GLOBAL EDITION

Starting Out with Python®

FOURTH EDITION

Tony Gaddis

Pearson

Digital Resources for Students

Your new textbook provides 12-month access to digital resources that may include VideoNotes (step-by-step video tutorials on programming concepts), source code, web chapters, quizzes, and more. Refer to the preface in the textbook for a detailed list of resources.

Follow the instructions below to register for the Companion Website for Tony Gaddis' *Starting Out with Python*, Fourth Edition, Global Edition.

1. Go to www.pearsonglobaleditions.com/gaddis
2. Enter the title of your textbook or browse by author name.
3. Click Companion Website.
4. Click Register and follow the on-screen instructions to create a login name and password.

ISSGPY-PRANK-CURRY-PIQUE-METIS-ROUSE

Use the login name and password you created during registration to start using the online resources that accompany your textbook.

IMPORTANT:

This prepaid subscription does not include access to MyProgrammingLab, which is available at www.myprogramminglab.com for purchase.

This subscription is valid for 12 months upon activation and is not transferable.

For technical support go to https://support.pearson.com/getsupport

This page intentionally left blank

STARTING OUT WITH

PYTHON®

FOURTH EDITION
GLOBAL EDITION

This page intentionally left blank

STARTING OUT WITH

PYTHON®

FOURTH EDITION
GLOBAL EDITION

Tony Gaddis
Haywood Community College

Pearson

330 Hudson Street, New York, NY 10013

Senior Vice President Courseware Portfolio Management:	Marcia J. Horton
Director, Portfolio Management: Engineering, Computer Science & Global Editions:	Julian Partridge
Portfolio Manager:	Matt Goldstein
Acquisitions Editor, Global Edition:	Sourabh Maheshwari
Portfolio Management Assistant:	Kristy Alaura
Field Marketing Manager:	Demetrius Hall
Product Marketing Manager:	Yvonne Vannatta
Managing Producer, ECS and Math:	Scott Disanno
Content Producer:	Sandra L. Rodriguez
Project Editors, Global Edition:	Tanima Ghosh and Neelakantan K.K.
Senior Manufacturing Controller, Global Edition:	Jerry Kataria
Manager, Media Production, Global Edition:	Vikram Kumar
Cover Designer:	Lumina Datamatics
Cover Photo:	Africa Studio/Shutterstock

Credits and acknowledgments borrowed from other sources and reproduced, with permission, appear on the Credits page in the endmatter of this textbook.

Pearson Education Limited

KAO Two
KAO Park
Harlow
CM17 9NA
United Kingdom

and Associated Companies throughout the world

Visit us on the World Wide Web at: www.pearsonglobaleditions.com

British Library Cataloguing-in-Publication Data

A catalogue record for this book is available from the British Library

10 9 8 7 6 5 4 3 2 1

ISBN 10: 1-292-22575-0
ISBN 13: 978-1-292-22575-3

Typeset by iEnergizer Aptara®, Ltd.
Printed and bound in Malaysia

Contents at a Glance

This page intentionally left blank

Contents

LOCATION OF VIDEONOTES IN THE TEXT

This page intentionally left blank

Preface

Welcome to *Starting Out with Python*, Fourth Edition. This book uses the Python language to teach programming concepts and problem-solving skills, without assuming any previous programming experience. With easy-to-understand examples, pseudocode, flowcharts, and other tools, the student learns how to design the logic of programs and then implement those programs using Python. This book is ideal for an introductory programming course or a programming logic and design course using Python as the language.

As with all the books in the *Starting Out with* series, the hallmark of this text is its clear, friendly, and easy-to-understand writing. In addition, it is rich in example programs that are concise and practical. The programs in this book include short examples that highlight specific programming topics, as well as more involved examples that focus on problem solving. Each chapter provides one or more case studies that provide step-by-step analysis of a specific problem and shows the student how to solve it.

Control Structures First, Then Classes

Python is a fully object-oriented programming language, but students do not have to understand object-oriented concepts to start programming in Python. This text first introduces the student to the fundamentals of data storage, input and output, control structures, functions, sequences and lists, file I/O, and objects that are created from standard library classes. Then the student learns to write classes, explores the topics of inheritance and polymorphism, and learns to write recursive functions. Finally, the student learns to develop simple event-driven GUI applications.

Changes in the Fourth Edition

This book's clear writing style remains the same as in the previous edition. However, many additions and improvements have been made, which are summarized here:

- New sections on the Python Turtle Graphics library have been added to Chapters 2 through 5. The Turtle Graphics library, which is a standard part of Python, is a fun and motivating way to introduce programming concepts to students who have never written code before. The library allows the student to write Python statements that draw graphics by moving a cursor on a canvas. The new sections that have been added to this edition are:
  - Chapter 2: Introduction to Turtle Graphics
  - Chapter 3: Determining the State of the Turtle
  - Chapter 4: Using Loops to Draw Designs
  - Chapter 5: Modularizing Turtle Graphics Code with Functions

The new Turtle Graphics sections are designed with flexibility in mind. They can be assigned as optional material, incorporated into your existing syllabus, or skipped altogether.

- Chapter 2 has a new section on named constants. Although Python does not support true constants, you can create variable names that symbolize values that should not change as the program executes. This section teaches the student to avoid the use of "magic numbers," and to create symbolic names that make his or her code more self-documenting and easier to maintain.
- Chapter 7 has a new section on using the `matplotlib` package to plot charts and graphs from lists. The new section describes how to install the `matplotlib` package, and use it to plot line graphs, bar charts, and pie charts.
- Chapter 13 has a new section on creating graphics in a GUI application with the `Canvas` widget. The new section describes how to use the `Canvas` widget to draw lines, rectangles, ovals, arcs, polygons, and text.
- Several new, more challenging, programming problems have been added throughout the book.
- Appendix E is a new appendix that discusses the various forms of the `import` statement.
- Appendix F is a new appendix that discusses installing third-party modules with the `pip` utility.

Brief Overview of Each Chapter

Chapter 1: Introduction to Computers and Programming

This chapter begins by giving a very concrete and easy-to-understand explanation of how computers work, how data is stored and manipulated, and why we write programs in high-level languages. An introduction to Python, interactive mode, script mode, and the IDLE environment are also given.

Chapter 2: Input, Processing, and Output

This chapter introduces the program development cycle, variables, data types, and simple programs that are written as sequence structures. The student learns to write simple programs that read input from the keyboard, perform mathematical operations, and produce screen output. Pseudocode and flowcharts are also introduced as tools for designing programs. The chapter also includes an optional introduction to the turtle graphics library.

Chapter 3: Decision Structures and Boolean Logic

In this chapter, the student learns about relational operators and Boolean expressions and is shown how to control the flow of a program with decision structures. The `if`, `if-else`, and `if-elif-else` statements are covered. Nested decision structures and logical operators are discussed as well. The chapter also includes an optional turtle graphics section, with a discussion of how to use decision structures to test the state of the turtle.

Chapter 4: Repetition Structures

This chapter shows the student how to create repetition structures using the `while` loop and `for` loop. Counters, accumulators, running totals, and sentinels are discussed, as well as

techniques for writing input validation loops. The chapter also includes an optional section on using loops to draw designs with the turtle graphics library.

Chapter 5: Functions

In this chapter, the student first learns how to write and call void functions. The chapter shows the benefits of using functions to modularize programs and discusses the top-down design approach. Then, the student learns to pass arguments to functions. Common library functions, such as those for generating random numbers, are discussed. After learning how to call library functions and use their return value, the student learns to define and call his or her own functions. Then the student learns how to use modules to organize functions. An optional section includes a discussion of modularizing turtle graphics code with functions.

Chapter 6: Files and Exceptions

This chapter introduces sequential file input and output. The student learns to read and write large sets of data and store data as fields and records. The chapter concludes by discussing exceptions and shows the student how to write exception-handling code.

Chapter 7: Lists and Tuples

This chapter introduces the student to the concept of a sequence in Python and explores the use of two common Python sequences: lists and tuples. The student learns to use lists for arraylike operations, such as storing objects in a list, iterating over a list, searching for items in a list, and calculating the sum and average of items in a list. The chapter discusses slicing and many of the list methods. One- and two-dimensional lists are covered. The chapter also includes a discussion of the `matplotlib` package, and how to use it to plot charts and graphs from lists.

Chapter 8: More About Strings

In this chapter, the student learns to process strings at a detailed level. String slicing and algorithms that step through the individual characters in a string are discussed, and several built-in functions and string methods for character and text processing are introduced.

Chapter 9: Dictionaries and Sets

This chapter introduces the dictionary and set data structures. The student learns to store data as key-value pairs in dictionaries, search for values, change existing values, add new key-value pairs, and delete key-value pairs. The student learns to store values as unique elements in sets and perform common set operations such as union, intersection, difference, and symmetric difference. The chapter concludes with a discussion of object serialization and introduces the student to the Python `pickle` module.

Chapter 10: Classes and Object-Oriented Programming

This chapter compares procedural and object-oriented programming practices. It covers the fundamental concepts of classes and objects. Attributes, methods, encapsulation and data hiding, `__init__` functions (which are similar to constructors), accessors, and mutators are discussed. The student learns how to model classes with UML and how to find the classes in a particular problem.

Chapter 11: Inheritance

The study of classes continues in this chapter with the subjects of inheritance and polymorphism. The topics covered include superclasses, subclasses, how `__init__` functions work in inheritance, method overriding, and polymorphism.

Chapter 12: Recursion

This chapter discusses recursion and its use in problem solving. A visual trace of recursive calls is provided, and recursive applications are discussed. Recursive algorithms for many tasks are presented, such as finding factorials, finding a greatest common denominator (GCD), and summing a range of values in a list, and the classic Towers of Hanoi example are presented.

Chapter 13: GUI Programming

This chapter discusses the basic aspects of designing a GUI application using the `tkinter` module in Python. Fundamental widgets, such as labels, buttons, entry fields, radio buttons, check buttons, and dialog boxes, are covered. The student also learns how events work in a GUI application and how to write callback functions to handle events. The chapter includes a discussion of the `Canvas` widget, and how to use it to draw lines, rectangles, ovals, arcs, polygons, and text.

Appendix A: Installing Python

This appendix explains how to download and install the Python 3 interpreter.

Appendix B: Introduction to IDLE

This appendix gives an overview of the IDLE integrated development environment that comes with Python.

Appendix C: The ASCII Character Set

As a reference, this appendix lists the ASCII character set.

Appendix D: Predefined Named Colors

This appendix lists the predefined color names that can be used with the turtle graphics library, `matplotlib` and `tkinter`.

Appendix E: More About the `import` Statement

This appendix discusses various ways to use the `import` statement. For example, you can use the `import` statement to import a module, a class, a function, or to assign an alias to a module.

Appendix F: Installing Modules with the `pip` Utility

This appendix discusses how to use the `pip` utility to install third-party modules from the Python Package Index, or PyPI.

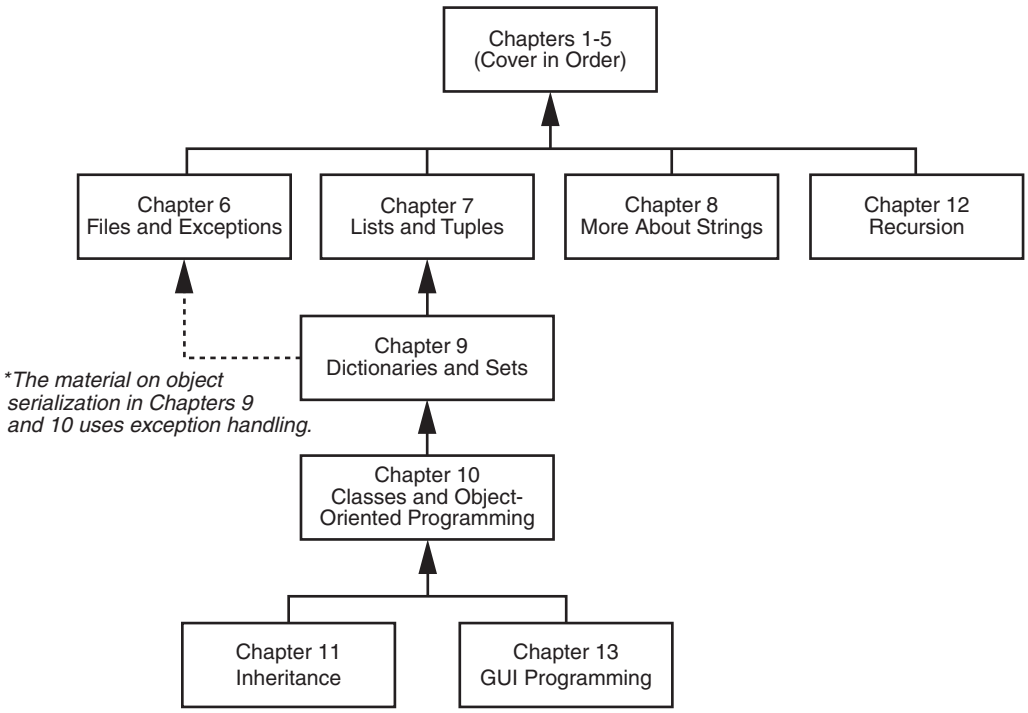
Appendix G: Answers to Checkpoints

This appendix gives the answers to the Checkpoint questions that appear throughout the text.

Organization of the Text

The text teaches programming in a step-by-step manner. Each chapter covers a major set of topics and builds knowledge as students progress through the book. Although the chapters can be easily taught in their existing sequence, you do have some flexibility in the order that you wish to cover them. Figure P-1 shows chapter dependencies. Each box represents a chapter or a group of chapters. An arrow points from a chapter to the chapter that must be covered before it.

Figure P-1 Chapter dependencies



Features of the Text

Concept Each major section of the text starts with a concept statement.

Statements This statement concisely summarizes the main point of the section.

Example Programs Each chapter has an abundant number of complete and partial example programs, each designed to highlight the current topic.

In the Spotlight Case Studies Each chapter has one or more *In the Spotlight* case studies that provide detailed, step-by-step analysis of problems and show the student how to solve them.

VideoNotes Online videos developed specifically for this book are available for viewing at www.pearsonglobaleditions.com/gaddis. Icons appear throughout the text alerting the student to videos about specific topics.

Notes Notes appear at several places throughout the text. They are short explanations of interesting or often misunderstood points relevant to the topic at hand.

Tips Tips advise the student on the best techniques for approaching different programming problems.

Warnings Warnings caution students about programming techniques or practices that can lead to malfunctioning programs or lost data.

Checkpoints Checkpoints are questions placed at intervals throughout each chapter. They are designed to query the student's knowledge quickly after learning a new topic.

Review Questions Each chapter presents a thorough and diverse set of review questions and exercises. They include Multiple Choice, True/False, Algorithm Workbench, and Short Answer.

Programming Exercises Each chapter offers a pool of programming exercises designed to solidify the student's knowledge of the topics currently being studied.

Supplements

Student Online Resources

Many student resources are available for this book from the publisher. The following items are available at www.pearsonglobaleditions.com/gaddis

- The source code for each example program in the book
- Access to the book's companion VideoNotes

Instructor Resources

The following supplements are available to qualified instructors only:

- Answers to all of the Review Questions
- Solutions for the exercises
- PowerPoint presentation slides for each chapter
- Test bank

Visit the Pearson Instructor Resource Center (www.pearsonglobaleditions.com/gaddis) or contact your local Pearson campus representative for information on how to access them.

Acknowledgments

I would like to thank the following faculty reviewers for their insight, expertise, and thoughtful recommendations:

Sonya Dennis
Morehouse College

Diane Innes
Sandhills Community College

John Kinuthia
Nazareth College of Rochester

Frank Liu
Sam Houston State University

Haris Ribic
SUNY at Binghamton

Anita Sutton
Germanna Community College

Christopher Urban
SUNY Institute of Technology

Nanette Veilleux
Simmons College

Brent Wilson
George Fox University

Reviewers of Previous Editions

Paul Amer
University of Delaware

James Atlas
University of Delaware

James Carrier
Guilford Technical Community College

John Cavazos
University of Delaware

Desmond K. H. Chun
Chabot Community College

Barbara Goldner
North Seattle Community College

Paul Gruhn
Manchester Community College

Bob Husson
Craven Community College

Diane Innes
Sandhills Community College

Daniel Jinguji
North Seattle Community College

Gary Marrer
Glendale Community College

Keith Mehl
Chabot College

Shyamal Mitra
University of Texas at Austin

Vince Offenback
North Seattle Community College

Smiljana Petrovic
Iona College

Raymond Pettit
Abilene Christian University

Janet Renwick
University of Arkansas–Fort Smith

Ken Robol
Beaufort Community College

Eric Shaffer
University of Illinois at Urbana-Champaign

Tom Stokke
University of North Dakota

Ann Ford Tyson
Florida State University

Karen Ughetta
Virginia Western Community College

Linda F. Wilson
Texas Lutheran University

I would also like to thank my family and friends for their support in all of my projects. I am extremely fortunate to have Matt Goldstein as my editor, and Kristy Alaura as editorial assistant. Their guidance and encouragement make it a pleasure to write chapters and meet deadlines. I am also fortunate to have Demetrius Hall as my marketing manager. His hard work is truly inspiring, and he does a great job of getting this book out to the academic community. The production team, led by Sandra Rodriguez, worked tirelessly to make this book a reality. Thanks to you all!

Acknowledgments for the Global Edition

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

Contributor
Gregory Baatard, *Edith Cowan University*
Kenneth Eustace, *Charles Sturt University*

Reviewers
Lindsay Ward, *James Cook University*
Ajay Mittal, *University Institute of Engineering and Technology*
Khyat Sharma

About the Author

Tony Gaddis is the principal author of the *Starting Out with* series of textbooks. Tony has nearly two decades of experience teaching computer science courses at Haywood Community College. He is a highly acclaimed instructor who was previously selected as the North Carolina Community College "Teacher of the Year" and has received the Teaching Excellence award from the National Institute for Staff and Organizational Development. The *Starting Out with* series includes introductory books covering C++, Java™, Microsoft® Visual Basic®, Microsoft® C#®, Python®, Programming Logic and Design, Alice, and App Inventor, all published by Pearson.

BREAKTHROUGH
To improving results

Pearson MyLab® Programming

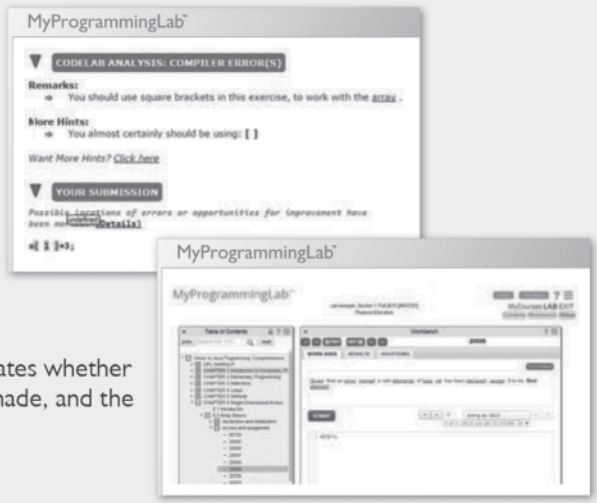
Through the power of practice and immediate personalized feedback, Pearson MyLab® Programming helps improve your students' performance.

PROGRAMMING PRACTICE

With Pearson MyLab® Programming, your students will gain first-hand programming experience in an interactive online environment.

IMMEDIATE, PERSONALIZED FEEDBACK

Pearson MyLab® Programming automatically detects errors in the logic and syntax of their code submission and offers targeted hints that enables students to figure out what went wrong and why.



GRADUATED COMPLEXITY

Pearson MyLab® Programming breaks down programming concepts into short, understandable sequences of exercises. Within each sequence the level and sophistication of the exercises increase gradually but steadily.

DYNAMIC ROSTER

Students' submissions are stored in a roster that indicates whether the submission is correct, how many attempts were made, and the actual code submissions from each attempt.

PEARSON eTEXT

The Pearson eText gives students access to their textbook anytime, anywhere.

STEP-BY-STEP VIDEONOTE TUTORIALS

These step-by-step video tutorials enhance the programming concepts presented in select Pearson textbooks.

For more information and titles available with **Pearson MyLab® Programming,** please visit **www.myprogramminglab.com**.

ALWAYS LEARNING

PEARSON

This page intentionally left blank

CHAPTER 1

Introduction to Computers and Programming

TOPICS

1.1 Introduction

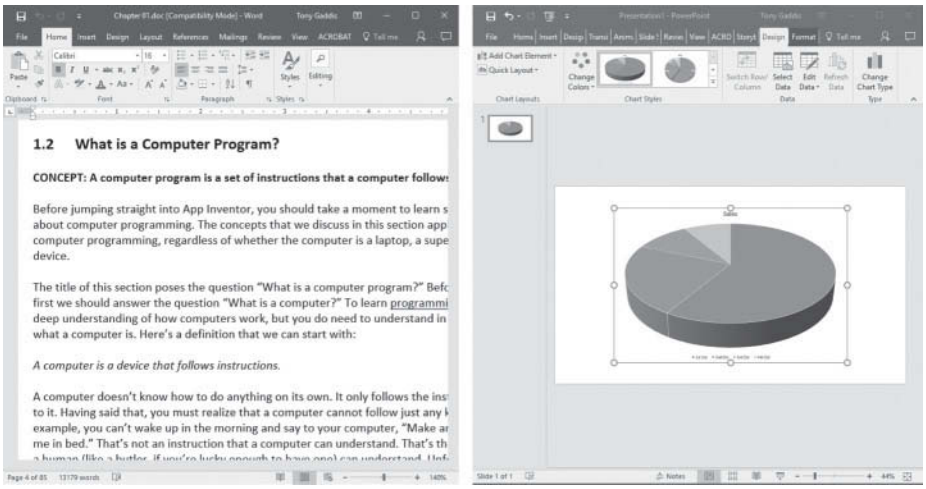
Think about some of the different ways that people use computers. In school, students use computers for tasks such as writing papers, searching for articles, sending email, and participating in online classes. At work, people use computers to analyze data, make presentations, conduct business transactions, communicate with customers and coworkers, control machines in manufacturing facilities, and do many other things. At home, people use computers for tasks such as paying bills, shopping online, communicating with friends and family, and playing games. And don't forget that cell phones, tablets, smart phones, car navigation systems, and many other devices are computers too. The uses of computers are almost limitless in our everyday lives.

Computers can perform such a wide variety of tasks because they can be programmed. This means that computers are not designed to do just one job, but to do any job that their programs tell them to do. A *program* is a set of instructions that a computer follows to perform a task. For example, Figure 1-1 shows screens using Microsoft Word and PowerPoint, two commonly used programs.

Programs are commonly referred to as *software*. Software is essential to a computer because it controls everything the computer does. All of the software that we use to make our computers useful is created by individuals working as programmers or software developers. A *programmer,* or *software developer,* is a person with the training and skills necessary to design, create, and test computer programs. Computer programming is an exciting and rewarding career. Today, you will find programmers' work used in business, medicine, government, law enforcement, agriculture, academics, entertainment, and many other fields.

This book introduces you to the fundamental concepts of computer programming using the Python language. The Python language is a good choice for beginners because it is easy to learn

Figure 1-1 A word processing program and an image editing program



and programs can be written quickly using it. Python is also a powerful language, popular with professional software developers. In fact, it has been reported that Python is used by Google, NASA, YouTube, various game companies, the New York Stock Exchange, and many others.

Before we begin exploring the concepts of programming, you need to understand a few basic things about computers and how they work. This chapter will build a solid foundation of knowledge that you will continually rely on as you study computer science. First, we will discuss the physical components of which computers are commonly made. Next, we will look at how computers store data and execute programs. Finally, you will get a quick introduction to the software that you will use to write Python programs.

1.2 Hardware and Software

CONCEPT: **The physical devices of which a computer is made are referred to as the computer's hardware. The programs that run on a computer are referred to as software.**

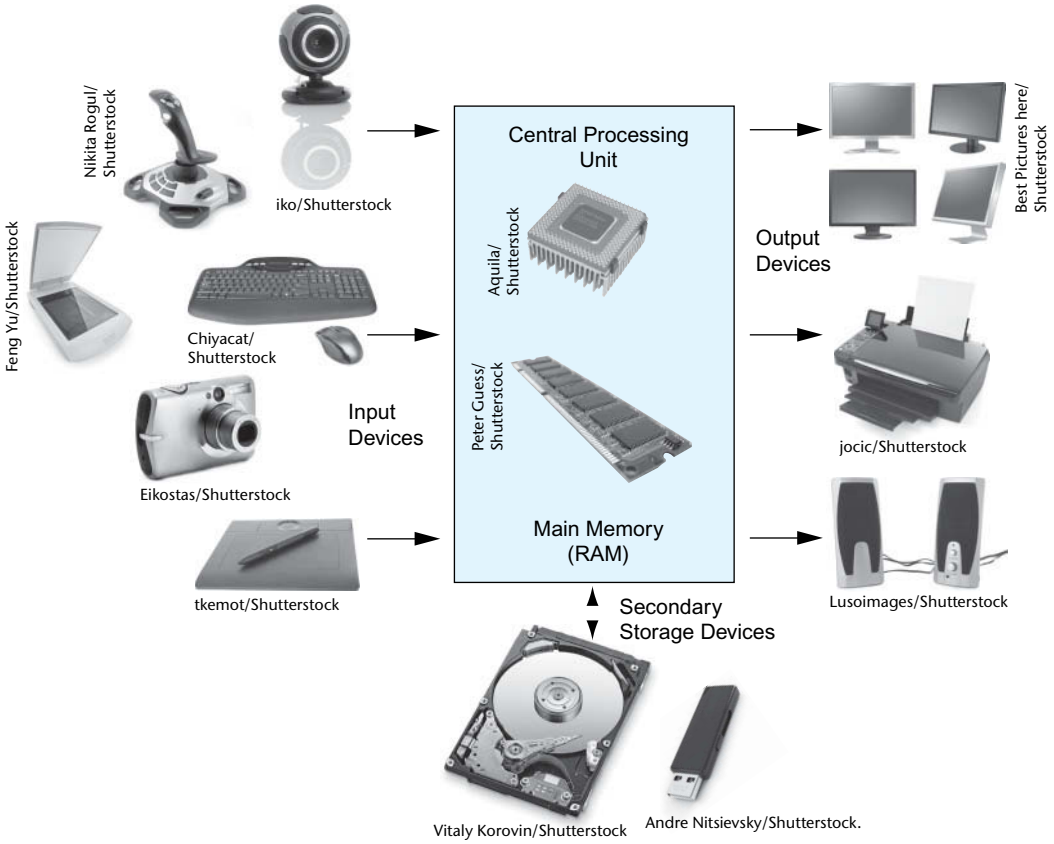
Hardware

The term *hardware* refers to all of the physical devices, or *components*, of which a computer is made. A computer is not one single device, but a system of devices that all work together. Like the different instruments in a symphony orchestra, each device in a computer plays its own part.

If you have ever shopped for a computer, you've probably seen sales literature listing components such as microprocessors, memory, disk drives, video displays, graphics cards, and so on. Unless you already know a lot about computers, or at least have a friend that does, understanding what these different components do might be challenging. As shown in Figure 1-2, a typical computer system consists of the following major components:

- The central processing unit (CPU)
- Main memory

Figure 1-2 Typical components of a computer system



- Secondary storage devices
- Input devices
- Output devices

Let's take a closer look at each of these components.

The CPU

When a computer is performing the tasks that a program tells it to do, we say that the computer is *running* or *executing* the program. The *central processing unit,* or *CPU,* is the part of a computer that actually runs programs. The CPU is the most important component in a computer because without it, the computer could not run software.

In the earliest computers, CPUs were huge devices made of electrical and mechanical components such as vacuum tubes and switches. Figure 1-3 shows such a device. The two women in the photo are working with the historic ENIAC computer. The ENIAC, which is considered by many to be the world's first programmable electronic computer, was built in 1945 to calculate artillery ballistic tables for the U.S. Army. This machine, which was primarily one big CPU, was 8 feet tall, 100 feet long, and weighed 30 tons.

Today, CPUs are small chips known as *microprocessors*. Figure 1-4 shows a photo of a lab technician holding a modern microprocessor. In addition to being much smaller than the old electromechanical CPUs in early computers, microprocessors are also much more powerful.