Chapter 1

Introduction

At a Glance

Instructor's Manual Table of Contents

Lecture Notes

Overview

Chapter 1 describes the basic features of an algorithm. Students learn how hardware and software collaborate in a computer's architecture. A brief history of computing is provided. Finally, students learn how to compose and run a simple Python program.

Objectives

After completing this chapter, students will be able to:

- Describe the basic features of an algorithm
- Explain how hardware and software collaborate in a computer's architecture
- Give a brief history of computing
- Compose and run a simple Python program

Teaching Tips

Two Fundamental Ideas of Computer Science: Algorithms and Information Processing

1. Explain that computer science focuses on a broad set of interrelated ideas; two of the most basic of these ideas are algorithms and information processing.

Algorithms

1. Introduce a simple algorithm in class. You may use the subtraction algorithm provided in the book, a cooking recipe, or any other sequence of instructions that your students can relate to. Stress that when these instructions are carried out, the *computing agent* is a human being.

Teaching Tip	For more information on the history of algorithms, ask your students to visit http://www.scriptol.com/programming/algorithm-history.php.

2. Provide a formal definition of the term *algorithm*.

3. List and explain the four features of an algorithm, specified on Pages 3 and 4 of the text.

Information Processing

1. Describe the role of information processing, introducing the terms *data*, *input* and *output*.

The Structure of a Modern Computer System

1. Explain that a modern computer system consists of *hardware* and *software*. Note that software is represented as *programs* in particular *programming languages*.

Computer Hardware

1. Use Figure 1.1 to describe the role of each of the most important hardware components of a modern computer system. Note that not shown in the figure are the computer *ports*, which enable computers to connect to *networks* and to other devices.

2. Use Figure 1.2 to help explain how the computer's *primary* or *internal* or *RAM* memory works, explaining that information is stored as patterns of *binary digits*.

3. Explain the concept of a *processor*, or CPU, and how it is structured to perform simple operations.

4. Explain the difference between internal and *external* or *secondary memory*. Note that examples of the latter are *magnetic media*, *semiconductor media*, and *optical media*, and give examples of each of these media.

Computer Software

1. Explain the concept of computer software as a program stored in memory that can be executed later. Point out that software is stored in *machine code* and is loaded into the memory by a *loader*.

2. Explain the concept of *system software*, specifying that the most important example of system software is the *operating system*, which includes a *file system*, one or more *user interfaces*, a *terminal-based interface*, and sometimes also a *graphical user interface* (*GUI*).

3. Explain the concept of *application software*, specifying how it is created. Be sure to introduce the terms *high-level programming languages*, *text editor*, *translator*, *syntax errors*, *run-time system*, *interpreter*, and *virtual machine.*

Teaching Tip	For more on the history of computer software, visit http://www.computerhistory.org/timeline/?category=sl.

4. Use Figure 1.3 to describe the role of the computer software used during the coding process.

Quick Quiz 1

1. What is an algorithm?
   Answer: Informally, an algorithm is like a recipe. It provides a set of instructions that tells us how to do something, such as make change, bake bread, or put together a piece of furniture. More precisely, an algorithm describes a process that ends with a solution to a problem.

2. The part of a computer that is responsible for processing data is the _____ (CPU).
   Answer: central processing unit

3. True or False: A program stored in computer memory must be represented in binary digits, which is also known as machine code.
   Answer: True

4. A modern _____ (GUI) organizes the monitor screen around the metaphor of a desktop, with windows containing icons for folders, files, and applications.
   Answer: graphical user interface

A Not-So-Brief History of Computing Systems

1. Use Figure 1.4 to provide a brief overview of some of the major developments in the history of computing.

Before Electronic Digital Computers

1. Briefly note some of the major developments in the history of computing before 1940. Some important names to mention are: Pascal, Leibnitz, Jacquard, Babbage, Hollerith, Boole, and Turing.

The First Electronic Digital Computers (1940–1950)

1. Briefly note some of the major developments in the history of computing that took place during the 1940s. Note that a few *mainframe computers* were developed in this period (e.g., Mark I, ENIAC, ABC, Colossus).

2. Stress that John von Neumann developed the first memory-stored programs.

The First Programming Languages (1950–1965)

1. Describe the evolution of the first programming languages, from *assembly languages* to *high-level programming languages* like FORTRAN, LISP, and COBOL. Be sure to explain the terms *keypunch machine*, *card reader*, *assembler*, *compiler*, *interpreter*, *artificial intelligence*, and *abstraction*.

Teaching Tip	For a brief history of computer languages and their evolution, visit http://www.scriptol.org/history.php.

Integrated Circuits, Interaction, and Timesharing (1965–1975)

1. Explain how the invention of the *transistor*, and later of the *integrated circuit*, led to the construction of smaller, faster, less expensive hardware components.

2. Introduce Moore's Law.

Teaching Tip	You can find a graph that illustrates Moore's Law at http://arstechnica.com/hardware/news/2008/09/moore.ars.

3. Note that computer processing evolved from *batch processing* to *time-sharing* to *concurrent processing*.

Personal Computing and Networks (1975–1990)

1. Describe the contribution of Engelbart to the creation of the first personal computers.

2. Note some other important developments, such as the Altair, the MS-DOS, the Ethernet, and ARPANET's evolution into the Internet.

Consultation, Communication, and Ubiquitous Computing (1990–Present)

1. Note some of the major developments of modern computing, like *optical storage media*, *virtual reality*, and *hypermedia*.

2. Provide a brief overview of the World Wide Web, making sure to explain the terms *Web server*, *Web browser*, and *Web client*.

Quick Quiz 2

1. In the late 1930s, _____, a mathematician and electrical engineer at M.I.T., wrote a classic paper titled "A Symbolic Analysis of Relay and Switching Circuits."
   Answer: Claude Shannon

2. True or False: In the early 1970s, computer scientists realized that a symbolic notation could be used instead of machine code, and the first assembly languages appeared.
   Answer: False

3. What is Moore's Law?
   Answer: This prediction states that the processing speed and storage capacity of hardware will increase and its cost will decrease by approximately a factor of 2 every 18 months.

4. By the mid 1980s, the _____ had grown into what we now call the Internet, connecting computers owned by large institutions, small organizations, and individuals all over the world.
   Answer: ARPANET

Getting Started with Python Programming

1. Note that Guido van Rossum invented the Python programming language in the early 1990s.

2. Explain that Python is a high-level, general-purpose programming language for solving problems on modern computer systems.

Teaching Tip	Python is an interpreted language. Are your students familiar with other interpreted languages? Ask them to discuss their experiences in class.

Running Code in the Interactive Shell

1. Use Figure 1.6 and one or more real examples to show students how to write and run code in the Python Shell.

2. Demonstrate to students how they can get help through the command prompt or through the drop-down menu in the Python shell.

3. Explain and demonstrate how to quit the Python shell.

Input, Processing, and Output

1. Explain the concept of programming language syntax as it applies to Python.

2. Define the concept of a variable, and explain how one can be created to store user input.

3. Define the concept of type conversion functions, and explain how they can be used to make the user input suitable for the program's needs.

4. Use a few examples to show students how to perform simple input and output operations in Python.

Editing, Saving, and Running a Script

1. Explain the concept of a script as it applies to Python programs.

2. Use Figures 1.7 and 1.8 to explain how to edit, save, and run scripts in an IDLE window.

3. Define the concept of a program library, and explain how such libraries can be used.

Behind the Scenes: How Python Works

1. Use Figure 1.9 to describe how Python code is interpreted and executed in the *Python Virtual Machine* (*PVM*).

Teaching Tip	A good Python programming tutorial is available at http://docs.python.org/tut/.

Detecting and Correcting Syntax Errors

1. Explain what the *syntax* of a programming language is.

2. Use one or more examples to stress that when Python encounters a syntax error in a program, it halts execution with an error message.

Quick Quiz 3

1. The easiest way to open a Python shell is to launch the _____.
   Answer: IDLE

2. What is a shell prompt?
   Answer: A shell window contains an opening message followed by the special symbol >>>, called a shell prompt.

3. True or False: In Python, to begin the next output on the same line as the previous one, you can place a colon at the end of the earlier `print` statement.
   Answer: False

4. The term _____ refers to the rules for forming sentences in a language.
   Answer: syntax

Class Discussion Topics

1. Ask your students to talk about any previous programming experience they might have had. What other programming languages are they familiar with?

2. Students familiar with other programming languages may be surprised that Python provides an interactive shell that can be used to test simple statements and expressions.

Do they think other programming languages should provide similar functionality? Why or why not?

Additional Projects

1. Ask students to do some research and create a timetable that describes the evolution of computer languages from machine language to the present day. The table does not have to be comprehensive, but it should include approximately ten major languages. For each language, they should include the creation date and author(s), a brief description, and the state of its use today.

2. Ask your students how they would like their final class grade to be determined in terms of the relative weight given to projects, weekly assignments, midterm, final, etc. Then, ask them to design an algorithm to calculate their final grade using the percentages they have chosen.

Additional Resources

1. History and evolution of computer languages:
   www.scriptol.org/history.php

2. Algorithm:
   http://www.wisegeek.com/what-is-an-algorithm.htm

3. Software:
   http://www.wordiq.com/definition/Software

4. Python Tutorial:
   http://docs.python.org/tut/

Key Terms

- **abacus:** An early computing device that allowed users to perform simple calculations by moving beads along wires.
- **abstraction:** A simplified view of a task or data structure that ignores complex detail.
- **algorithm:** A finite sequence of instructions that, when applied to a problem, will solve it.
- **Analytical Engine:** A general-purpose computer designed in the nineteenth century by Charles Babbage, but never completed by him.
- **application software:** Programs that allow human users to accomplish specialized tasks, such as word processing or database management.
- **argument:** A value or expression passed in a function or method call.
- **artificial intelligence:** A field of computer science whose goal is to build machines that can perform tasks that require human intelligence.
- **assembler:** A program that translates an assembly language program to machine code.

- **assembly language:** A computer language that allows the programmer to express operations and memory addresses with mnemonic symbols.
- **batch processing:** The scheduling of multiple programs so that they run in sequence on the same computer.
- **binary digit:** A digit, either 0 or 1, in the binary number system. Program instructions are stored in memory using a sequence of binary digits. See also bit.
- **bit:** A binary digit.
- **bitmap:** A data structure used to represent the values and positions of points on a computer screen or image.
- **bit-mapped display screen:** A type of display screen that supports the display of graphics and images.
- **byte:** A sequence of bits used to encode a character in memory.
- **byte code:** The kind of object code generated by a Python compiler and interpreted by a Python virtual machine. Byte code is platform independent.
- **card reader:** A device that inputs information from punched cards into the memory of a computer.
- **central processing unit (CPU):** A major hardware component that consists of the arithmetic/logic unit and the control unit. Also sometimes called a **processor**.
- **coding:** The process of writing executable statements that are part of a program to solve a problem. See also implementation.
- **compiler:** A computer program that automatically converts instructions in a high-level language to byte code or machine language.
- **computing agent:** The entity that executes instructions in an algorithm.
- **concurrent processing:** The simultaneous performance of two or more tasks.
- **data:** The symbols that are used to represent information in a form suitable for storage, processing, and communication.
- **execute:** To carry out the instructions of a program.
- **external (or secondary) memory:** Memory that can store large quantities of data permanently.
- **file system:** Software that organizes data on secondary storage media.
- **GUI (graphical user interface):** A means of communication between human beings and computers that uses a pointing device for input and a bitmapped screen for output. The bitmap displays images of windows and window objects such as buttons, text fields, and drop-down menus. The user interacts with the interface by using the mouse to directly manipulate the window objects. See also window object.
- **hardware:** The computing machine and its support devices.
- **high-level programming language:** Any programming language that uses words and symbols to make it relatively easy to read and write a program. See also assembly language and machine language.
- **HTML (HyperText Markup Language):** A programming language that allows the user to create pages for the World Wide Web.
- **hypermedia:** A data structure that allows the user to access different kinds of information (text, images, sound, video, applications) by traversing links.
- **hypertext:** A data structure that allows the user to access different chunks of text by traversing links.
- **identifiers:** Words that must be created according to a well-defined set of rules but can have any meaning subject to these rules.
- **IDE (integrated development environment):** A set of software tools that allows you to edit, compile, run, and debug programs within one user interface.

- **information processing:** The transformation of one piece of information into another piece of information.
- **input:** Data obtained by a program during its execution.
- **input device:** A device that provides information to the computer. Typical input devices are a mouse, keyboard, disk drive, microphone, and network port. See also I/O device and output device.
- **integrated circuit:** The arrangement of computer hardware components in a single, miniaturized unit.
- **internal memory:** See main memory
- **interpreter:** A program that translates and executes another program.
- **I/O device:** Any device that allows information to be transmitted to or from a computer. See also input device and output device.
- **keypunch machine:** An early input device that allowed the user to enter programs and data onto punched cards.
- **library:** A collection of methods and data organized to perform a set of related tasks.
- **linear:** An increase of work or memory in direct proportion to the size of a problem.
- **loader:** A system software tool that places program instructions and data into the appropriate memory locations before program start-up.
- **machine code:** The language used directly by the computer in all its calculations and processing. Also called machine language.
- **magnetic storage media:** Any media that allow data to be stored as patterns in a magnetic field.
- **main (primary or internal) memory:** The high-speed internal memory of a computer, also referred to as random access memory (RAM). See also memory and secondary memory.
- **mainframe:** Large computers typically used by major companies and universities.
- **memory:** The ordered sequence of storage cells that can be accessed by address. Instructions and variables of an executing program are temporarily held here. See also main memory and secondary memory.
- **memory location:** A storage cell that can be accessed by address.
- **microcomputer:** A computer capable of fitting on a laptop or desktop, generally used by one person at a time.
- **Moore's Law:** A hypothesis that states that the processing speed and storage capacity of computers will increase by a factor of two every 18 months.
- **network:** A collection of resources that are linked together for communication.
- **newline character:** A special character ('\n') used to indicate the end of a line of characters in a string or a file stream.
- **operating system:** A large program that allows the user to communicate with the hardware and performs various management tasks.
- **optical storage media:** Devices such as CDs and DVDs that store data permanently and from which the data are accessed by using laser technology.
- **output:** Information that is produced by a program.
- **output device:** A device that allows you to see the results of a program. Typically, it is a monitor, printer, speaker, or network port.
- **port:** A channel through which several clients can exchange data with the same server or with different servers.
- **primary memory:** See main memory
- **processor:** See Central Processing Unit (CPU)
- **program:** A set of instructions that tells the machine (the hardware) what to do.

- **program library:** A collection of operations and data organized to perform a set of related tasks.
- **programming language:** A formal language that computer scientists use to give instructions to the computer.
- **PVM (Python Virtual Machine):** A program that interprets Python byte codes and executes them.
- **RAM (random access memory):** High-speed memory where programs and their data reside during program execution.
- **run-time system:** Software that supports the execution of a program.
- **script:** A Python program that can be launched from a computer's operating system.
- **secondary (external) memory:** An auxiliary device for memory, usually a disk or magnetic tape.
- **semiconductor storage media:** Devices, such as flash sticks, that use solid state circuitry to store data permanently.
- **shell:** A program that allows users to enter and run Python program expressions and statements interactively.
- **software:** Programs that make the machine (the hardware) do something, such as word processing, database management, or games.
- **software reuse:** The process of building and maintaining software systems out of existing software components.
- **solid-state device:** An electronic device, typically based on a transistor, which has no moving parts.
- **source code:** The program text as viewed by the human being who creates or reads it, prior to compilation.
- **statement:** An individual instruction in a program.
- **syntax:** The rules for constructing well-formed programs in a language. Also, the rules for forming sentences in a language.
- **syntax error:** An error in spelling, punctuation, or placement of certain key symbols in a program. See also compilation error, design error, and run time error.
- **system software:** The programs that allow users to write and execute other programs, including operating systems such as Windows and Mac OS.
- **Terminal-based interface:** A user interface that allows the user to enter input from a keyboard and view output as text in a window. Also called a terminal-based interface.
- **text editor:** A program that allows the user to enter text, such as a program, and save it in a file.
- **time sharing:** The scheduling of multiple programs so that they run concurrently on the same computer.
- **time-sharing operating system:** A computer system that can run multiple programs in such a manner that its users have the illusion that they are running simultaneously.
- **transistor:** A device with no moving parts that can hold an electromagnetic signal and that is used to build computer circuitry for memory and a processor.
- **translator:** A program that converts a program written in one language to an equivalent program in another language.
- **type conversion functions:** A function that takes one type of data as an argument and returns the same data represented in another type.
- **user interface:** The part of a software system that handles interaction with users.
- **variable:** A memory location, referenced by an identifier, whose value can be changed during execution of a program.
- **variable identifier:** A name used to reference a variable.

- **virtual machine:** A software tool that behaves like a high-level computer.
- **virtual reality:** A technology that allows a user to interact with a computer-generated environment, usually simulating movement in three dimensions.
- **Web browser:** Software used to view information on the Web.
- **Web client:** Software on a user's computer that makes requests for resources from the Web.
- **Web server:** Software on a computer that responds to requests for resources and makes them available on the Web.
- **window:** A rectangular area of a computer screen that can contain window objects. Windows typically can be resized, minimized, maximized, zoomed, or closed.