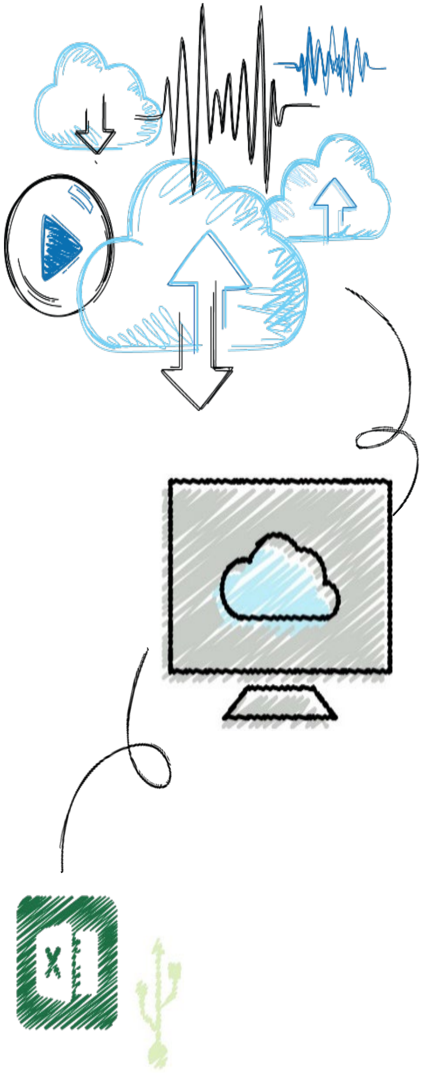
Fundamentals of Python: First Programs Second Edition

Chapter 1

Introduction



Objectives

1.1 Describe the basic features of an algorithm

1.2 Explain how hardware and software collaborate in a computer's architecture

1.3 Give a brief history of computing

1.4 Compose and run a simple Python program

CENGAGE

Two Fundamental Ideas of Computer Science: Algorithms and Information Processing

- Computer science focuses on a broad set of interrelated ideas
- Two of the most basic ones are:
  - **Algorithms**
  - **Information processing**

Algorithms (1 of 2)

- Steps for subtracting two numbers:
  - **Step 1:** Write down the numbers, with larger number above smaller one, digits column-aligned from right
  - **Step 2:** Start with rightmost column of digits and work your way left through the various columns
  - **Step 3:** Write down difference between the digits in the current column of digits, borrowing a 1 from the top number’s next column to the left if necessary
  - **Step 4:** If there is no next column to the left, stop
    - Otherwise, move to column to the left; go to Step 3
- The **computing agent** is a human being
- Sequence of steps that describes each of these computational processes is called an **algorithm**

CENGAGE

Algorithms (2 of 2)

- Features of an algorithm:
  - Consists of a finite number of instructions
  - Each individual instruction is well defined
    - Action described by the instruction can be performed effectively or be executed by a computing agent
  - Describes a process that eventually halts after arriving at a solution to a problem
  - Solves a general class of problems
- Computers can be designed to run a small set of algorithms for performing specialized tasks

CENGAGE

Information Processing

- Information is also commonly referred to as **data**
- In carrying out the instructions of an algorithm, computing agent manipulates information
  - Starts with **input**
  - Transforms information according to well-defined rules
  - Produces **output**
- The algorithms that describe information processing can also be represented as information
- Computer scientists recently discovered how to represent many other things, such as:
  - Images, music, human speech, and video

The Structure of a Modern Computer System

- A modern computer system consists of **hardware** and **software**
  - Hardware: physical devices required to execute algorithms
  - Software: set of these algorithms, represented as **programs** in particular **programming languages**

Computer Hardware (1 of 3)

- Basic hardware components of a computer are:
  - Memory
  - Central processing unit (CPU)
  - Set of input/output devices
- Computers can also communicate with the external world through various **ports** that connect them to **networks** and to other devices

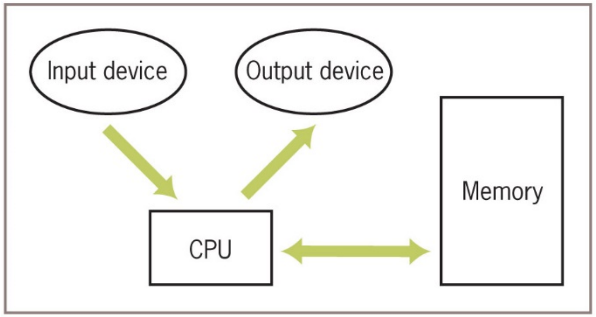
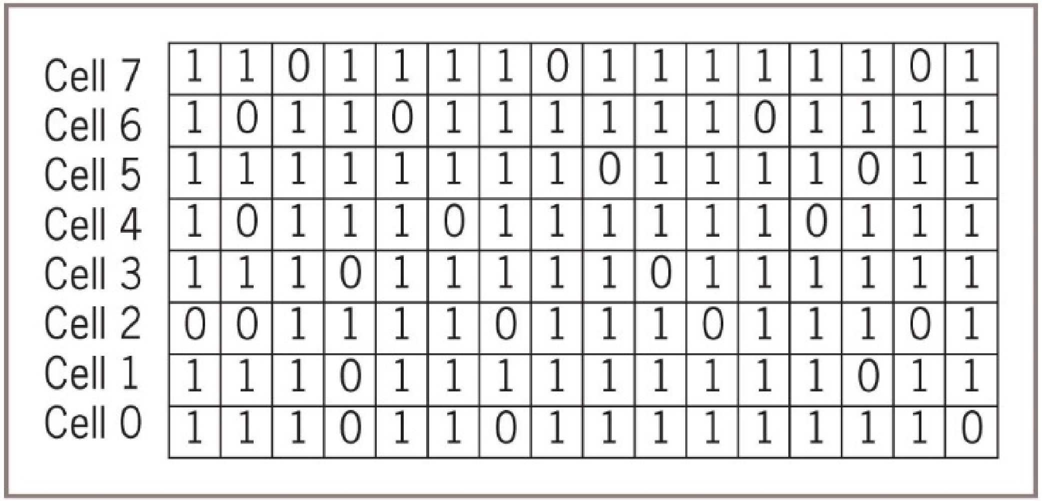


Figure 1-1 Hardware components of a modern computer system

Computer Hardware (2 of 3)

- Computer memory is set up to represent and store information in electronic form
  - Stored as patterns of binary digits (1s and 0s)
- **Random access memory (RAM)** is also called **internal** or primary
- Part of a computer responsible for processing data is the central processing unit (CPU), also called **processor**
- **External** or secondary memory can be **magnetic**, **semiconductor**, or **optical**

Computer Hardware (3 of 3)



Cell 7	1	1	0	1	1	1	1	0	1	1	1	1	1	1	0	1
Cell 6	1	0	1	1	0	1	1	1	1	1	1	0	1	1	1	1
Cell 5	1	1	1	1	1	1	1	1	0	1	1	1	1	0	1	1
Cell 4	1	0	1	1	1	0	1	1	1	1	1	1	0	1	1	1
Cell 3	1	1	1	0	1	1	1	1	1	0	1	1	1	1	1	1
Cell 2	0	0	1	1	1	1	0	1	1	1	0	1	1	1	0	1
Cell 1	1	1	1	0	1	1	1	1	1	1	1	1	1	0	1	1
Cell 0	1	1	1	0	1	1	0	1	1	1	1	1	1	1	1	0

Figure 1-2 A model of computer memory

Computer Software (1 of 3)

- A program stored in computer memory must be represented in binary digits, or **machine code**
- A **loader** takes a set of machine language instructions as input and loads them into the appropriate memory locations
- The most important example of **system software** is a computer's **operating system**
  - Some important parts: **file system**, **user interfaces** (**terminal-based**, **GUIs**, or **touchscreen interface**)
- **Applications** include Web browsers, word processors, spreadsheets, database managers, graphic design packages, games, etc…

Computer Software (2 of 3)

- Scientists have developed **high-level programming languages** for expressing algorithms
  - Resemble English and allow the author to express algorithms in a form that other people can understand
- Programmers usually start by writing high-level language statements in a **text editor**
  - Runs another program called a **translator** to convert program code into executable code
  - Translator checks for **syntax errors**
- If no errors are found, program can be executed by the **run-time system**
  - Might execute program directly on the hardware or run another program called an **interpreter** or **virtual machine** to execute the program

CENGAGE

Computer Software (3 of 3)

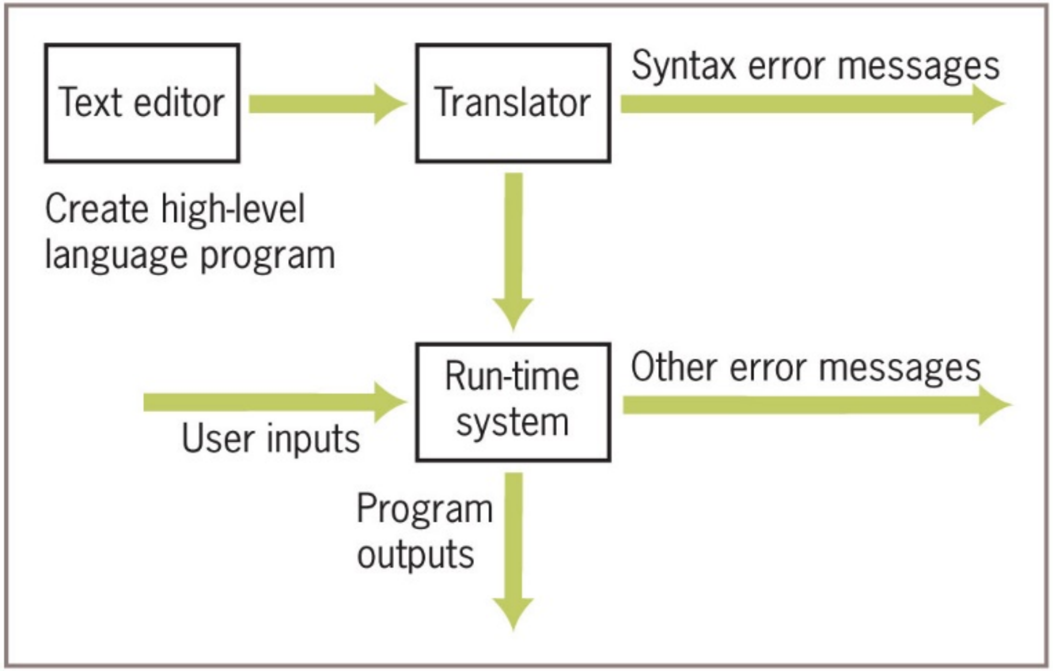
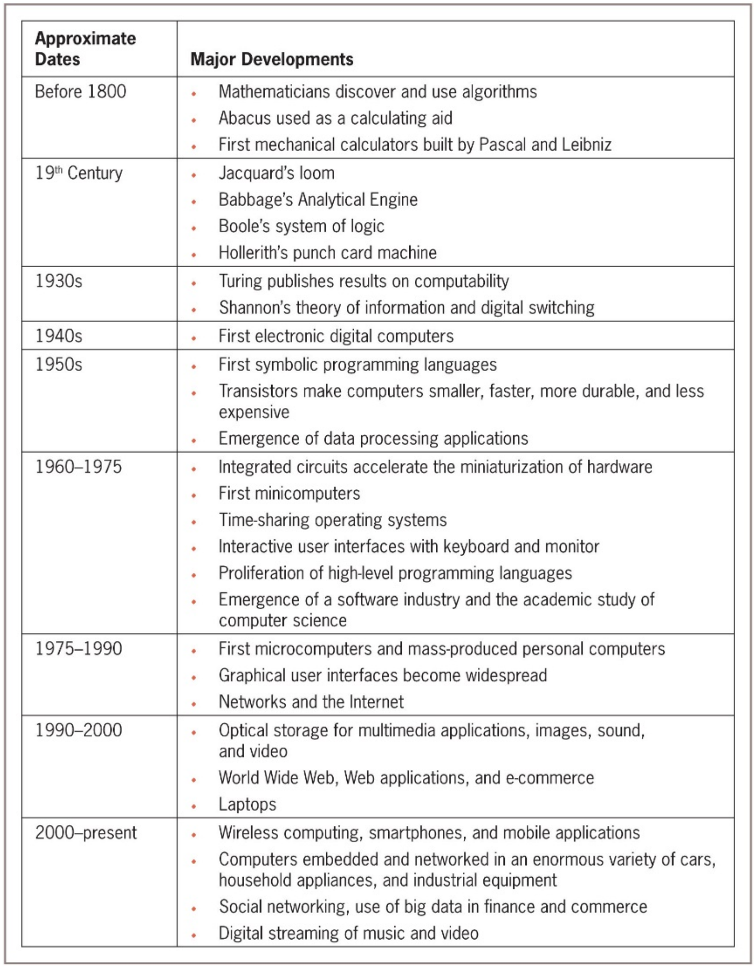


Figure 1-3 Software used in the coding process

A Not-So-Brief History of Computing Systems



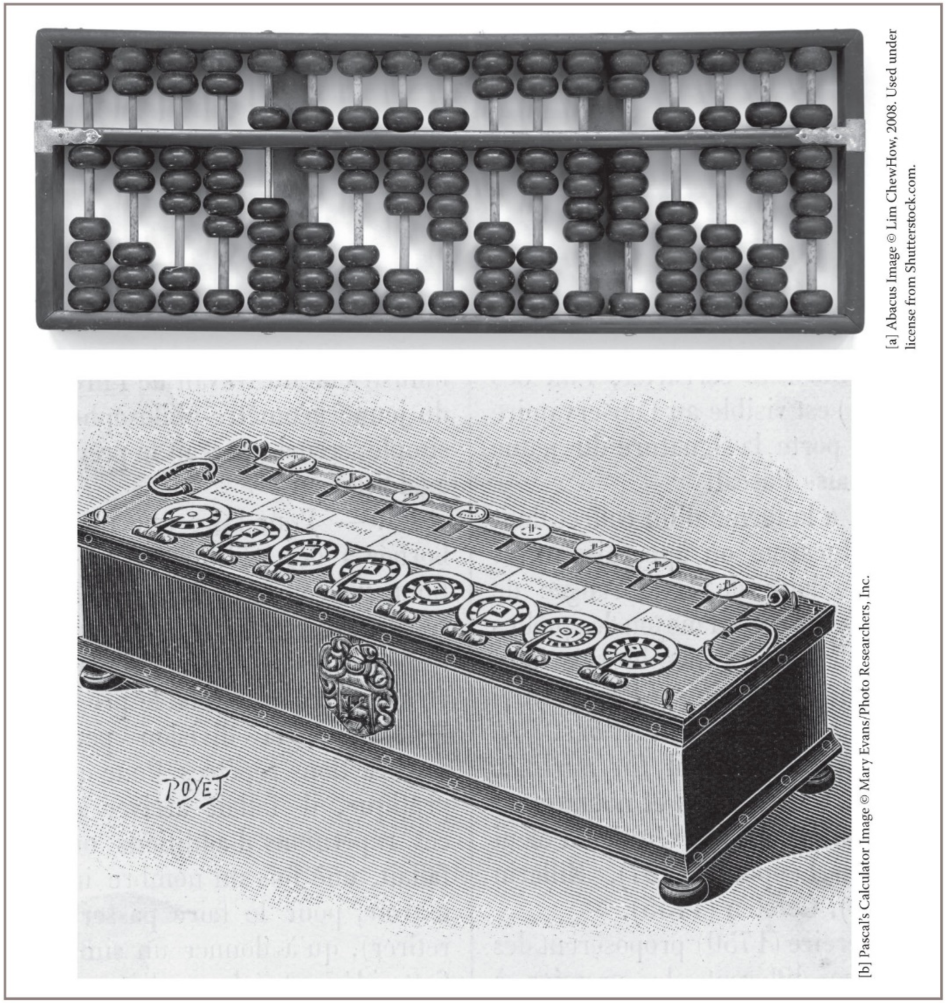
Approximate Dates	Major Developments
Before 1800	- Mathematicians discover and use algorithms - Abacus used as a calculating aid - First mechanical calculators built by Pascal and Leibniz
19th Century	- Jacquard's loom - Babbage's Analytical Engine - Boole's system of logic - Hollerith's punch card machine
1930s	- Turing publishes results on computability - Shannon's theory of information and digital switching
1940s	- First electronic digital computers
1950s	- First symbolic programming languages - Transistors make computers smaller, faster, more durable, and less expensive - Emergence of data processing applications
1960–1975	- Integrated circuits accelerate the miniaturization of hardware - First minicomputers - Time-sharing operating systems - Interactive user interfaces with keyboard and monitor - Proliferation of high-level programming languages - Emergence of a software industry and the academic study of computer science
1975–1990	- First microcomputers and mass-produced personal computers - Graphical user interfaces become widespread - Networks and the Internet
1990–2000	- Optical storage for multimedia applications, images, sound, and video - World Wide Web, Web applications, and e-commerce - Laptops
2000–present	- Wireless computing, smartphones, and mobile applications - Computers embedded and networked in an enormous variety of cars, household appliances, and industrial equipment - Social networking, use of big data in finance and commerce - Digital streaming of music and video

Figure 1-4 Summary of major developments in the history of computing

Before Electronic Digital Computers (1 of 4)

- “Algorithm” comes from Muhammad ibn Musa Al-Khawarizmi, a Persian mathematician
- Euclid developed an algorithm for computing the greatest common divisor of two numbers
- The **abacus** also appeared in ancient times
- Blaise Pascal (1623–1662): built one of the first mechanical devices to automate addition
- Wilhelm Leibniz (1646-1716): built another calculator that included other arithmetic functions such as multiplication
- Joseph Jacquard (1752–1834): designed and constructed a machine that automated weaving

Before Electronic Digital Computers (2 of 4)

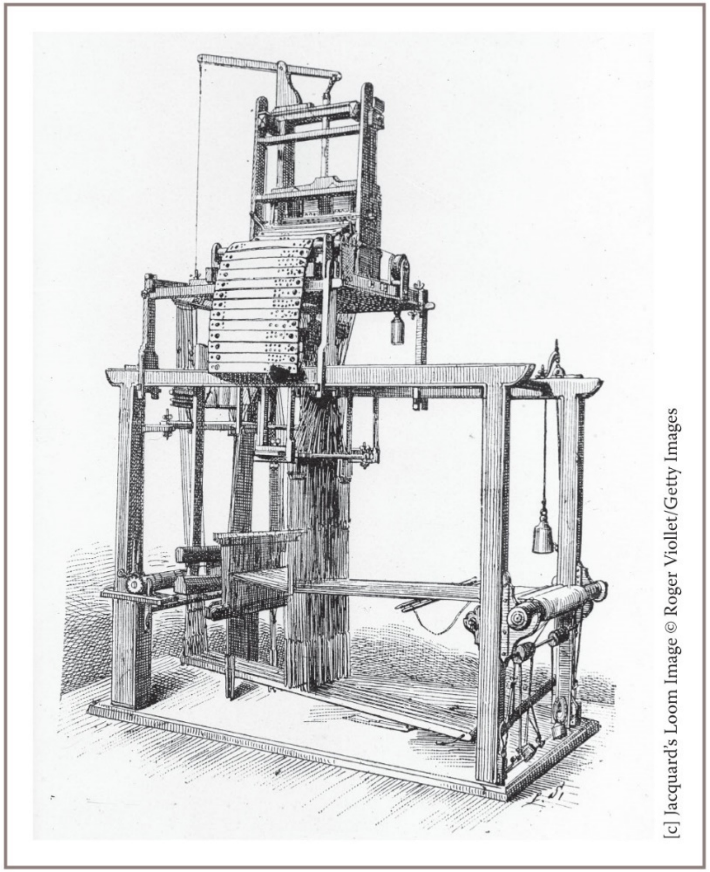


[a] Abacus Image © Lim ChewHow, 2008. Used under license from Shutterstock.com.

[b] Pascal's Calculator Image © Mary Evans/Photo Researchers, Inc.

Figure 1-5 Some early computing devices

Before Electronic Digital Computers (3 of 4)



[c] Jacquard's Loom Image © Roger Viollet/Getty Images

Figure 1-5 *(Continued)*

Before Electronic Digital Computers (4 of 4)

- Charles Babbage (1792–1871): conceived Analytical Engine
- Herman Hollerith (1860–1929): developed a machine that automated data processing for the U.S. Census
  - One of the founders of company that became IBM
- George Boole (1815–1864): developed Boolean logic
- Alan Turing (1912–1954): explored the theoretical foundations and limits of algorithms and computation

The First Electronic Digital Computers (1940-1950)

- Late 1930s: Claude Shannon wrote paper titled “A Symbolic Analysis of Relay and Switching Circuits”
- 1940s:
  - Mark I (electromechanical)
  - ENIAC (Electronic Numerical Integrator and Calculator)
  - ABC (Atanasoff-Berry Computer)
  - Colossus by a group working under Alan Turing
  - John von Neumann: first memory-stored programs
- **Mainframe computers** consisted of vacuum tubes, wires, and plugs, and filled entire rooms

The First Programming Languages (1950-1965)

- The first **assembly languages** had operations like ADD and OUTPUT
- Programmers entered mnemonic codes for operations at **keypunch machine**
- **Card reader**—translated holes in cards to patterns in computer's memory
- **Assembler—**translated application programs in memory to machine code
- **Compiler** – translated programs to machine code
- High-level programming languages: FORTRAN, LISP, COBOL
  - common feature: **abstraction**

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

- Late 1950s: vacuum tube gave way to **transistor**
    - Transistor is **solid-state** device
- Early 1960s: **integrated circuit** enabled smaller, faster, less expensive hardware components
    - Moore’s Law: processing speed and storage capacity of HW will increase and cost will decrease by approximately a factor of 2 every 18 months
- Minicomputers appeared
- Processing evolved from:
    - **Batch processing**
    - **Time-sharing**
    - **Concurrent processing**

Personal Computing and Networks (1975-1990)

- Late 1960s: Douglas Engelbart
  - First pointing device (mouse) and software to represent windows, icons, and pull-down menus on a **bit-mapped display screen**
  - Member of team that developed Alto (Xerox PARC)
- 1975: Altair, first mass-produced personal computer
  - With Intel’s 8080 processor, first **microcomputer** chip
- Early 1980s: Gates and Allen build MS-DOS
- Bob Metcalfe created Ethernet, used in LANs
- ARPANET grew into what we call Internet

Consultation, Communication, and E-Commerce (1990-2000) (1 of 2)

- Optical storage media was developed for mass storage
- **Virtual reality:** capacity to create lifelike 3-D animations of whole-environments
- Berners-Lee at CERN created WWW
  - Based on concepts of **hypermedia**
  - **HTTP**: Hypertext Transfer Protocol
  - **HTML**: Hypertext Markup Language
- Components of WWW:
  - Web servers
  - Web browsers
  - Web clients

Consultation, Communication, and E-Commerce (1990-2000) (2 of 2)

- **Web applications** – presented a revolution in the way software services were delivered to people
  - Made online stores pervasive
  - Web application providing the service ran on a remote computer or server
- **Client/server applications** – such as e-mail, bulletin boards, and chat rooms
  - Were already in use
  - Simply deployed on the Web when it became available
- Sergey Brin and Larry Page
  - Developed algorithms for indexing and searching the Web

Mobile Applications and Ubiquitous Computing (2000-present)

- Personal digital assistants (PDAs) – first handheld computing devices
  - Limited to video games, address books, to-do lists, and note taking
- Steve Jobs (Apple) created several key devices
  - iPod
  - iPhone
  - iPad
- Social networking applications – major addition to the digital landscape
- Big data – a technology where governments, businesses, and hackers continually monitor Internet traffic
  - Researchers have created algorithms that process massive amounts of data to discover trends and predict outcomes

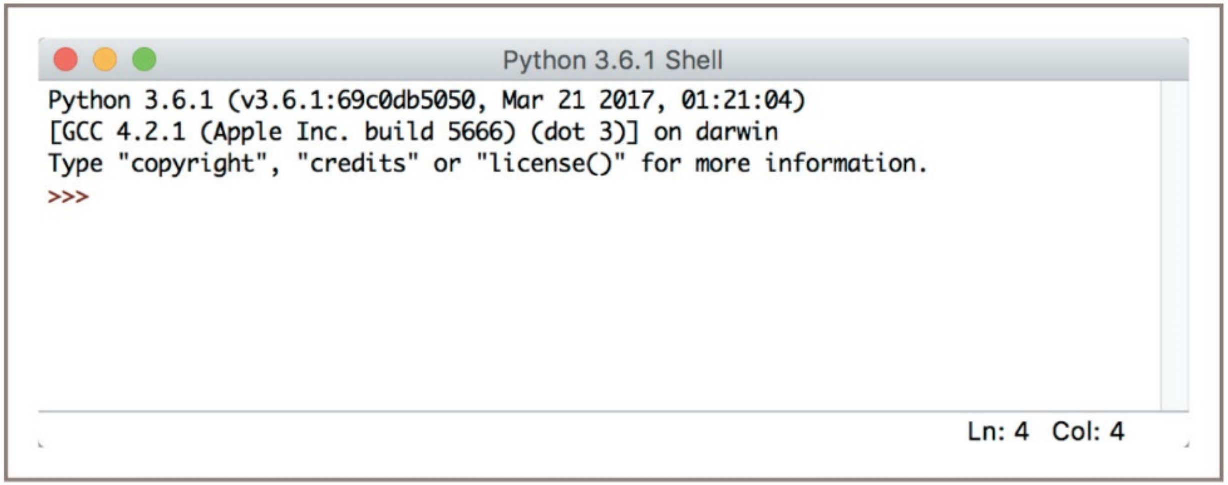
Getting Started with Python Programming

- Early 1990s: Guido van Rossum
  - invented the Python programming language
- **Python** is a high-level, general-purpose programming language for solving problems on modern computer systems
- Useful resources at www.python.org

Running Code in the Interactive Shell (1 of 2)

- Python is an **interpreted** language
- Simple Python expressions and statements can be run in the **shell**
  - Easiest way to open a Python shell is to launch the IDLE
  - To quit, select the window’s close box or press Control+D
  - Shell is useful for:
    - Experimenting with short expressions or statements
    - Consulting the documentation

Running Code in the Interactive Shell (2 of 2)



```
Python 3.6.1 (v3.6.1:69c0db5050, Mar 21 2017, 01:21:04)
[GCC 4.2.1 (Apple Inc. build 5666) (dot 3)] on darwin
Type "copyright", "credits" or "license()" for more information.
>>>
```

Figure 1-6 Python shell window

Input, Processing, and Output (1 of 5)

- Programs usually accept inputs from a source, process them, and output results to a destination
  - In terminal-based interactive programs, these are the keyboard and terminal display
- In Python, inputs are Python expressions or statements
  - Outputs are the results displayed in the shell
- Programmers can also force output of a value by using the print function
  - **print (<expression>)**
- Example:

```
>>>print (“Hi there”)
Hi there
```

CENGAGE

Input, Processing, and Output (2 of 5)

- The following example receives an input string from the user and saves it for further processing:

```
>>> name = input(“Enter your name:”)
Enter your name: Ken Lambert
>>> name
‘Ken Lambert’
>>> print(name)
Ken Lambert
>>>
```

Input, Processing, and Output (3 of 5)

- The **input** function always builds a string from the user’s keystrokes and returns it to the program
- Strings that represent numbers must be converted from strings to appropriate number types
    - Two type conversion functions: **int** (for integers) and **float** (for floating-point numbers)

Input, Processing, and Output (4 of 5)

- The next session inputs two integers and displays their sum:

```
>>> first = int(input(“Enter the first number: ”))
Enter the first number: 23
>>> second = int(input(“Enter the second number:”))
Enter the second number: 44
>>> print(“The sum is”, first + second)
The sum is 67
```

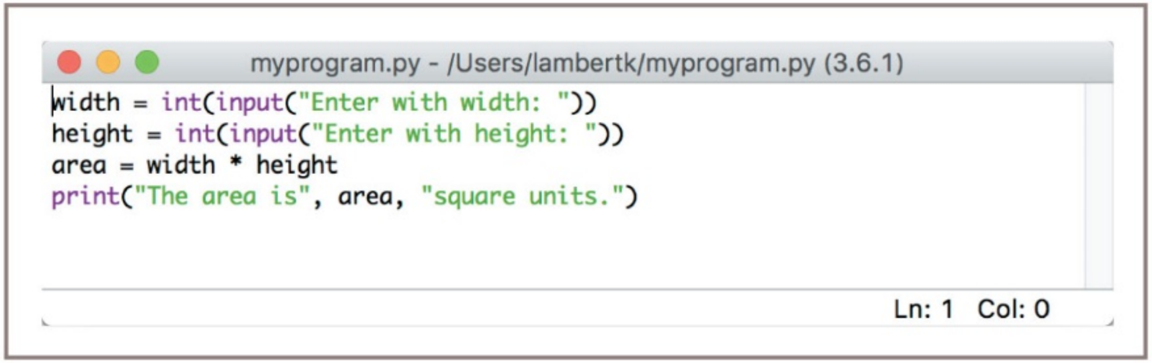
Input, Processing, and Output (5 of 5)

Function	What It Does
float(<a *string of digits*>)	Converts a string of digits to a floating-point value.
int(<a string of digits>)	Converts a string of digits to an integer value.
input(<a *string prompt*>)	Displays the string prompt and waits for keyboard input. Returns the string of characters entered by the user.
print(<expression>, ...,<expression>)	Evaluates the expressions and displays them, separated by one space, in the console window.
<string 1> + <string 2>	Glues the two strings together and returns the result.

Editing, Saving, and Running a Script (1 of 3)

- We can then run Python program files or **scripts** within IDLE or from the OS's command prompt
  - Run within IDLE using menu option, F5 (Windows), or Control+F5 (Mac or Linux)
- Python program files use **.py** extension
- Running a script from IDLE allows you to construct some complex programs, test them, and save them in **program libraries** to reuse or share with others

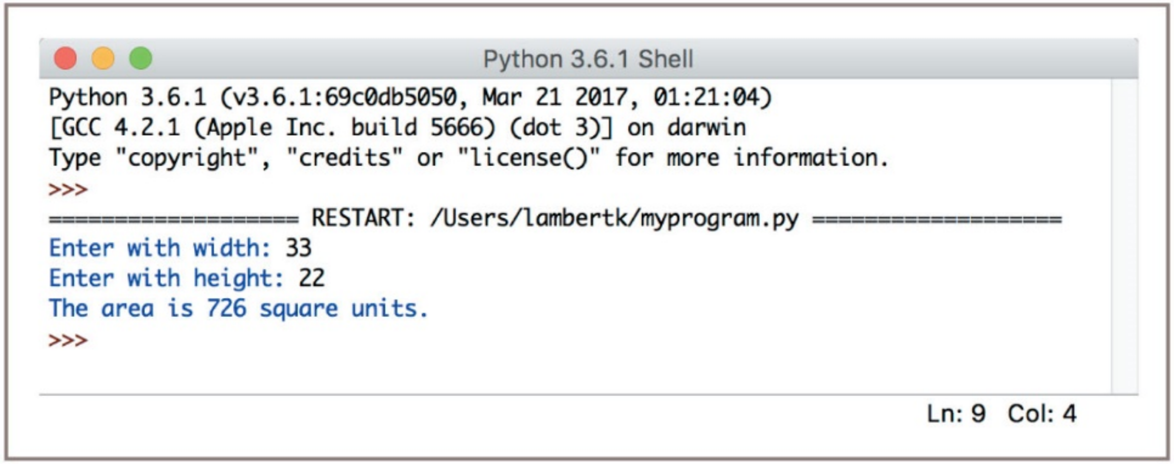
Editing, Saving, and Running a Script (2 of 3)



```
width = int(input("Enter with width: "))
height = int(input("Enter with height: "))
area = width * height
print("The area is", area, "square units.")
```

Figure 1-7 Python script in an IDLE window

Editing, Saving, and Running a Script (3 of 3)



```
Python 3.6.1 (v3.6.1:69c0db5050, Mar 21 2017, 01:21:04)
[GCC 4.2.1 (Apple Inc. build 5666) (dot 3)] on darwin
Type "copyright", "credits" or "license()" for more information.
>>>
=================== RESTART: /Users/lambertk/myprogram.py ===================
Enter with width: 33
Enter with height: 22
The area is 726 square units.
>>>
```

Figure 1-8 Interaction with a script in a shell window

Behind the Scenes: How Python Works

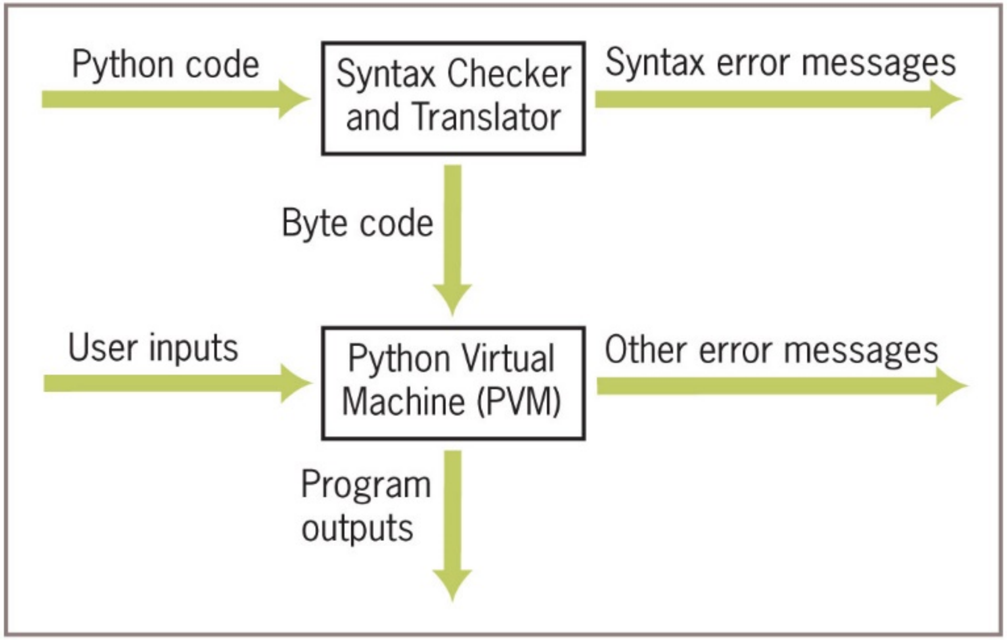


Figure 1-9 Steps in interpreting a Python program

Detecting and Correcting Syntax Errors (1 of 2)

- Programmers inevitably make typographical errors when editing programs, called **syntax errors**
  - The Python interpreter will usually detect these
- **Syntax:** rules for forming sentences in a language
- When Python encounters a syntax error in a program, it halts execution with an error message
- Example:

```
>>> length = int(input("Enter the length: "))
Enter the length: 44
>>> print(lenth)
Traceback (most recent call last):
File "<pyshell#l>", line 1, in <module>
NameError: name 'lenth' is not defined
```

Detecting and Correcting Syntax Errors (2 of 2)

- The next statement attempts to print the value of the correctly spelled variable:

```
>>> print(length)
SyntaxError: unexpected indent
```

- Final example, programmer attempts to add two numbers, but forgets to include the second one:

```
>>> 3 +
SyntaxError: invalid syntax
```

Chapter Summary (1 of 3)

- Fundamental ideas of computer science
  - The algorithm
  - Information processing
- Real computing agents can be constructed out of hardware devices
  - CPU, memory, and input and output devices
- Some real computers are specialized for a small set of tasks, whereas a desktop or laptop computer is a general-purpose problem-solving machine
- Software provides the means whereby different algorithms can be run on a general-purpose hardware device
  - Written in programming languages

Chapter Summary (2 of 3)

- Languages such as Python are high-level
- Interpreter translates a Python program to a lower-level form that can be executed on a real computer
- Python shell provides a command prompt for evaluating and viewing the results of Python expressions and statements
- IDLE is an integrated development environment that allows the programmer to save programs in files and load them into a shell for testing
- Python scripts are programs that are saved in files and run from a terminal command prompt

Chapter Summary (3 of 3)

- When a Python program is executed, it is translated into byte code
  - Sent to PVM for further interpretation and execution
- Syntax: set of rules for forming correct expressions and statements in a programming language