# *Python Programming for Engineers and Scientists 1st edition Cengage Solutions Manual*

SKU: 9798214002446-SOLUTIONS

# Solution and Answer Guide

CENGAGE, PYTHON PROGRAMMING FOR SCIENTISTS AND ENGINEERS, 1E, ©2025, 9798214002446; CHAPTER 1, INTRODUCTION

## TABLE OF CONTENTS



## EXERCISE SOLUTIONS

### EXERCISE 1.1

1. List three common types of computing agents.

   **Solution**:

   Human beings, desktop computers, cell phones

2. Write an algorithm that describes the second part of the process of making change (counting out the coins and bills).

   **Solution:**

   There are various ways to do this, but here is one:

   Repeat
   Select the largest unit of money that is less than or equal to the remaining change
   Subtract this unit from the remaining change
   Until the remaining change is 0
   The collection of units selected represent the change

3. Write an algorithm that describes a common task, such as baking a cake.

   **Solution:**

There are various ways to do this, but here is one:

Preheat an oven to 375 degrees
Add 1 cup of water and 1 egg to a mixing bowl
Beat the liquid mixture in the bowl until the ingredients are blended
Add the contents of a boxed cake mix to the mixing bowl
Beat the mixture in the bowl until the ingredients are blended
Pour the contents of the mixing bowl into a lightly greased cake pan
Bake the cake in the oven for 45 minutes

4. Describe an instruction that is not well defined and thus could not be included as a step in an algorithm. Give an example of such an instruction.

   **Solution:**

   Attempting to divide a number by 0

5. In what sense is a laptop computer a general-purpose problem-solving machine?

   **Solution:**

   A laptop computer is a general-purpose problem-solving machine because it is programmable and can solve any problem for which there is an algorithm.

6. List four devices that use computers and describe the information that they process. (*Hint:* Think of the inputs and outputs of the devices.)

   **Solution:**
   Digital camera—images, music player—sound, cell phone—text, ATM—numbers

## EXERCISE 1.2

1. List two examples of input devices and two examples of output devices.

   **Solution:**
   Input devices—keyboard and mouse, output devices—monitor and speakers

2. What does the central processing unit (CPU) do?

   **Solution:**
   The CPU fetches, decodes, and executes instructions.

3. How is information represented in hardware memory?

   **Solution:**
   Information is represented using binary notation, which in hardware is a pattern of voltage levels.

4. What is the difference between a terminal-based interface and a graphical user interface?

   **Solution:**
   A terminal-based interface supports only the input and output of text with a keyboard and monitor. A graphical user interface supports the output of images and the manipulation of them with a pointing device, the mouse.

5. What role do translators play in the programming process?

   **Solution:**
   A translator converts a program written in a high-level language (human readable and writable) to an equivalent program in a low-level language (machine readable and executable).

## EXERCISE 1.3

1. Describe what happens when the programmer enters the string **"Greetings!"** in the Python shell.

   **Solution:**

   Python reads the string **`"Greetings!"`**, evaluates it, and displays this string (including single quotes) in the shell.

2. Write a line of code that prompts the user for their name and saves the user's input in a variable called **`name`**.

   **Solution:**
   ```
   name = input("Enter your name: ")
   ```

3. What is a Python script?

   **Solution:**

   A Python script is a complete Python program that can be run from a computer's operating system.

4. Explain what goes on behind the scenes when your computer runs a Python program.

   **Solution:**

   If the program has not already been translated, Python's compiler translates it to byte code. The Python virtual machine then executes this code.

## REVIEW QUESTIONS ANSWERS

1. Which of the following is an example of an algorithm?

   a. A dictionary

   b. A recipe

   c. A shopping list

   d. The spelling checker of a word processor

   **Answer:** b

   **Feedback:**

   a. **Incorrect**. A dictionary is a data structure.
   b. **Correct.** A recipe is a set of instructions that describes a process that halts with a solution to a problem.
   c. **Incorrect.** A shopping list is a data structure.
   d. **Incorrect.** A word processor is a program that consists of instructions and data.

2. Which of the following contains information?

   a. An audio CD

   b. A refrigerator

   c. An automobile

   d. A stereo speaker

   **Answer:** a

   **Feedback:**

   a. **Correct.** The information on an audio CD represents sound.
   b. **Incorrect.** A refrigerator contains just food, if it's not empty.
   c. **Incorrect.** An automobile contains a steering wheel, engine, exhaust pipe, and so on.
   d. **Incorrect.** A stereo speaker contains a magnetic coil, wires, and so on.

3. Which of the following is a general-purpose computing device?

a. A smartphone

b. A portable music player

c. A microwave oven

d. A programmable thermostat

**Answer**: a

**Feedback:**

a. **Correct.** A smartphone is capable of running any program.
b. **Incorrect.** A portable music player is dedicated to the specialized tasks of playing digital music.
c. **Correct.** A microwave is dedicated to the task of heating food.
d. **Incorrect.** A programmable thermostat is dedicated to the specialized tasks of climate control.

4. Which of the following is an input device?

a. Speaker

b. Microphone

c. Printer

d. Display screen

**Answer**: b

**Feedback:**

a. **Incorrect.** A speaker is a device for the output of sound.
b. **Correct.** A microphone is a device for the input of sound.
c. **Incorrect.** A printer is a device for the output of words or graphics on paper.
d. **Incorrect.** A display screen is a device for the output of images and text.

5. Which of the following are output devices?

a. A digital camera

b. A keyboard

c. A flatbed scanner

d. A monitor

**Answer:** d

**Feedback:**

a. **Incorrect.** A digital camera is a device for the input of images.

b. **Incorrect.** A keyboard is a device for the input of characters and commands via keystrokes.

c. **Incorrect.** A flatbed scanner is a device for the input of images from paper.

d. **Correct.** A monitor is a device for the visual display or output of words and graphics.

6. What is the purpose of the CPU?

   a. Store information

   b. Receive inputs from the human user

   c. Decode and execute instructions

   d. Send output to the human user

   **Answer:** c

   **Feedback:**

   a. **Incorrect.** The purpose of memory is to store information.

   b. **Incorrect.** The purpose of input devices, such as a keyboard, is to receive input from the human user.

   c. **Correct.** The CPU contains circuitry that decodes and executes instructions.

   d. **Incorrect.** The purpose of output devices, such as a monitor, is to send output to the human user.

7. Which of the following translates and executes instructions in a programming language?

   a. A compiler

   b. A text editor

   c. A loader

   d. An interpreter

   **Answer:** d

**Feedback:**

a. **Incorrect.** A compiler translates instructions in a source program to instructions in byte code or machine code.

b. **Incorrect.** A text editor allows a programmer to edit and save source code for a program.

c. **Incorrect**. A loader transfers instructions and data from external memory to random access memory at program startup.

d. **Correct.** An interpreter both translates (compiles) and executes (runs) instructions in a programming language.

8. Which of the following outputs data in a Python program?

   a. The input function

   b. The assignment statement

   c. The print function

   d. The main function

   **Answer:** c

   a. **Incorrect.** The input statement receives data from a file or at the keyboard and makes these data available to a running program.

   b. **Incorrect.** The assignment statement binds a variable to a value.

   c. **Correct.** The print statement sends data from a running program to a an output device, such as a monitor or a speaker.

   d. **Incorrect.** The main function organizes a program at the top level into a sequence of instructions.

9. What is IDLE used to do?

   a. Edit Python programs

   b. Save Python programs to files

c. Run Python programs

d. All of the other answers

**Answer**: a

**Feedback:**

a. **Correct.** IDLE can be used for all three tasks.

b. **Incorrect**. IDLE can be used to compile and run programs, too.

c. **Incorrect** IDLE can be used to edit and run programs, too.

d. **Incorrect** IDLE can be used to edit and compile programs, too.

10. What is the set of rules for forming sentences in a language called?

a. Semantics

b. Pragmatics

c. Syntax

d. Logic

**Answer: c**

**Feedback:**

a. **Incorrect.** Semantics is the set of rules for interpreting the meaning of sentences in a language.
b. **Incorrect**. Pragmatics is the set rules that describe the types of uses to which sentences in a language can be put.
c. **Correct.** Semantics is the set of rules for forming sentences in a language.
d. **Incorrect**. Logic is the set of rules for determining valid or invalid arguments in a language.

## Programming Exercises Solutions

### For all Programming Exercises Solutions, see solution files posted on the Instructor Companion Site.

1. Write a Python program in a file named **myinfo.py** that prints (displays) your name, address, and telephone number. (LO: 1.1)

   **Solution:** See `myinfo.py` posted on the Instructor Companion Site.

2. Open an IDLE window and enter the program from Figure 1-7 that computes the area of a rectangle. Save the program to a file named **rectangle.py** and load it into the shell by pressing the F5 key and correct any errors that occur. Test the program with different inputs by running it at least three times. (LO: 1.1)

   **Solution:** See `rectangle.py` posted on the Instructor Companion Site.

3. Write a program in a file named **triangle.py** to compute the area of a triangle. Issue the appropriate prompts for the triangle's base and height. Then, use the formula **`.5 * base * height`** to compute the area. Test the program from an IDLE window. (LO: 1.1)

   **Solution:** See `triangle.py` posted on the Instructor Companion Site.

4. Write and test a program in a file named **circle.py** that computes the area of a circle. This program should request a number representing a radius as input from the user. It should use the formula **`3.14 * radius ** 2`** to compute the area and then output this result suitably labeled. (LO: 1.1)

   **Solution:** See `circle.py` posted on the Instructor Companion Site.

5. A cuboid is a solid figure bounded by six rectangular faces. Its dimensions are its height, width, and depth. Write a Python program in a file named **cuboid.py** that computes and prints the volume of a cuboid, given its height, width, and depth as inputs. The volume is just the product of these three inputs. The output should be labeled as "cubic units."

   **Solution:** See `cuboid.py` posted on the Instructor Companion Site.

6. Laboratory Exercise: Setup an Initial Testcode. In this beginning assignment, the learner will setup their initial environment and produce some basic output. This will showcase the skills and understanding to setup a basic development environment, IDE and ability to produce a basic script with a beginning type of message.

   **Solution:** See `Chapter 1_Setup an Initial Testcode.py` posted on the Instructor Companion Site.

## DEBUGGING EXERCISES SOLUTIONS

1. Consider the following interaction at the Python shell:

```
>>> first = input("Enter the first integer: ")
Enter the first number: 23
>>> second = input("Enter the second integer: ")
Enter the second number: 44
>>> print("The sum is", first + second)
The sum of the two integers is 2344
```

The expected output is 67, but the output of this computation is 2344. Explain what causes this error and describe how to correct it.

**Solution**

The program prints the value 2344 when given the inputs 23 and 44. The output is the result of applying the **+** operator to the two the input strings “23” and “44” to produce the resulting string “2344”. To produce the arithmetic sum of the integers 23 and 44, one should convert the input strings “23” and “44” to integer values, using the type conversion function **int**. The revised statements

```
first = int(input("Enter the first integer: "))
second = int(input("Enter the second integer: "))
```

accomplish this. Then the + operator will perform an arithmetic sum on the two integers, producing the expected integer value of 67.

# Instructor Manual

Cengage, Python Programming for Scientists and Engineers 1e ©2025, 9798214002446; Chapter 1: Introduction

## TABLE OF CONTENTS

# Purpose and Perspective of the Chapter

The purpose of this chapter is to provide an introduction and background to learning Python programming. The basic features of an algorithm are described. 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.

## List of Student Downloads

Students should download the following items from the Student Companion Center to complete the activities and assignments related to this chapter:

- Data Files (provides files needed to complete the end of chapter Programming Exercises and Debugging Exercises)

# Chapter Objectives

The following objectives are addressed in this chapter:

01.01 Describe the basic features of an algorithm

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

01.03 Summarize a brief history of computing

01.04 Compose and run a simple Python program

# Complete List of Chapter Activities and Assessments

The following table organizes activities and assessments by objective, so that you can see how all this content relates to objectives and make decisions about which content you would like to emphasize in your class based on your objectives. For additional guidance, refer to the Teaching Online Guide.

| Chapter Objective | Activity/Assessment | Source (i.e., PPT slide, Workbook) | Duration |
|---|---|---|---|
| 01.01 – 01.06 | Chapter 1: Exercises | MindTap | 10-20 minutes |
| 01.01 – 01.06 | Chapter 1: Review Questions | MindTap | 10-20 minutes |

| | | | |
|---|---|---|---|
| 01.04 | Chapter 1: Programming Exercises | MindTap | 30-40 minutes per exercise |
| 01.01 – 01.06 | Chapter 1: Debugging Exercise | MindTap | 30-40 minutes |
| 01.01 – 01.06 | Chapter 1: HTML Activity - Basic I/O Example | MindTap | 5-10 minutes |
| 01.01 – 01.06 | Chapter 1: HTML Activity - Syntax Errors | MindTap | 5-10 minutes |
| 01.01 – 01.06 | Chapter 1: HTML Activity - Logic Errors | MindTap | 5-10 minutes |
| 01.01 | Activity 1.1: Discussion | PPT slide 8 | 15 minutes |
| 01.04 | Activity 1.2: Knowledge Check | PPT slides 40-41 | 5 minutes |
| Laboratory Exercise | Chapter 1: Setup an Initial Testcode | MindTap | 30-40 minutes |

[return to top]

## 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.

**applications software**: Programs that allow human users to accomplish specialized tasks, such as word processing or database management. Also called applications or apps.

**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.

**big data**: The gathering and analysis of massive amounts of data.

**binary digits**: 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* bits.

**bits**: Binary digits.

**bit-mapped display screen**: A type of display screen that supports the display of graphics and images.

**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.

**cathode ray tube (CRT) screen**: The first type of display device used to show computer output to users.

**central processing unit (CPU)**: A major hardware component that consists of the arithmetic/logic unit and the control unit. Also sometimes called a processor.

**client**: An agent that requests and receives some service.

**client/server application**: A type of application that allows many agents to receive service from one provider.

**cloud computing**: The use of server farms to provide data and applications to devices via wireless technology. *See also* server farm.

**compiler**: A computer program that automatically converts instructions in a high-level language to 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.

**data science**: A field of study that incorporates computer science and statistics to develop algorithms and data structures for data analysis.

**digital cloud**: A storage technology that allows users to store and access their data from remote locations, usually wirelessly.

**execute**: To carry out the instructions of a program.

**file system**: Software that organizes data on secondary storage media.

**graphical user interface (GUI)**: 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 component.

**hardware**: The computing machine and its support devices.

**high-level programming languages**: Programming languages whose vocabulary and sentence structure are fairly close to those of English.

**hypermedia**: A data structure that allows the user to access different kinds of information (text, images, sound, video, applications) by traversing links.

**information processing**: The transformation of one piece of information into another piece of information.

**input**: Data obtained by a program from the external world during execution.

**input/output devices**: Devices that allow information to be transmitted between the central processing unit of a computer and the external world.

**integrated circuit**: The arrangement of computer hardware components in a single, miniaturized unit.

**Internet of Things (IOT)**: The embedding of computer chips in physical objects, allowing them to send and receive digital information.

**interpreter**: A program that translates and executes another program.

**keypunch machine**: An early input device that allowed the user to enter programs and data onto punched cards.

**loader**: A software program that copies program code and data from secondary memory into primary memory before program execution begins.

**machine code**: The language used directly by the computer in all its calculations and processing.

**magnetic storage media**: Any media that allow data to be stored as patterns in a magnetic field.

**mainframe computers**: 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.

**microprocessor**: A processor that incorporates the entire central processing unit on a single integrated chip.

**Moore's Law**: A hypothesis that states that the processing speed and storage capacity of computers will increase by a factor of 2 every 18 months.

**network**: 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 and sent to the external world.

**personal digital assistant (PDA)**: A handheld device that allows the user to perform some simple tasks.

**ports**: Channels through which several clients can exchange data with the same server.

**primary memory**: A device that provides temporary storage for data and programs for fast access by a computer's central processing unit. *See also* random access memory.

**processor**: The hardware components that perform computation and control the flow of execution.

**program libraries**: Software tools or resources used in applications.

**programming language**: A formal language that computer scientists use to give instructions to the computer.

**program**: A set of instructions that tells the machine (the hardware) what to do.

**Python Shell**: An interactive program that allows the programmer to enter Python code and receive immediate feedback.

**Python virtual machine (PVM)**: A program that interprets Python byte codes and executes them.

**random access memory (RAM)**: Memory where a program and data are loaded for execution. Same as primary memory.

**run-time system**: Software that supports the execution of a program.

**secondary memory**: A device such as a hard drive or flash stick where data can be backed up or stored permanently.

**semiconductor storage media**: Devices, such as flash sticks, that use solid state circuitry to store data permanently.

**server**: An agent that receives requests and provides a service.

**server farms**: A collection of computers which can store massive amounts of data for many users.

**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 engineering**: The construction of large software systems using a disciplined method of requirements analysis, design, coding, and testing that involves the coordination of a team of specialized developers.

**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.

**string**: A sequence of zero or more characters enclosed in quote marks.

**syntax**: The form or structure of a sentence in a programming language.

**syntax errors**: Errors in spelling, punctuation, or placement of certain key symbols in a program. *See also* design error.

**system software**: The programs that allow users to write and execute other programs, including operating systems such as Windows and macOS.

**terminal-based interface**: A user interface that allows the user to enter input from a keyboard and view output as text in a window.

**text editor**: A program that allows the user to enter text, such as a program, and save it in a file.

**touchscreen interface**: A user interface that allows the user to enter input by tapping or gesturing while touching its screen.

**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 function**: A function that takes one type of data as an argument and returns the same data represented in another type.

**user interfaces**: Software and hardware devices that present information to human users and receive input data or commands from them.

**variable identifier**: A name that refers to a memory location, whose value can be changed during execution of a program.

**virtual assistant**: A computing device that responds to a user's commands, usually via voice.

**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 application**: A program that runs on a remote server but uses clients' web browsers to deliver them services.

**web browser**: Software that makes requests for web pages, receives them from a web server, and renders them on a display.

**web client**: Software on a computer that makes requests for resources and receives them from the web.

**web servers**: Software on a computer that responds to requests for resources and makes them available on the web.

[return to top]

## What's New in This Chapter

The following elements are improvements in this chapter from the previous edition:

New and Expanded Topics:

- Updated the section on the history of computing.

Global:

- New list of Key Terms in each chapter
- Updated and new Case Studies, Review Questions and Programming Exercises
- New Debugging Exercises in each chapter provide examples of challenging programming errors and experience in diagnosing and correcting them.
- Text revisions throughout with a focus on readability
- Update to Python 3
- A-heads and Programming Exercises mapped to chapter learning objectives
- Suggestions for Further Reading moved to Appendix E

[return to top]

## CHAPTER OUTLINE

The following outline organizes activities (including any existing discussion questions in PowerPoints or other supplements) and assessments by chapter (and therefore by topic), so that you can see how all the content relates to the topics covered in the text.

I. Two Fundamental Ideas of Computer Science: Algorithms and Information Processing (Objective 01.01, PPT Slides 4-8)
    a. Computer science focuses on a broad set of interrelated ideas. Two of the most basic ones are algorithms and information processing.
    b. Algorithms
        I. The following is an example of an algorithm for subtracting two numbers:
            (1) Step 1: Write down the numbers, larger above smaller one, digits column-aligned from right.
            (2) Step 2: Start with rightmost column of digits and work your way left through the various columns.
            (3) 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.
            (4) Step 4: If there is no next column to the left, stop.
            (5) Otherwise, move to the next column to the left; go to Step 3.

II. In this example, the computing agent is a human being. The sequence of steps that describes each of these computational processes is called an algorithm.
III. Computers can be designed to run a small set of algorithms for performing specialized tasks. An algorithm:
(1) Consists of a finite number of instructions.
(2) Includes well-defined individual instructions. Each action described by the instruction can be performed effectively or be executed by a computing agent.
(3) Describes a process that eventually halts after arriving at a solution to a problem.
(4) Solves a general class of problems.

c. Information Processing
I. Information is also commonly referred to as data.
II. In carrying out the instructions of an algorithm, the computing agent manipulates information. It starts with input, transforms information according to well-defined rules, and produces output.
III. 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.

**d. Activity 1.1: Discussion (01.01, PPT Slide 8)**
I. Imagine that you want to design an algorithm to develop your course schedule for next term.
(1) Which part(s) of your algorithm would be most straightforward to design? Why?
(2) Which part(s) of your algorithm would be most difficult to design? Why?

II. The Structure of a Modern Computer System (Objective 01.02, PPT Slides 9-15)
a. A modern computer system consists of hardware and software.
I. Hardware refers to physical devices required to execute algorithms.
II. Software refers to a set of algorithms, represented as programs in particular programming languages.
b. Basic hardware components of a computer are memory, a central processing unit (CPU), and a set of input/output devices.
I. Computers can also communicate with the external world through various ports that connect them to networks and to other devices.

II. Computer memory is set up to represent and store information in electronic form. Information is stored as patterns of binary digits (1s and 0s).
    (1) Random access memory (RAM) is also called primary or internal memory.
    (2) The part of a computer responsible for processing data is the central processing unit (CPU), also called the processor.
    (3) Secondary or external memory can be magnetic, semiconductor, or optical.

c. A computer software program stored in computer memory must be represented in binary digits, or machine code.
    I. A loader takes a set of machine language instructions as input and loads them into the appropriate memory locations.
    II. The most important example of system software is a computer's operating system. Two important parts of this are the file system and user interfaces (terminal-based interfaces, GUIs, or touchscreen interfaces).
    III. Software applications include Web browsers, word processors, spreadsheets, database managers, graphic design packages, and games.
    IV. Scientists have developed high-level programming languages for expressing algorithms. These resemble English and allow the author to express algorithms in a form that other people can understand.
        (1) Programmers usually start by writing high-level language statements in a text editor. They run another program called a translator to convert program code into executable code. The translator checks for syntax errors.
        (2) If no errors are found, the program can be executed by the run-time system. It might execute the program directly on the hardware or run another program called an interpreter or virtual machine to execute the program.

III. A Not-So-Brief History of Computing Systems (Objective 01.03, PPT Slides 16-28)

a. Before Electronic Digital Computers
    I. The term "algorithm" came from Muhammad ibn Musa Al-Khawarizmi, a Persian mathematician.
    II. Euclid developed an algorithm for computing the greatest common divisor of two numbers.

III. The abacus also appeared in ancient times.
IV. Blaise Pascal (1623–1662) built one of the first mechanical devices to automate addition.
V. Wilhelm Leibniz (1646-1716) built another calculator that included other arithmetic functions such as multiplication.
VI. Joseph Jacquard (1752–1834) designed and constructed a machine that automated weaving.
VII. Charles Babbage (1792–1871) conceived the Analytical Engine.
VIII. Herman Hollerith (1860–1929) developed a machine that automated data processing for the U.S. Census. He was one of the founders of company that became IBM.
IX. George Boole (1815–1864) developed Boolean logic.
X. Alan Turing (1912–1954) explored the theoretical foundations and limits of algorithms and computation.

b. Human beings were used as computers between 1940 and 1945.
   I. The needs of combatants in World War II expanded the practical applications of computing, specifically in the areas of cryptanalysis and ballistics calculations.
   II. Since no machines were available to automate these tasks, they were performed manually by humans, primarily women highly skilled in mathematics and puzzle solving. These humans were known as “computers.” Major breakthroughs were achieved by Agnes Meyer Driscoll and Genevieve Grotjan Feinstein.
c. The first electronic digital computers were born between 1940 and 1950.
   I. In the late 1930s, Claude Shannon wrote paper titled “A Symbolic Analysis of Relay and Switching Circuits.”
   II. In the 1940s the following computers were built: the Mark I (electromechanical), the ENIAC (Electronic Numerical Integrator and Calculator), the ABC (Atanasoff-Berry Computer), and the Colossus, created by a group working under Alan Turing. John von Neumann developed the first memory-stored programs.
   III. These computers, sometimes called mainframe computers, consisted of vacuum tubes, wires, and plugs, and filled entire rooms.
d. The first programming languages were developed between 1950 and 1965.
   I. The first assembly languages had operations like ADD and OUTPUT.
   II. Programmers entered mnemonic codes for operations at a keypunch machine.

III. A card reader translated holes in cards to patterns in the computer's memory.
IV. An assembler translated application programs in memory to machine code.
V. A compiler translated programs to machine code.
VI. High-level programming languages developed during this time included FORTRAN, LISP, and COBOL, which all shared the common feature of abstraction.

e. The years 1965-1975 were characterized by integrated circuits, interaction, time-sharing, and software engineering.
   I. In the late 1950s the vacuum tube gave way to the transistor.
   II. In the early 1960s, the integrated circuit enabled smaller, faster, less expensive hardware components. This led to the formulation of Moore's Law, which stated that the processing speed and storage capacity of hardware will increase and cost will decrease by approximately a factor of 2 every 18 months.
   III. Minicomputers appeared.
   IV. Processing evolved from batch processing to time-sharing to concurrent processing.
   V. Interactive, multiuser computer systems could now support the development of large, complex software applications by teams of programmers. Margaret Hamilton coined the term *software engineering*.
f. The period from 1975-1990 was characterized by personal computing and networks.
   I. In the late 1960s, Douglas Engelbart developed the first pointing device (mouse) and software to represent windows, icons, and pull-down menus on a bit-mapped display screen. He was a member of the team that developed Alto (Xerox PARC).
   II. In 1975, Altair, the first mass-produced personal computer, was developed. It contained Intel's 8080 processor, the first microcomputer chip.
   III. In the early 1980s, Gates and Allen built MS-DOS.
   IV. Bob Metcalfe created Ethernet, used in LANs.
   V. ARPANET grew into what we call the Internet.
g. The period from 1990-2000 was the era of consultation, communication, and e-commerce.
   I. Optical storage media was developed for mass storage.

II. Berners-Lee at CERN created the World Wide Web (WWW), which was based on concepts of hypermedia, including HTTP (Hypertext Transfer Protocol) and HTML (Hypertext Markup Language).
III. The components of the WWW are web servers, web browsers, and web clients.
IV. Web applications presented a revolution in the way software services were delivered to people. They made online stores pervasive. The web application providing the service ran on a remote computer or server.
V. Client/server applications, such as e-mail, bulletin boards, and chat rooms, were already in use. They were simply deployed on the web when it became available.
VI. Sergey Brin and Larry Page developed algorithms for indexing and searching the web.

h. The period from 2000 to the present is the era of mobile applications and ubiquitous computing.
I. Personal digital assistants (PDAs) were the first handheld computing devices, limited to video games, address books, to-do lists, and note taking.
II. Steve Jobs (Apple) created several key devices: the iPod, iPhone, and iPad.
III. Social networking applications were a major addition to the digital landscape.
IV. Cloud computing and wireless technology underlie an Internet of Things (IOT).
V. Big data technology has emerged: governments, businesses, and hackers continually monitor Internet traffic. Researchers have created algorithms that process massive amounts of data to discover trends and predict outcomes.

IV. Getting Started with Python Programming (Objective 01.04, PPT Slides 29-44)
a. In the early 1990s, Guido van Rossum invented the Python programming language.
I. Python is a high-level, general-purpose programming language for solving problems on modern computer systems.
II. Useful resources can be found at www.python.org.
b. Python is an interpreted language.
I. Simple Python code (expressions and statements) can be run in the interactive shell.
(1) The easiest way to open a Python shell is to launch the IDLE.

(2) To quit, select the window's close box or press Control+D.
(3) The shell is useful for experimenting with short expressions or statements and consulting the documentation.

c. Programs usually accept inputs from a source, process them, and output results to a destination.
   I. In terminal-based interactive programs, these are the keyboard and terminal display.
   II. In Python, inputs are Python expressions or statements. Outputs are the results displayed in the shell.
   III. Programmers can also force output of a value by using the `print` function, in the format `print (<expression>)`. For example:
```
>>>print ("Hi there")
Hi there
```
   IV. 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
>>>
```
   V. The `input` function always builds a string from the user's keystrokes and returns it to the program.
   VI. Strings that represent numbers must be converted from strings to appropriate number types. Two type conversion functions are `int` (for integers) and `float` (for floating-point numbers).
   VII. This 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
```

d. We can run Python program files or scripts within IDLE or from the OS's command prompt.
   I. We can run within IDLE using the menu option, F5 (Windows), or Control+F5 (Mac or Linux).
   II. Python program files use the .py extension.

III. 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.

e. **Activity 1.2: Knowledge Check (01.04, PPT Slides 41-42)**
   I. What does the `input` function do?
   Answer: The `input` function builds a string from the user's keystrokes and returns it to the program.
   II. Name the two functions that convert strings to number types.
   Answer: `int` (for integers) and `float` (for floating-point numbers)
   III. What does the `print` function do?
   Answer: The `print` function evaluates expressions and displays them, separated by one space, in the console window.

f. The Python interpreter carries out the following steps to execute the instructions in a program:
   I. A syntax checker and translator reads source code
      (1) If an expression/statement is not well-formed, outputs an error message
      (2) If an expression/statement is well-formed, translates it to byte code
   II. The Python virtual machine (PVM) executes the byte code

g. Programmers inevitably make typographical errors when editing programs, called syntax errors. Syntax refers to rules for forming sentences in a language.
   I. The following is an example of a syntax error:

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

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

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

   III. In this final example, the programmer attempts to add two numbers, but forgets to include the second one:

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

h. **Self-Assessment (01.01-01.04, PPT Slide 45)**

I. What part of the evolution of computing systems throughout the past 400 years was most surprising or interesting to you? Which would you like to learn more about?
II. What are your initial thoughts on developing Python programs?

[return to top]

## Suggested Usage for Lab Activities

Coding Labs, graded (in the MindTap Learning Path):

1. Chapter 1: Programming Exercise 1-1 (LO: 1.4)
2. Chapter 1: Programming Exercise 1-2 (LO: 1.4)
3. Chapter 1: Programming Exercise 1-3 (LO: 1.4)
4. Chapter 1: Programming Exercise 1-4 (LO: 1.4)
5. Chapter 1: Programming Exercise 1-5 (LO: 1.4)
6. Chapter 1: Debugging Exercise

## Additional Activities and Assignments

The following are activities and assignments developed by Cengage but not included in the text, PPTs, or courseware (if courseware exists) – they are for you to use if you wish.

1. **Quick Quiz 1**: A quick, multiple-choice quizzing activity covering L.O.s 1.2-1.3

    1. Which part of a computer is responsible for processing data?
        a. memory
        b. hard disk drive
        c. central processing unit
        d. file system
        Answer: C

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

    3. Which of the following organizes the monitor screen around the metaphor of a desktop, with windows containing icons for folders, files, and applications?
        a. cathode ray tube
        b. terminal
        c. web server

d. graphical user interface
Answer: D

4. In the late-1930s, which mathematician and electrical engineer at M.I.T., wrote a classic paper titled “A Symbolic Analysis of Relay and Switching Circuits?”
   a. Alan Turing
   b. Claude Shannon
   c. George Boole
   d. Howard Aiken
   Answer: B

5. By the mid-1980s, which entity had grown into what we now call the Internet, connecting computers owned by large institutions, small organizations, and individuals all over the world?
   a. ARPANET
   b. FORTRAN
   c. WAN
   d. UNIX
   Answer: A

2. **Quick Quiz 2**: A quick, multiple-choice quizzing activity covering L.O. 1.4

   1. Which part of a computer is responsible for processing data?
      a. memory
      b. hard disk drive
      c. central processing unit
      d. file system
      Answer: C

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

   3. Which of the following organizes the monitor screen around the metaphor of a desktop, with windows containing icons for folders, files, and applications?
      a. cathode ray tube
      b. terminal
      c. web server
      d. graphical user interface
      Answer: D

6. In the late-1930s, which mathematician and electrical engineer at M.I.T., wrote a classic paper titled “A Symbolic Analysis of Relay and Switching Circuits?”
   a. Alan Turing
   b. Claude Shannon
   c. George Boole
   d. Howard Aiken
   Answer: B

7. By the mid-1980s, which entity had grown into what we now call the Internet, connecting computers owned by large institutions, small organizations, and individuals all over the world?
   a. ARPANET
   b. FORTRAN
   c. WAN
   d. UNIX
   Answer: A

3. **Class Discussion Topics**: Related topics of interest to encourage group conversation (L.O. 1.4)

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

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

   3. 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?

4. **Additional Projects**: Possible assignments/projects for students to apply and further explore chapter content

   1. 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. (L.O. 1.1)

   2. 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. (L.O. 1.3)

[return to top]

## ADDITIONAL RESOURCES

### CENGAGE VIDEO RESOURCES

- MindTap Videos:
  - What is Computer Programming?

### INTERNET RESOURCES

- Home page for the Python language: **Python**
- Algorithms: **What is an Algorithm?**
- History of algorithms: **History of Algorithms and Algorithmics**
- History of computer software: **Timeline of Computer History**
- History of computer languages and their evolution: **History of Programming Languages**
- Evolution of computer languages: **The Evolution of Computer Languages (Infographic)**
- A graph that illustrates Moore's Law: **Understanding Moore's Law**
- Python programming tutorial: **The Python Tutorial**

[return to top]

# APPENDIX

## GENERIC RUBRICS

Providing students with rubrics helps them understand expectations and components of assignments. Rubrics help students become more aware of their learning process and progress, and they improve students' work through timely and detailed feedback. Customize these rubrics as you wish.

### STANDARD DISCUSSION RUBRIC

| Criteria | Meets Requirements | Needs Improvement | Incomplete |
|---|---|---|---|
| Participation | Submits or participates in discussion by the posted deadlines. Follows all assignment. instructions for initial post and responses. 5 points | Does not participate or submit discussion by the posted deadlines. Does not follow instructions for initial post and responses. 3 points | Does not participate in discussion. 0 points |
| Contribution Quality | Comments stay on task. Comments add value to discussion topic. Comments motivate other students to respond. 20 points | Comments may not stay on task. Comments may not add value to discussion topic. Comments may not motivate other students to respond. 10 points | Does not participate in discussion. 0 points |
| Etiquette | Maintains appropriate language. Offers criticism in a constructive manner. Provides both positive and negative feedback. 5 points | Does not always maintain appropriate language. Offers criticism in an offensive manner. Provides only negative feedback. 3 points | Does not participate in discussion. 0 points |

### STANDARD COLLABORATION RUBRIC

| Criteria | Meets Requirements | Needs Improvement | Incomplete |
|---|---|---|---|
| Share knowledge | Consistently and actively contributes knowledge, opinions, and skills. 5 points | Contributes to the group only when prompted. 3 points | Does not contribute to the group. 0 points |
| Adjust to unforeseen circumstances | Seeks different solutions, approaches, and strategies in an effective, original, and/or creative way. 5 points | Seeks a single solution, approach, or strategy, but does not pursue better or original alternatives. 3 points | Relies on the first solution generated and does not offer other solutions. 0 points |

| | | | |
|---|---|---|---|
| Make decisions | Helps the group identify necessary changes and encourages group action for change.<br>10 points | Participates in needed changes only when prompted.<br>5 points | Does not participate in decision making.<br>0 points |
| Build consensus | Values, encourages, and acknowledges the work of other group members.<br><br>Takes responsibility for the assignment that reflects minority and majority conclusions of the group.<br>10 points | Listens attentively to members of the group but contributes little to the group and assignment.<br>5 points | Does not participate in building consensus within the group.<br>0 points |
| Complete deliverables (only if applicable) | Completes all assigned components or deliverables with quality, creativity, and accuracy.<br><br>Turns in all assigned components by agreed upon deadlines without reminders.<br>10 points | Completes all assigned components with bare minimum of quality and creativity.<br><br>Requires reminders and follow-up from team members in order to complete deliverables.<br>5 points | Does not complete assigned components or deliverables or turns them in late.<br><br>Assigned components are poorly executed, inaccurate, or not functioning.<br><br>Does not respond to communication from team.<br>0 points |

## Programming Rubrics

Providing students with programming rubrics helps them understand expectations and components of writing code. Rubrics help students become more aware of their learning process and progress, and they improve students' work through timely and detailed feedback. Customize these rubrics as you wish.

## Standard Programming Rubric (Simple)

| Criteria | Excellent | Good | Fair | Poor | None |
|---|---|---|---|---|---|
| **Function:**<br>The code works without errors and fulfills all project requirements.<br>Test cases for in- and out-of-range inputs are successful.<br>Error handling is used correctly, and where necessary and appropriate. | 4 | 3 | 2 | 1 | 0 |

| | | | | | |
|---|---|---|---|---|---|
| **Formatting:**<br>The code is readable, formatted properly with indents, and line spacing.<br>Variables are named appropriately and consistently.<br>The code is well-commented. | 4 | 3 | 2 | 1 | 0 |
| **Simplicity:**<br>The code efficiently fulfills the project requirements.<br>Hardcoded values are minimized.<br>Complex programming logic is distributed in short, single-purpose lines of code. | 4 | 3 | 2 | 1 | 0 |
| **Modularity:**<br>Programming follows the Don't Repeat Yourself best practice.<br>Each method/function performs one discrete task.<br>Variables are strongly typed and used for singular, discrete purposes. | 4 | 3 | 2 | 1 | 0 |
| **Comprehension:**<br>The code demonstrates the student understands the concepts required.<br>The student can explain how the program works.<br>The student can explain why alternative solutions are not as good as the one(s) used. | 4 | 3 | 2 | 1 | 0 |

## STANDARD PROGRAMMING RUBRIC (DETAILED)

| Criteria | 4 | 3 | 2 | 1 | 0 | Total |
|---|---|---|---|---|---|---|
| **Function** | The code works without errors and fulfills all project requirements.<br><br>Test cases for in- and out-of-range inputs are successful.<br><br>Error handling is used correctly, where necessary and appropriate. | The code works but may generate some errors with edge cases.<br><br>Most project requirements are met.<br><br>Test cases for in-range inputs are successful.<br><br>Error handling, if required, is present. | The code compiles/executes but does not fulfill all project requirements.<br><br>Test case for at least one in-range input is successful.<br><br>Error handling, if required, is present. | The code is mostly complete but does not compile/execute.<br><br>Error handling, if required, is present. | The code is substantially incomplete or nonfunctional.<br><br>No error handling is present. | |

| | | | | | | |
|---|---|---|---|---|---|---|
| **Formatting** | The code is readable and formatted properly into control structures with indents and line spacing.<br><br>Variables are named appropriately and consistently.<br><br>The code is well-commented. | The code is readable with appropriate line spacing but not indented.<br><br>Variables are named appropriately but not consistently.<br><br>The code is mostly commented. | The code is readable but written in large blocks without indents or spacing.<br><br>Variables are inconsistently named.<br><br>The code is commented sporadically. | The code is written haphazardly and/or appears to be copied from external sources with minimal modification.<br><br>Variables are randomly named or single characters.<br><br>Minimal comments are present. | No attempt made to format the code.<br><br>Variables are single characters.<br><br>No comments are present. | |
| **Simplicity** | The code efficiently accomplishes what is in the project requirements.<br><br>Hardcoded values are used only where appropriate.<br><br>Complex programming logic is distributed in short, single-purpose lines of code. | The code includes some unnecessary extra steps but accomplishes the project requirements.<br><br>Some hardcoded values that will require future code updates are present.<br><br>Some multifunction or multistep lines of code are used. | Numerous extra blocks of code and/or unused variables are present.<br>The code includes many extra unnecessary steps.<br><br>Hardcoded values are used where variables should be.<br><br>Lines of code combine numerous logical functions/steps. | Code complexity is great enough to challenge future modification or debugging.<br>Extra/unnecessary steps in code throughout.<br><br>Programming flow is difficult to follow.<br><br>Hardcoded values are used instead of variables. | Code complexity is too great to allow future modifications or debugging.<br><br>Programming flow does not exist.<br><br>No logical order to programmatic steps.<br><br>Variables are not used or used incorrectly. | |

| | | | | | | |
|---|---|---|---|---|---|---|
| **Modularity** | The code follows the Don't Repeat Yourself best practice.<br><br>Each method/function performs one discrete task.<br><br>Variables are strongly typed and used for singular, discrete purposes. | The code mostly follows the Don't Repeat Yourself best practice.<br><br>Some methods/functions perform more than one discrete task.<br><br>Occasional variable repurposing or reuse exists. | Logical blocks of code are duplicated in different methods/functions.<br><br>Methods/functions combine multiple logical operations and steps.<br><br>Variables are untyped and/or occasionally reused. | Minimal effort to create logical blocks of code.<br><br>Procedural coding practices used for logical flow without code reuse.<br><br>Variables are untyped and/or repurposed. | No attempt made to create modular code.<br><br>All code exists in one large block.<br><br>Variables, if present, are untyped and/or repurposed. | |
| **Comprehension** | The code clearly demonstrates the student understands the concepts required.<br><br>The student can explain how the entire program works.<br><br>The student can explain why alternative solutions are not as good as the one(s) used. | The code shows the student understands most of the concepts required.<br><br>The student can explain how most of the program works.<br><br>The student can explain at least one alternative solution. | The code shows the student is aware of some of the concepts required.<br><br>The student can explain how some parts of the program work.<br><br>The student can discuss alternative solutions. | The code shows the student is aware of at least one of the concepts required.<br><br>The student can explain how at least one part of the program works.<br><br>The student can discuss at least one alternative solution. | The code does not show an awareness of the concepts required.<br><br>The student cannot explain how the program should work.<br><br>The student cannot discuss alternative solutions. | |
| | | | | | **Total** | |