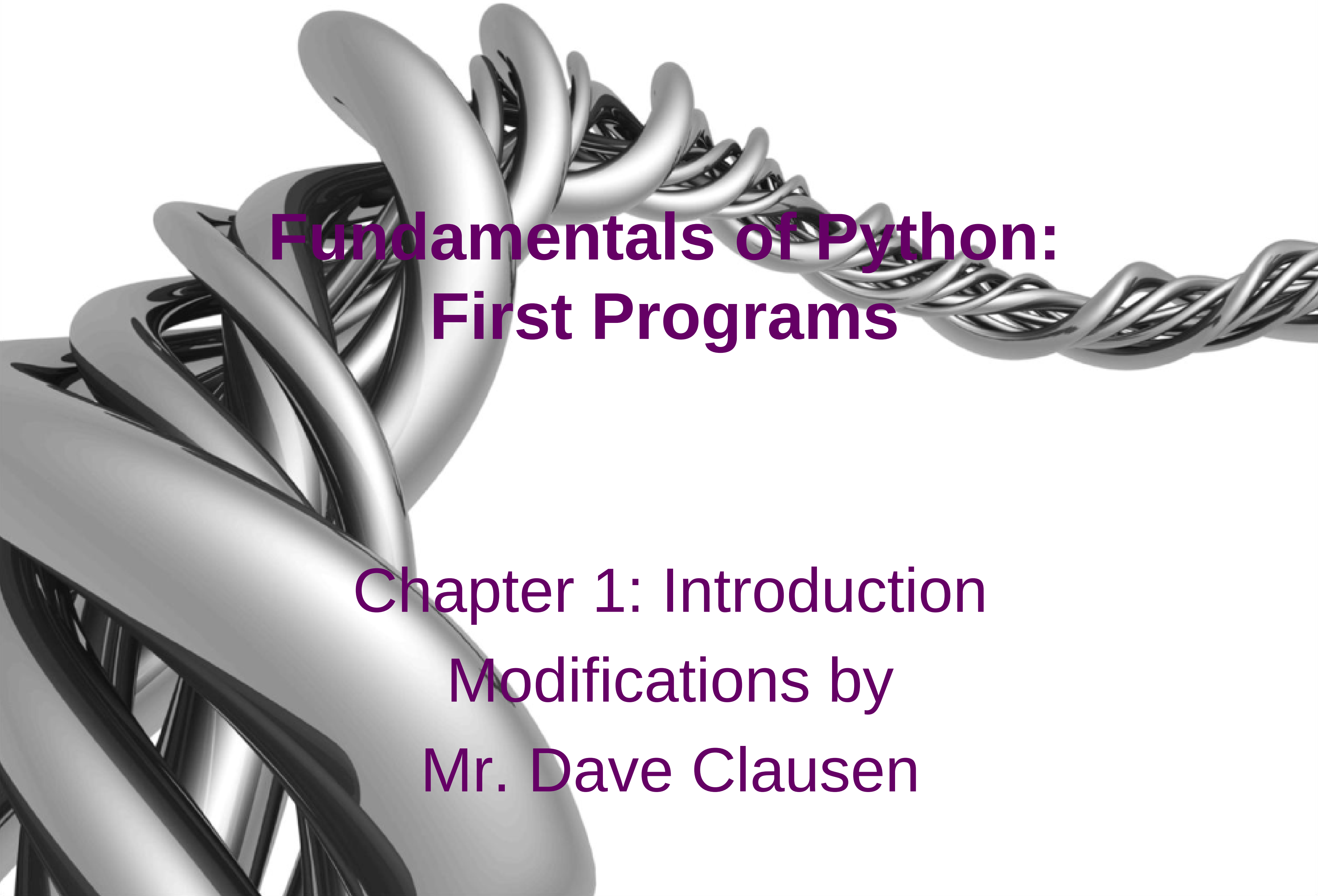


Fundamentals of Python: First Programs

Chapter 1: Introduction

Modifications by

Mr. Dave Clausen

Objectives

After completing this chapter, you 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

Fundamentals 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

- 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

Algorithms (continued)

- Sequence of steps that describes each of these computational processes is called an **algorithm**
- Features of an algorithm:
  - Consists of a finite number of instructions
  - Each individual instruction is well defined
  - Describes a process that eventually halts after arriving at a solution to a problem
  - Solves a general class of problems

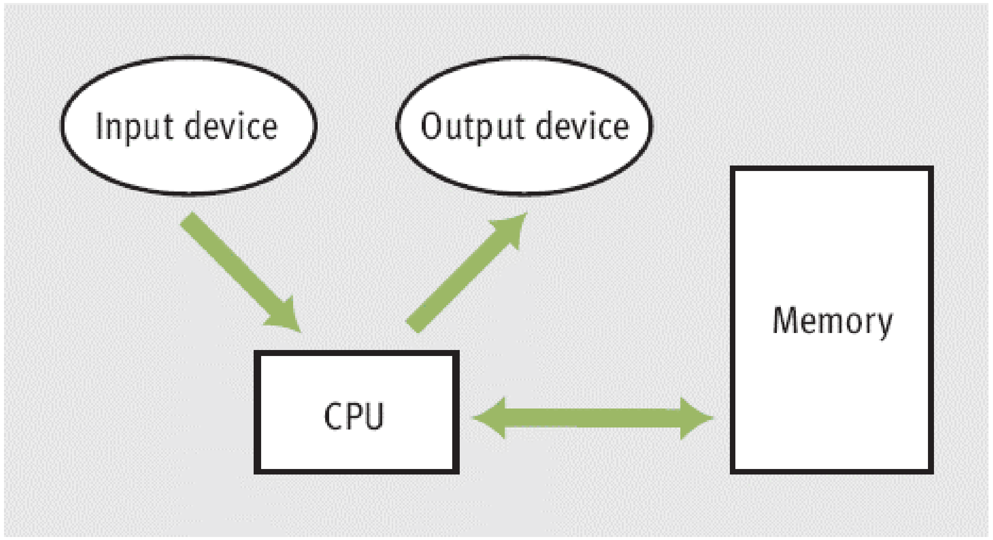
Information Processing

- Information is also commonly referred to as **data**
- In carrying out the instructions of an algorithm, computing agent manipulates information
  - Starts with **input** → produces **output**
- The algorithms that describe information processing can also be represented as information

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**

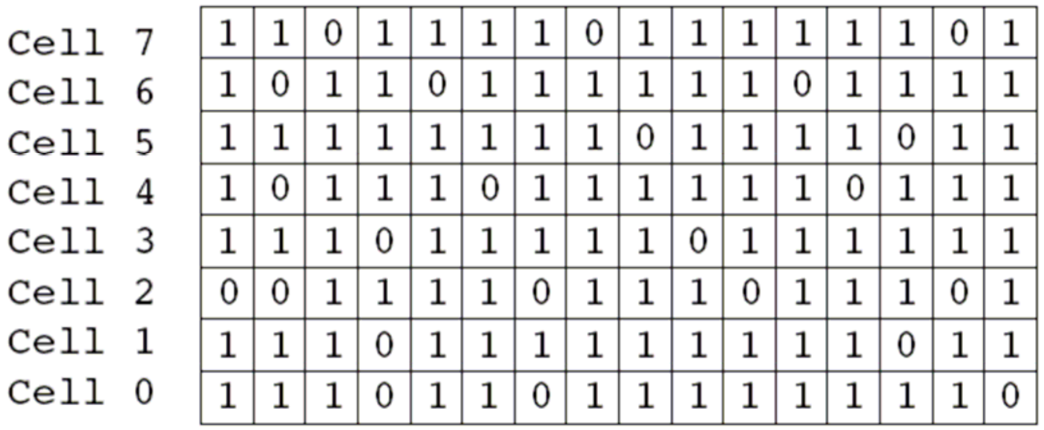
Computer Hardware



[FIGURE 1.1] Hardware components of a modern computer system

- Computers can also communicate with the external world through various **ports** that connect them to **networks** and to other devices

Computer Hardware (continued)



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

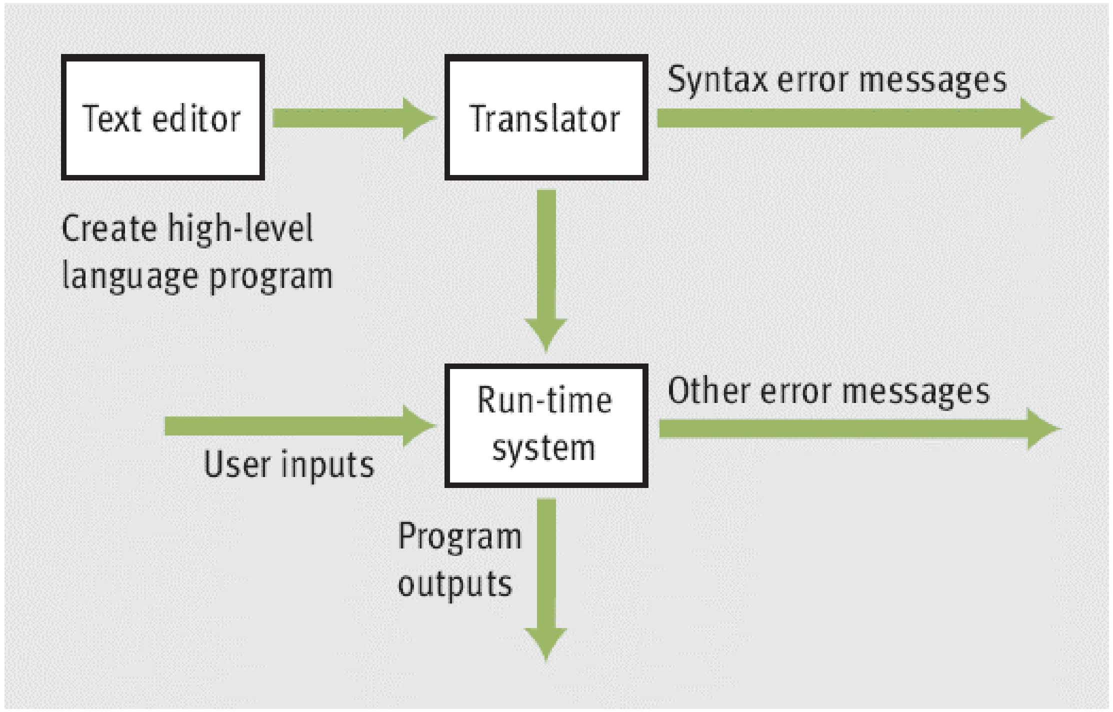
[FIGURE 1.2] A model of computer memory

- **Random access memory (RAM)** is also called **internal** or primary
- **External** or secondary memory can be **magnetic**, **semiconductor**, or **optical**

Computer Software

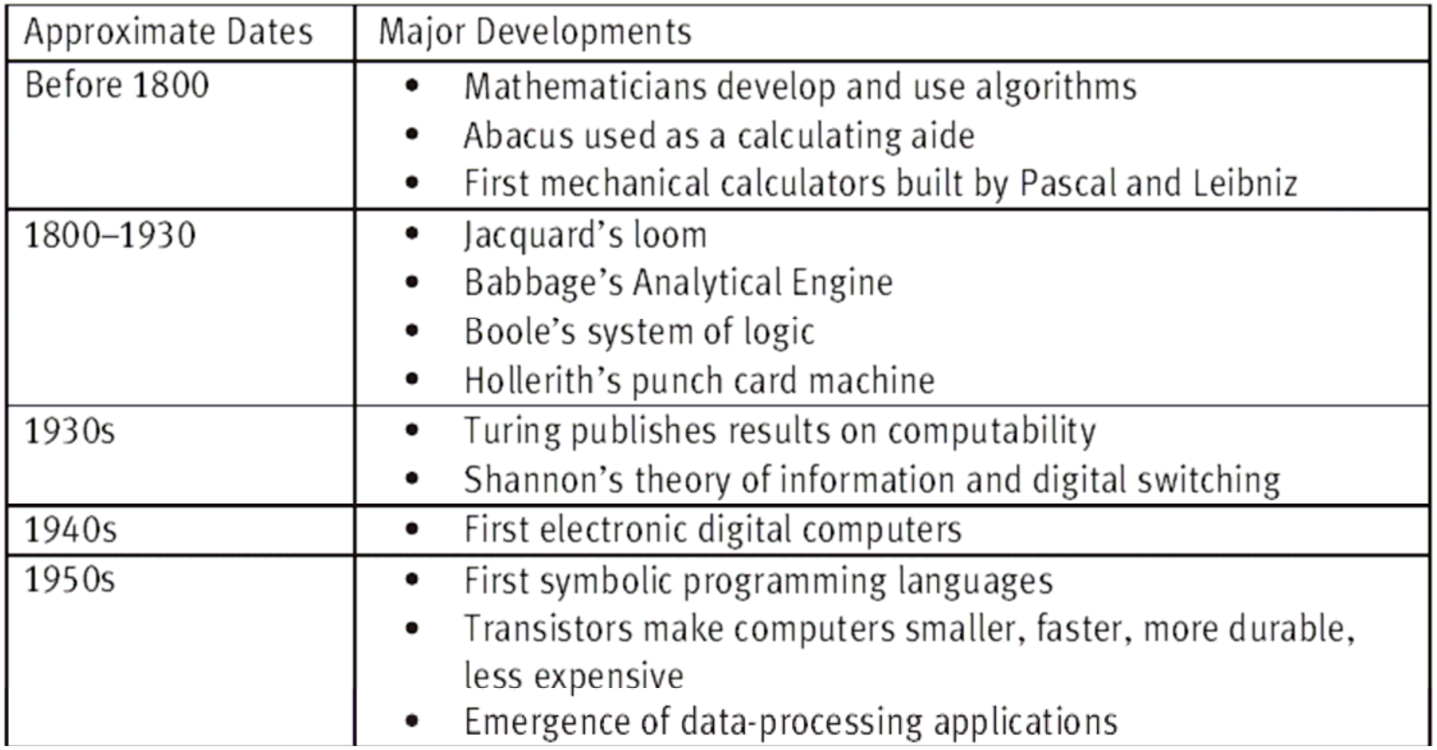
- A program stored in computer memory must be represented in binary digits, or **machine code**
- A **loader** takes a set of machine language instructions as input and loads them into the appropriate memory locations
- The most important example of **system software** is a computer's **operating system**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

Computer Software (continued)



[FIGURE 1.3] Software used in the coding process

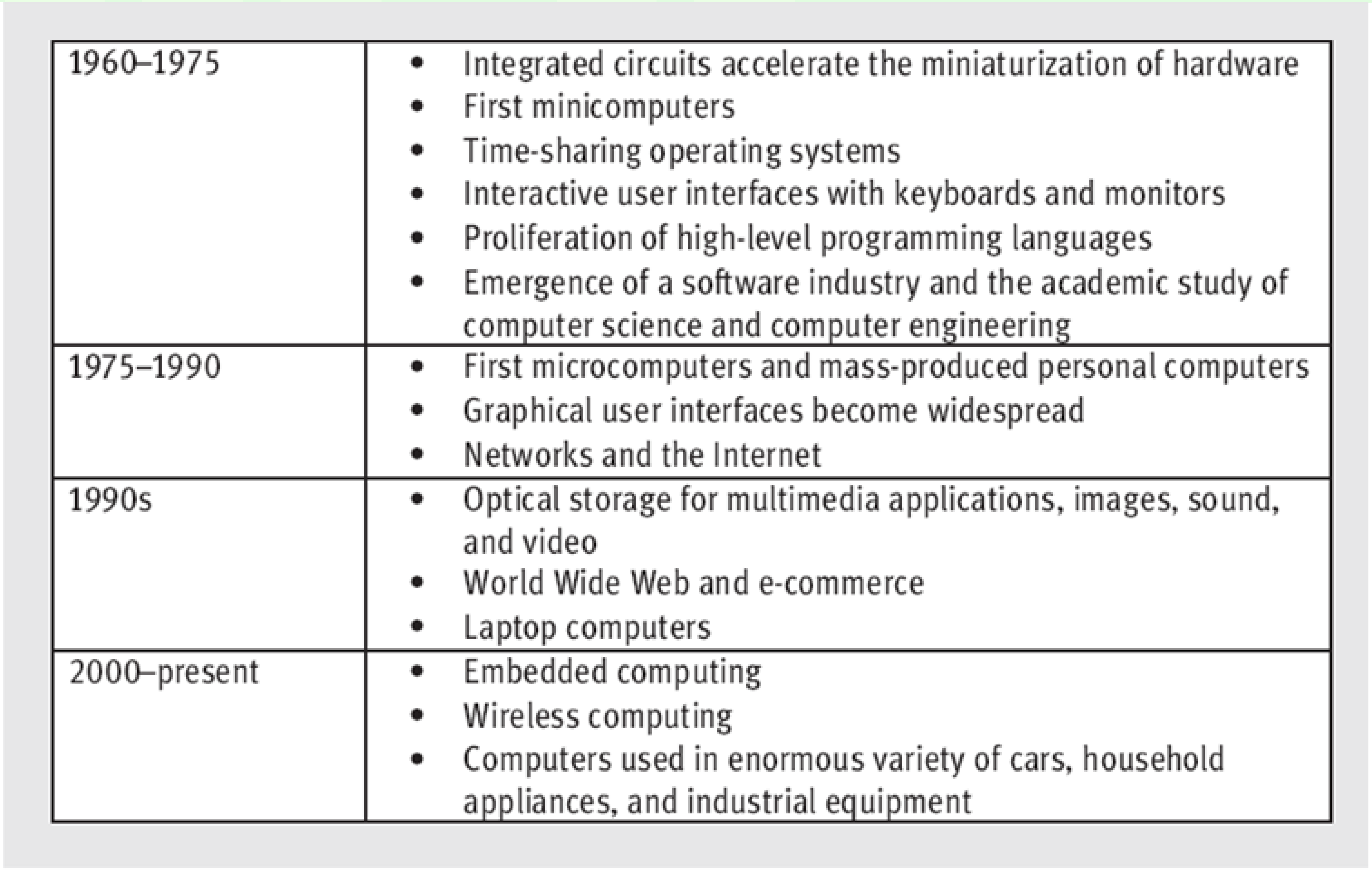
A Not-So-Brief History of Computing Systems



Approximate Dates	Major Developments
Before 1800	• Mathematicians develop and use algorithms • Abacus used as a calculating aide • First mechanical calculators built by Pascal and Leibniz
1800–1930	• 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, less expensive • Emergence of data-processing applications

[FIGURE 1.4] Summary of major developments in the history of computing

A Not-So-Brief History of Computing Systems (continued)



1960–1975	• Integrated circuits accelerate the miniaturization of hardware • First minicomputers • Time-sharing operating systems • Interactive user interfaces with keyboards and monitors • Proliferation of high-level programming languages • Emergence of a software industry and the academic study of computer science and computer engineering
1975–1990	• First microcomputers and mass-produced personal computers • Graphical user interfaces become widespread • Networks and the Internet
1990s	• Optical storage for multimedia applications, images, sound, and video • World Wide Web and e-commerce • Laptop computers
2000–present	• Embedded computing • Wireless computing • Computers used in enormous variety of cars, household appliances, and industrial equipment

[FIGURE 1.4] Summary of major developments in the history of computing

Before Electronic Digital Computers

- “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
- Joseph Jacquard (1752–1834): designed and constructed a machine that automated weaving
- Charles Babbage (1792–1871): conceived Analytical Engine

Before Electronic Digital Computers (continued)

- 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
- 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**

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 Ubiquitous Computing (1990–Present)

- **Optical storage media** developed for mass storage
- **Virtual reality:** capacity to create lifelike 3-D animations of whole-environments
- Computing is becoming ubiquitous, yet less visible
- Berners-Lee at CERN created WWW
  - Based on concepts of **hypermedia**
  - **HTTP**: Hypertext Transfer Protocol
  - **HTML**: Hypertext Markup Language

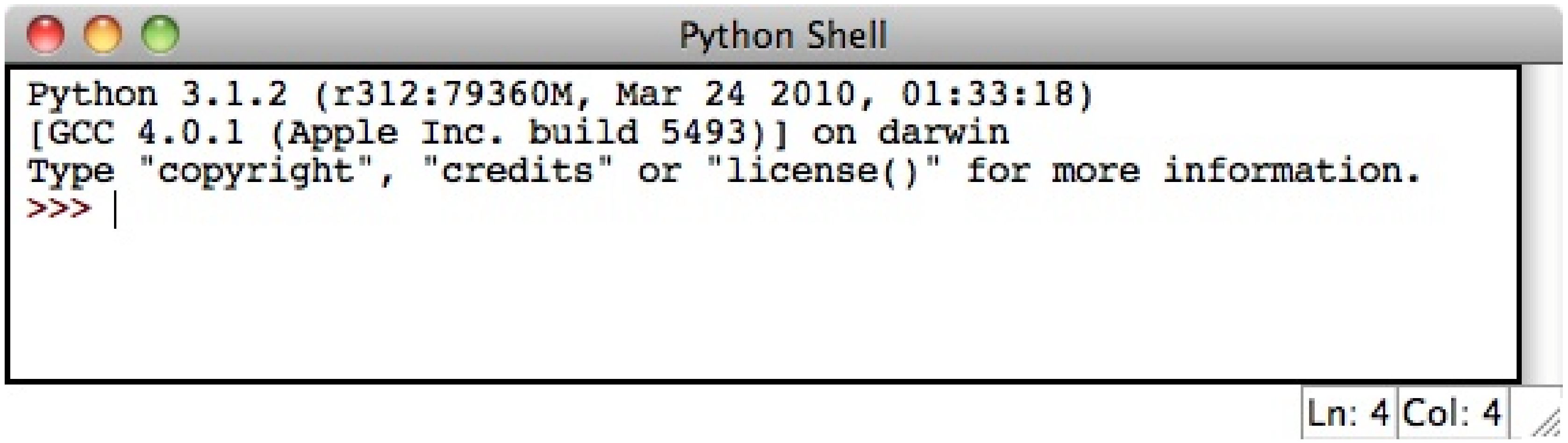
Getting Started with Python Programming

- 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

- 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 (continued)



```
Python 3.1.2 (r312:79360M, Mar 24 2010, 01:33:18)
[GCC 4.0.1 (Apple Inc. build 5493)] on darwin
Type "copyright", "credits" or "license()" for more information.
>>> 
```

[FIGURE 1.6] Python shell window

Input, Processing, and Output

- 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

```
print(<expression>)
```

```
>>> print('Hi there')
Hi there
```

```
print(<expression>, … , <expression>)
```

```
print(<expression>, end="")
```

Input, Processing, and Output (cont'd)

```
>>> name = input("Enter your name: ")
Enter your name: Ken Lambert
>>> name
'Ken Lambert'
>>> print(name)
Ken Lambert
>>>
```

```
<variable identifier> = input(<a string prompt>)
```

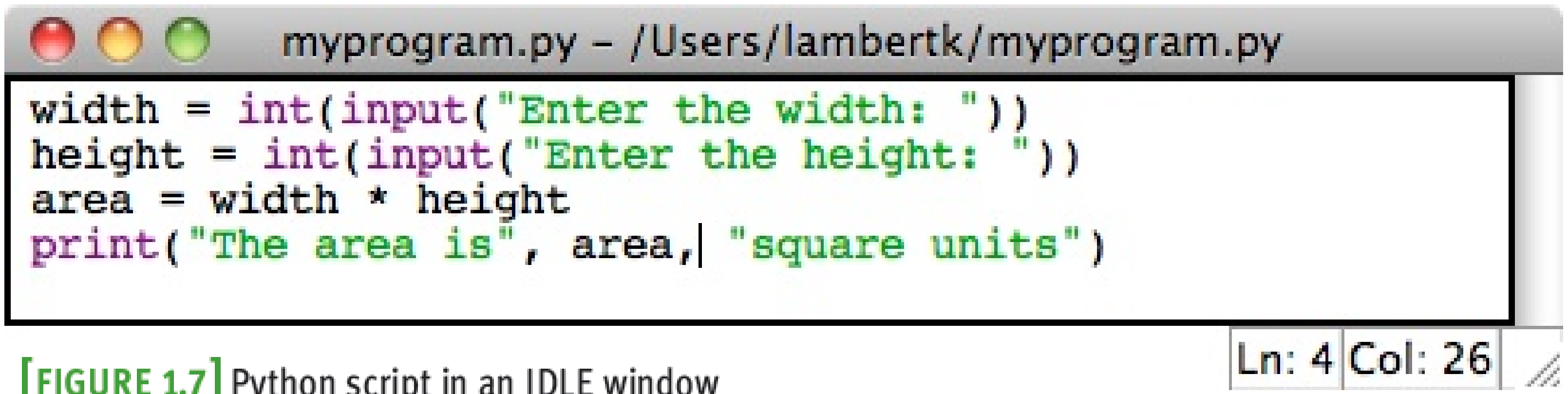
```
>>> name
'Ken Lambert'
```

```
>>> 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
>>>
```

Editing, Saving, and Running a Script

- Select **New Window** from the **File Menu**
- Type your Python Source Code.
- Use the **File Menu**, then **Save** using the **.py** extension
- We can then run Python program files or **scripts** within IDLE
  - **Run Menu**, then **Run Module** or press **F5** (Windows)
- 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

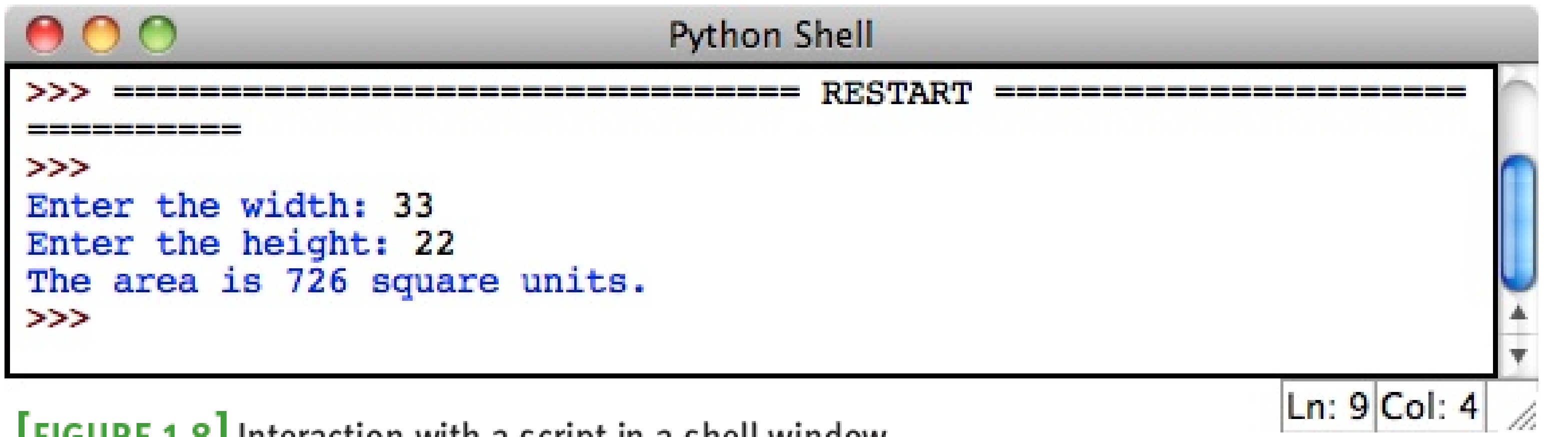
Editing, Saving, and Running a Script (continued)



```
width = int(input("Enter the width: "))
height = int(input("Enter the 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 (continued)

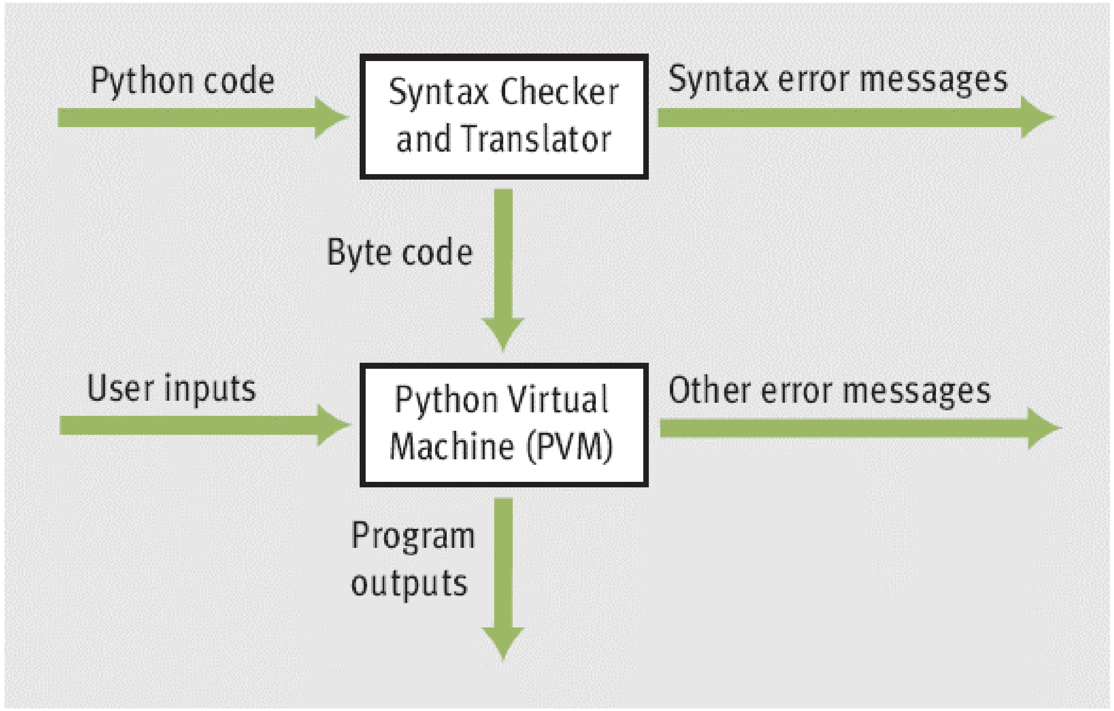


```
>>> ================================ RESTART ======================
===========
>>>
Enter the width: 33
Enter the height: 22
The area is 726 square units.
>>>
```

[FIGURE 1.8] Interaction with a script in a shell window

myprogram.py

Behind the Scenes: How Python Works



[FIGURE 1.9] Steps in interpreting a Python program

Detecting and Correcting Syntax Errors

- 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

Program Comments and Docstrings

- **Docstring** example:

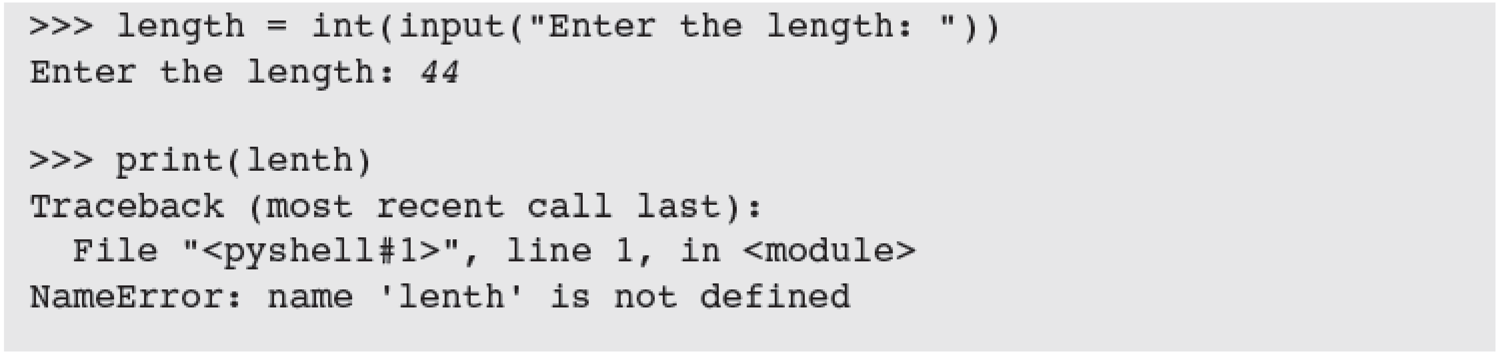
```
"""
Program: circle.py
Author: Ken Lambert
Last date modified: 2/10/11

The purpose of this program is to compute the area of a circle.
The input is an integer or floating-point number representing the
radius of the circle. The output is a floating-point number
labeled the area of the circle.
"""
```

- **End-of-line comment** example:

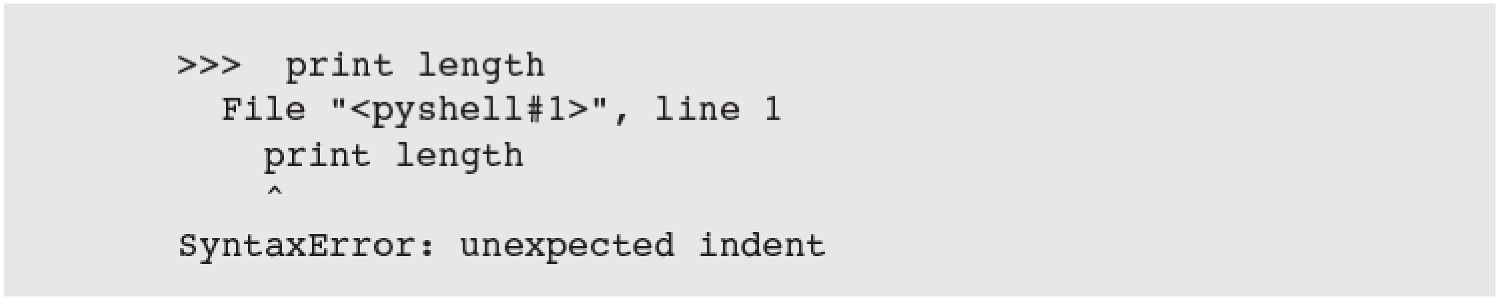
```
>>> RATE = 0.85    # Conversion rate for Canadian to US dollars
```

Detecting and Correcting Syntax Errors (continued)

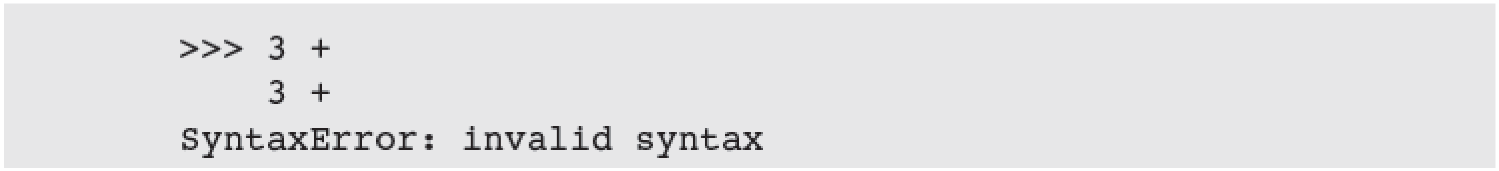


```
>>> length = int(input("Enter the length: "))
Enter the length: 44

>>> print(lenth)
Traceback (most recent call last):
  File "<pyshell#1>", line 1, in <module>
NameError: name 'lenth' is not defined
```



```
>>>  print length
  File "<pyshell#1>", line 1
    print length
    ^
SyntaxError: unexpected indent
```



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

Summary

- 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

Summary (continued)

- Software provides the means whereby different algorithms can be run on a general-purpose hardware device
  - Written in programming languages
- 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

Summary (continued)

- 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
- 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