacks Using Lists](#)

[Working with queues](#)

[Working with deques](#)

## **Chapter 15: Creating and Using Classes**

[Considering the Parts of a Class](#)

[Creating and Using an External Class](#)

[Extending Classes to Make New Classes](#)

## **Part 4: Performing Advanced Tasks**

### **Chapter 16: Storing Data in Files**

[Understanding How Permanent Storage Works](#)

[Creating Content for Permanent Storage](#)

[Creating a File](#)

[Reading File Content](#)

[Updating File Content](#)

[Deleting a File](#)

## **Chapter 17: Sending an Email**

[Understanding What Happens When You Send Email](#)

## **Part 5: The Part of Tens**

## **Chapter 18: Ten Amazing Programming Resources**

[Working with the Python Documentation Online](#)

[Discovering Details Using a Tutorial](#)

[Performing Web Programming by Using Python](#)

[Locating Useful \(versus Useless\) Modules](#)

[Creating Applications Faster by Using an IDE](#)

[Checking Your Syntax with Greater Ease](#)

[Using XML to Your Advantage](#)

[Getting Past the Common Python Newbie Errors](#)

[Understanding Unicode](#)

[Making Your Python Application Fast](#)

## **Chapter 19: Ten Ways to Make a Living with Python**

[Working in QA](#)

[Becoming the IT Staff for a Smaller Organization](#)

[Performing Specialty Scripting for Applications](#)

[Administering a Network](#)

[Teaching Programming Skills](#)

[Helping People Decide on Location](#)

[Performing Data Mining](#)

[Interacting with Embedded Systems](#)

[Carrying Out Scientific Tasks](#)

[Performing Real-Time Analysis of Data](#)

## **Chapter 20: Ten Tools That Enhance Your Python Experience**

[Tracking Bugs with Roundup Issue Tracker](#)

[Creating a Virtual Environment by Using VirtualEnv](#)

[Installing Your Application by Using PyInstaller](#)

[Building Developer Documentation by Using pdoc](#)  
[Developing Application Code by Using Komodo Edit](#)  
[Debugging Your Application by Using pydbgr](#)  
[Entering an Interactive Environment by Using IPython](#)  
[Testing Python Applications by Using PyUnit](#)  
[Tidying Your Code by Using Isort](#)  
[Providing Version Control by Using Mercurial](#)

## **Chapter 21: Ten (Plus) Libraries You Need to Know About**

[Developing a Secure Environment by Using CryptLib](#)  
[Interacting with Databases by Using SQLAlchemy](#)  
[Seeing the World by Using Google Maps](#)  
[Adding a Graphical User Interface by Using TkInter](#)  
[Providing a Nice Tabular Data Presentation by Using PrettyTable](#)  
[Enhancing Your Application with Sound by Using PyAudio](#)  
[Manipulating Images by Using PyQtGraph](#)  
[Locating Your Information by Using Whoosh](#)  
[Creating an Interoperable Java Environment by Using JPyype](#)  
[Accessing Local Network Resources by Using Twisted Matrix](#)  
[Accessing Internet Resources by Using Libraries](#)

**[Index](#)**

**[About the Author](#)**

**[Advertisement Page](#)**

**[Connect with Dummies](#)**

**[End User License Agreement](#)**

# List of Tables

**Chapter 1**

[TABLE 1-1 Large Organizations That Use Python](#)

## Chapter 5

[TABLE 5-1 Common IPython Line Magic Commands](#)

[TABLE 5-2 Common IPython Cell Magic Commands](#)

## Chapter 7

[TABLE 7-1 Python Unary Operators](#)

[TABLE 7-2 Python Arithmetic Operators](#)

[TABLE 7-3 Python Relational Operators](#)

[TABLE 7-4 Python Logical Operators](#)

[TABLE 7-5 Python Bitwise Operators](#)

[TABLE 7-6 Python Assignment Operators](#)

[TABLE 7-7 Python Membership Operators](#)

[TABLE 7-8 Python Identity Operators](#)

[TABLE 7-9 Python Operator Precedence](#)

## Chapter 12

[TABLE 12-1 Python Escape Sequences](#)

# List of Illustrations

## Chapter 1

[FIGURE 1-1: Communication with your computer may be invisible unless you really...](#)

## Chapter 2

[FIGURE 2-1: Using Colab commands makes configuring your Notebook easy.](#)

[FIGURE 2-2: The Settings dialog box helps you configure the Colab IDE.](#)

[FIGURE 2-3: Customize shortcut keys for speed of access to commands.](#)

[FIGURE 2-4: Colab lets you compare two files to see how they differ.](#)

[FIGURE 2-5: Create a new Python 3 Notebook.](#)

[FIGURE 2-6: Use this dialog box to open existing notebooks.](#)

[FIGURE 2-7: When using GitHub, you must provide the location of the source code...](#)

[FIGURE 2-8: Using GitHub means storing your data in a repository.](#)

[FIGURE 2-9: Colab code cells contain a few extras not found in Notebook.](#)

[FIGURE 2-10: Use the GUI to make formatting your text easier.](#)

## Chapter 3

[FIGURE 3-1: Issuing commands tells Python what to tell the computer to do.](#)

[FIGURE 3-2: Some commands require further input from you to complete.](#)

[FIGURE 3-3: You ask Python about other commands in help mode.](#)

[FIGURE 3-4: The topics help topic provides you with a starting point for your P...](#)

[FIGURE 3-5: You must use uppercase when requesting topic information.](#)

[FIGURE 3-6: Request command help information by typing the command using whatev...](#)

[FIGURE 3-7: You can ask for help on the help you receive.](#)

[FIGURE 3-8: Exit help mode by pressing Enter without typing anything.](#)

[FIGURE 3-9: Python lets you obtain help whenever you need it without leaving th...](#)

[FIGURE 3-10: You can browse at the Python prompt if you really want to.](#)

## Chapter 4

[FIGURE 4-1: Adding headings helps you separate and document your code.](#)

[FIGURE 4-2: Using heading levels provides emphasis for cell content.](#)

[FIGURE 4-3: A scratch cell provides a place to experiment and run shell command...](#)

[FIGURE 4-4: The Add New Form Field dialog box allows you to add GUI inputs to y...](#)

[FIGURE 4-5: Adding a form field to your notebook displays a combination of code...](#)

[FIGURE 4-6: Design code to interact with the form fields you create.](#)

[FIGURE 4-7: Viewing the executed code history can tell you a lot about your cod...](#)

[FIGURE 4-8: The Edit window automatically indents some types of text.](#)

[FIGURE 4-9: Use concatenation to make multiple lines of text appear on a single...](#)

[FIGURE 4-10: Multiline comments do work, but they also provide output.](#)

[FIGURE 4-11: Use docstrings for strings that require multiple lines.](#)

[FIGURE 4-12: Use comments to keep code from executing.](#)

[FIGURE 4-13: The text cell toolbar contains a variety of interesting options.](#)

[FIGURE 4-14: Use indents to create nested lists for your documentation.](#)

[FIGURE 4-15: The Table of Contents feature provides an outline of your notebook...](#)

## **Chapter 5**

[FIGURE 5-1: The `%lsmagic` command displays a list of magic commands for you.](#)

[FIGURE 5-2: The `%quickref` function displays more detailed magic command informa...](#)

[FIGURE 5-3: The `%quickref` function may not tell you everything you need to know...](#)

## **Chapter 6**

[FIGURE 6-1: Use the assignment operator to place information into a variable.](#)

[FIGURE 6-2: Integers have many interesting features, including the capability t...](#)

[FIGURE 6-3: Floating-point values provide multiple assignment techniques.](#)

[FIGURE 6-4: Converting a string to a number is easy by using the `int\(\)` and `fl...`](#)

[FIGURE 6-5: You can convert numbers to strings as well.](#)

[FIGURE 6-6: Get the current date and time by using the `now\(\)` function.](#)

[FIGURE 6-7: Make the date and time more readable by using the `str\(\)` function.](#)

## Chapter 7

[FIGURE 7-1: Define the name of your function.](#)

[FIGURE 7-2: Whenever you type the function's name, you get the output the funct...](#)

[FIGURE 7-3: You must supply an argument or you get an error message.](#)

[FIGURE 7-4: Supply default arguments when possible to make your functions easie...](#)

[FIGURE 7-5: Variable argument functions can make your applications more flexibl...](#)

[FIGURE 7-6: Use your functions to perform a wide variety of tasks.](#)

[FIGURE 7-7: Provide a username and see a greeting as output.](#)

[FIGURE 7-8: Data conversion changes the input type to whatever you need, but co...](#)

## Chapter 8

[FIGURE 8-1: Simple if statements can help your application know what to do in c...](#)

[FIGURE 8-2: A code block can contain multiple lines of code — one for each task...](#)

[FIGURE 8-3: The application verifies the value is in the right range and output...](#)

[FIGURE 8-4: Typing the wrong type of information results in an error message.](#)

[FIGURE 8-5: Providing feedback for incorrect input is always a good idea.](#)

[FIGURE 8-6: Menus let you choose one option from a list of options.](#)

[FIGURE 8-7: Every application you create should include some means of detecting...](#)

[FIGURE 8-8: Adding multiple levels lets you perform tasks with greater complexi...](#)

## Chapter 9

[FIGURE 9-1: Use the for loop to process the characters in a string one at a tim...](#)

[FIGURE 9-2: Long strings are truncated to ensure that they remain a certain siz...](#)

[FIGURE 9-3: Use the `continue` clause to avoid processing specific elements.](#)

[FIGURE 9-4: The `else` clause enables you to perform tasks based on an empty sequ...](#)

[FIGURE 9-5: The simple `while` loop displays a sequence of numbers.](#)

[FIGURE 9-6: The multiplication table is pleasing to the eye thanks to its forma...](#)

## Chapter 10

[FIGURE 10-1: Typing the wrong input type generates an error message instead of ...](#)

[FIGURE 10-2: Exception handling doesn't ensure that the value is in the correct...](#)

[FIGURE 10-3: The exception handlers are in the wrong order.](#)

[FIGURE 10-4: Generic exception handling traps the `KeyboardInterrupt` exception.](#)

[FIGURE 10-5: Attempting to open a nonexistent file never works.](#)

[FIGURE 10-6: The order in which Python processes exceptions is important.](#)

[FIGURE 10-7: Using a loop means that the application can recover from the error...](#)

[FIGURE 10-8: You can add error information to any exception.](#)

[FIGURE 10-9: Use the `finally` clause to ensure that specific actions take place ...](#)

[FIGURE 10-10: Be sure to remember that the `finally` clause always executes.](#)

## Chapter 11

[FIGURE 11-1: Make sure you place a copy of the package in your `BPP4D3E` folder.](#)

[FIGURE 11-2: A directory listing shows that Python imports both functions from ...](#)

[FIGURE 11-3: Removing a package from the environment requires two steps.](#)

[FIGURE 11-4: Use the `dir\(\)` function to obtain information about the specific at...](#)

[FIGURE 11-5: The `sys.path` attribute contains a listing of the individual paths ...](#)



[FIGURE 11-6: You see a lot of information during the update process.](#)

[FIGURE 11-7: PyDoc makes it easy to get instructions on its use directly within...](#)

[FIGURE 11-8: PyDoc will tell you all about the `print\(\)` function if you ask.](#)

[FIGURE 11-9: Keyword searches help you locate items that you may not know how t...](#)

## Chapter 12

[FIGURE 12-1: Strings consist of individual characters that are linked together.](#)

[FIGURE 12-2: Use special characters as needed to present special information or...](#)

[FIGURE 12-3: You can select individual pieces of a string.](#)

[FIGURE 12-4: Using functions makes string manipulation a lot more flexible.](#)

[FIGURE 12-5: Using string search features correctly is essential.](#)

[FIGURE 12-6: The `%` operator is good for simple output needs.](#)

[FIGURE 12-7: Use formatting to present data in precisely the form you want.](#)

[FIGURE 12-8: The f-string approach is simpler than other methods, yet quite fle...](#)

[FIGURE 12-9: String templates enable you to create reusable content with ease.](#)

## Chapter 13

[FIGURE 13-1: Python displays the content of `List1`.](#)

[FIGURE 13-2: Knowing the correct indexing operation to perform is important.](#)

[FIGURE 13-3: A loop makes it easy to obtain a copy of each item and process it ...](#)

[FIGURE 13-4: Python provides a wide range of actions you can perform using `list`...](#)

[FIGURE 13-5: Combining loops with conditional code allows multiple selection en...](#)

[FIGURE 13-6: Sorting a list is as easy as calling the `sort\(\)` function.](#)

[FIGURE 13-7: Python provides a lot of different ways to print lists.](#)

[FIGURE 13-8: The Counter is helpful in obtaining statistics about longer lists.](#)

## Chapter 14

[FIGURE 14-1: Tuples use parentheses, not square brackets.](#)

[FIGURE 14-2: Use indexes to gain access to the individual tuple members.](#)

[FIGURE 14-3: A dictionary places entries in sorted order.](#)

[FIGURE 14-4: Dictionaries work much like lists do, but with a little extra func...](#)

[FIGURE 14-5: Nested dictionaries follow the same rules as regular dictionaries,...](#)

[FIGURE 14-6: Recursion makes working with nested dictionaries easier.](#)

[FIGURE 14-7: After displaying your selection, the application displays the menu...](#)

## Chapter 15

[FIGURE 15-1: The class name is also correct, so you know that this instance is ...](#)

[FIGURE 15-2: Python provides help for each of the attributes it adds to your cl...](#)

[FIGURE 15-3: The class method outputs a simple message.](#)

[FIGURE 15-4: Supplying the constructor with a name provides a customized output...](#)

[FIGURE 15-5: You can change the value of Greeting.](#)

[FIGURE 15-6: The change to Greeting carries over to the instance of the class.](#)

[FIGURE 15-7: The output is simply the sum of two numbers.](#)

[FIGURE 15-8: The code can process any number of entries in the list.](#)

[FIGURE 15-9: The result of adding two MyClass objects is a third object of the ...](#)

[FIGURE 15-10: The output shows that the class is fully functional.](#)

[FIGURE 15-11: Sally has a birthday and then says a few words.](#)

## Chapter 16

[FIGURE 16-1: The example presents how the data might look in CSV format.](#)

[FIGURE 16-2: Colab displays the file content using the first row as a header.](#)

[FIGURE 16-3: Jupyter Notebook makes no assumption about the header row.](#)

[FIGURE 16-4: The application input after it has been processed.](#)

## **Chapter 17**

[FIGURE 17-1: Python adds some additional information required to make your mess...](#)

[FIGURE 17-2: You see the message transmitted by the SMTP server at the Anaconda...](#)

# Introduction

---

Python is an example of a language that does everything right within the domain of things that it's designed to do. This isn't just me saying it, either: Programmers have voted by using Python enough that it's now the first-ranked language in the world (see <https://www.tiobe.com/tiobe-index/> for details). The amazing thing about Python is that you really can write an application on one platform and use it on every other platform that you need to support. In contrast to other programming languages that promised to provide platform independence, Python really does make that independence possible. In this case, the promise is as good as the result you get.

Python emphasizes code readability and a concise syntax that lets you write applications using fewer lines of code than other programming languages require. You can also use a coding style that meets your needs, given that Python supports the functional, imperative, object-oriented, and procedural coding styles (see [Chapter 3](#) for details). In addition, because of the way Python works, you find it used in all sorts of fields that are filled with nonprogrammers. *Beginning Programming with Python For Dummies*, 3rd Edition is designed to help everyone, including nonprogrammers, get up and running with Python quickly.

Some people view Python as a scripted language, but it really is so much more. ([Chapter 19](#) gives you just an inkling of the occupations that rely on Python to make things work.) However, Python does lend itself to educational and other uses for which other programming languages can fall short. In fact, this book uses both Google Colab and Jupyter Notebook for examples, which

rely on the highly readable literate programming paradigm advanced by Stanford computer scientist Donald Knuth (see [Chapter 4](#) for details). Your examples end up looking like highly readable reports that almost anyone can understand with ease.

## ***About This Book***

*Beginning Programming with Python For Dummies*, 3rd Edition is all about getting up and running with Python quickly. You want to learn the language fast so that you can become productive in using it to perform your real job, which could be anything. With the goal in mind of making things simple in every environment, this book emphasizes a code anywhere approach. If you want to code on your smart phone (not really recommended unless you like to squint a lot), you can do so as long as your smart phone has a browser that can access Google Colab. Likewise, coding while watching a TV equipped with a keyboard is possible, but not necessarily recommended because of the distractions involved. Besides, trying to write code that you can see only in that small square in the corner of the screen would be very tough. Highly recommended is your desktop, laptop, or tablet.

Unlike most books on the topic, this one starts you right at the beginning by showing you what makes Python different from other languages and how it can help you perform useful work in a job other than programming. As a result, you gain an understanding of what you need to do from the start, using hands-on examples and spending a good deal of time performing actually useful tasks. By the time you finish working through the examples in this book, you'll be writing simple programs and performing tasks such as sending an email using Python. No, you

won't be an expert, but you will be able to use Python to meet specific needs in the job environment. To make absorbing the concepts even easier, this book uses the following conventions:

- » Text that you're meant to type just as it appears in the book is **bold**. The exception is when you're working through a step list: Because each step is bold, the text to type is not bold.
- » When you see words in *italics* as part of a typing sequence, you need to replace that value with something that works for you. For example, if you see "Type **Your Name** and press Enter," you need to replace *Your Name* with your actual name.
- » Web addresses and programming code appear in monofont. If you're reading a digital version of this book on a device connected to the Internet, note that you can click the web address to visit that website, like this: [www.dummies.com](http://www.dummies.com).
- » When you need to type command sequences, you see them separated by a special arrow, like this: File ⇒ New File. In this case, you go to the File menu first and then select the New File entry on that menu. The result is that you see a new file created.

## ***Foolish Assumptions***

You might find it difficult to believe that I've assumed anything about you — after all, I haven't even met you yet! Although most assumptions are indeed foolish, I made these assumptions to provide a starting point for the book.

Familiarity with the platform you want to use is important because the book doesn't provide any

guidance in this regard. To provide you with maximum information about Python, this book doesn't discuss any platform-specific issues. You really do need to know how to install applications (when working with a desktop system), use applications, work with your browser, and generally work with your chosen platform before you begin working with this book.

This book also assumes that you can locate information on the Internet. Sprinkled throughout are numerous references to online material that will enhance your learning experience. However, these added sources are useful only if you actually find and use them.

## *Icons Used in This Book*

As you read this book, you see icons in the margins that indicate material of interest (or not, as the case may be). This section briefly describes each icon in this book.



**TIP**

Tips are nice because they help you save time or perform some task without a lot of extra work. The tips in this book are time-saving techniques or pointers to resources that you should try in order to get the maximum benefit from Python.



**WARNING**

I don't want to sound like an angry parent or some kind of maniac, but you should avoid doing anything marked with a Warning icon. Otherwise, you could find that your program only serves to confuse users, who will then refuse to work with it.



TECHNICAL  
STUFF

Whenever you see this icon, think advanced tip or technique. You might find these tidbits of useful information just too boring for words, or they could contain the solution you need to get a program running. Skip these bits of information whenever you like.



REMEMBER

If you don't get anything else out of a particular chapter or section, remember the material marked by this icon. This text usually contains an essential process or a bit of information that you must know to write Python programs successfully.

## ***Beyond the Book***

This book isn't the end of your Python programming experience — it's really just the beginning. I provide online content to make this book more flexible and better able to meet your needs. That way, as I receive email from you, I can do things like address questions and tell you how updates to either Python or its associated libraries affect book content. In fact, you gain access to all these cool additions:

- » **Cheat sheet:** You remember using crib notes in school to make a better mark on a test, don't you? You do? Well, a cheat sheet is sort of like that. It provides you with some special notes about tasks that you can do with Python that not every other developer knows. You can find the cheat sheet for this book by going to [www.dummies.com](http://www.dummies.com) and searching for *Beginning*



*Programming with Python For Dummies*, 3rd Edition Cheat Sheet. It contains really neat information like how to perform magic when using Python.

- » **Updates:** Sometimes changes happen. For example, I might not have seen an upcoming change when I looked into my crystal ball during the writing of this book. In the past, that simply meant the book would become outdated and less useful, but you can now find updates to the book by going to [www.dummies.com](http://www.dummies.com) and searching for this book's title.

In addition to these updates, check out the blog posts with answers to reader questions and demonstrations of useful book-related techniques at <http://blog.johnmuelเลอร์books.com/>.

- » **Companion files:** Hey! Who really wants to type all the code in the book? Most readers would prefer to spend their time actually working through coding examples, rather than typing. Fortunately for you, the source code is available for download, so all you need to do is read the book to learn Python coding techniques. Each of the book examples even tells you precisely which example project to use. You can find these files by going to [www.dummies.com](http://www.dummies.com) and searching for this book's title. You can also find the downloadable source on my website at <http://www.johnmuelเลอร์books.com/source-code/>; just click the Download button for *Beginning Programming with Python For Dummies*, 3rd Edition. Be sure to unzip the file using the instructions at <https://support.microsoft.com/en-us/windows/zip-and-unzip-files-8d28fa72-f2f9-712f-67df-f80cf89fd4e5> before attempting to use the source code, even if you can see it in Windows Explorer.

# *Where to Go from Here*

It's time to start your Programming with Python adventure! If you're a complete programming novice, you should start with [Chapter 1](#) and progress through the book at a pace that allows you to absorb as much of the material as possible.

If you're a novice who's in an absolute rush to get going with Python as quickly as possible, you could skip to [Chapter 2](#) with the understanding that you may find some topics a bit confusing later. Skipping to [Chapter 3](#) is possible if you want to start working with Python immediately and have access to Google Colab or a Jupyter Notebook installation.

Readers who have some exposure to Python can save time by moving directly to [Chapter 4](#). This chapter gets you started working with notebooks so that you have a better idea of how to work with Google Colab or Jupyter Notebook for the examples in the remainder of the book. Make sure you also read through [Chapter 5](#), which tells you how to perform magic in notebooks.

Assuming that you already have access to either Google Colab or Jupyter Notebook and know how to use your IDE of choice, you can move directly to [Chapter 6](#). You can always go back to earlier chapters as necessary when you have questions. However, it's important that you understand how each example works before moving to the next one. Every example has important lessons for you, and you could miss vital content if you start skipping too much information.

**Part 1**  
**Getting Started with  
Python**

## **IN THIS PART ...**

Defining the association between Python and applications

Using Google Colab to work with Python

Performing essential tasks using Python

Creating your first application

Performing feats of magic

# Chapter 1

## Talking to Your Computer

---

### IN THIS CHAPTER

- » Talking to your computer
  - » Creating programs to talk to your computer
  - » Understanding programs and their creation
  - » Considering why you want to use Python
- 

Having a conversation with your computer might sound like the script of a science fiction movie. After all, the members of the *Enterprise* on *Star Trek* regularly talked with their computer. In fact, the computer often talked back. However, with the rise of Apple's Siri (<https://www.apple.com/siri/>), Amazon's Echo (<https://www.amazon.com/dp/B07XKF5RM3/>) and other interactive software (<https://windowsreport.com/talking-pc-software/>), perhaps you really don't find a conversation so unbelievable.



**REMEMBER** Asking the computer for information is one thing, but providing it with instructions is quite another. This chapter considers why you want to instruct your computer about anything and what benefit you gain from it. You also discover the need for a special language when performing this kind of communication and why you want to use Python to accomplish it. However, the main thing to get out of this chapter is that programming is simply a kind of communication that is akin to other forms of communication you already have with your computer.

## ***Understanding Why You Want to Talk to Your Computer***

Talking to a machine may seem quite odd at first (then again, people do talk to cats, dogs, cars, toasters, and other odd assorted things), but it's necessary because a computer can't read your mind — yet. Mind-reading computers are getting closer, as described in the article at <https://www.psychnewsdaily.com/this-computer-can-read-your-mind-and-render-your-thoughts-as-pictures/>. Even if the computer did read your mind, it would still be communicating with you. Nothing can occur without an exchange of information between the machine and you. Activities such as

- » Reading your email
- » Writing about your vacation