IT104
Introduction to Computer Programming
[Onsite]

Credit hours: 4
Contact hours: 50 (30 Theory, 20 Lab)
Prerequisite: TB143 Introduction to Personal Computers or
TB145 Introduction to Computing
## Course Revision Table

<table>
<thead>
<tr>
<th>Footer Date</th>
<th>Section</th>
<th>Reason for Change</th>
<th>Implementation Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>09/30/2008</td>
<td>Entire Document</td>
<td>Newly developed course</td>
<td>December 2008</td>
</tr>
<tr>
<td>04/09/2009</td>
<td>Objectives</td>
<td>Identify math functions in objectives</td>
<td>Immediately</td>
</tr>
</tbody>
</table>
Where Does This Course Belong?

How does this course relate to the program? Take a look!

Introduction to Computer Programming is a course required to earn an associate’s degree in the Information Technology programs.

This course covers basic concepts in computer programming. The goals of this course are to:

- Lead the learner into the subject of computer programming, assuming no prior knowledge or experience
- Provide an applicable learning experience in programming literacy.

The following course sequence provides an overview of how Introduction to Computer Programming fits in the programs.

Note: Refer to the catalog for the state-specific course information.
Table of Contents

FIRST THINGS FIRST .............................................................................................................. I
SYLLABUS: INTRODUCTION TO COMPUTER PROGRAMMING---------------- 1
INSTRUCTOR GUIDE.............................................................................................................. 7
UNIT 1: FUNCTIONAL CONCEPTS.................................................................................................. 11
UNIT 2: SOFTWARE PROGRAM DESIGN ....................................................................................... 19
UNIT 3: SOFTWARE PROGRAM DESIGN (CON’T.)......................................................................... 24
UNIT 4: PROGRAM MODULES .................................................................................................... 32
UNIT 5: PROGRAM MODULES AND FUNCTIONS.......................................................................... 39
UNIT 6: DECISIONS I .................................................................................................................. 46
UNIT 7: DECISIONS II ................................................................................................................ 53
UNIT 8: REPETITIVE PROCESSING I ............................................................................................. 60
UNIT 9: REPETITIVE PROCESSING II ............................................................................................ 68
UNIT 10: REPETITIVE PROCESSING III AND COMPREHENSIVE LAB .................................... 74
UNIT 11: COURSE REVIEW AND COMPREHENSIVE LAB ......................................................... 78
APPENDIX A: TEST AND ANSWER KEYS .................................................................................... 79
APPENDIX B: PROJECT FOR THIS COURSE ................................................................................ 111
APPENDIX C: LAB SOLUTIONS .................................................................................................. 112
APPENDIX D................................................................................................................................ 113
APPENDIX E................................................................................................................................ 114
APPENDIX F................................................................................................................................ 115
APPENDIX G: HOMEWORK SOLUTIONS ...................................................................................... 116
APPENDIX H: LAB PRACTICUM .................................................................................................. 124
First Things First

Welcome! This courseware is designed to provide strategies and resources that will aid you in teaching this course.

The courseware components check list below provides a snapshot of this course. You may want to review the components of the courseware. A good way to track your review is by checking in the box next to each component as you review it. Reviewing the components will give you the bigger picture and better prepare you for what is coming up in the next few weeks. All the best!

Courseware Components Check List

<table>
<thead>
<tr>
<th>Courseware Components</th>
<th>Reviewed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syllabus</td>
<td></td>
</tr>
<tr>
<td>Course Description</td>
<td></td>
</tr>
<tr>
<td>Major Instructional Areas</td>
<td></td>
</tr>
<tr>
<td>Course Objectives</td>
<td></td>
</tr>
<tr>
<td>SCANS Objectives</td>
<td></td>
</tr>
<tr>
<td>Course Outline</td>
<td></td>
</tr>
<tr>
<td>Instructional Methods</td>
<td></td>
</tr>
<tr>
<td>Instructional Materials and References</td>
<td></td>
</tr>
<tr>
<td>Course Evaluation and Grading</td>
<td></td>
</tr>
<tr>
<td>Instructor Guide</td>
<td></td>
</tr>
<tr>
<td>Important Notes before You Start</td>
<td></td>
</tr>
<tr>
<td>Course Overview</td>
<td></td>
</tr>
<tr>
<td>Instructor Resources</td>
<td></td>
</tr>
<tr>
<td>Classroom and Lab Setup</td>
<td></td>
</tr>
<tr>
<td>Your Feedback</td>
<td></td>
</tr>
<tr>
<td>Unit 1</td>
<td></td>
</tr>
<tr>
<td>Objectives</td>
<td></td>
</tr>
<tr>
<td>Content Covered</td>
<td></td>
</tr>
<tr>
<td>Key Concepts That Must Be Covered in Class</td>
<td></td>
</tr>
<tr>
<td>Teaching Tips for This Unit</td>
<td></td>
</tr>
<tr>
<td>Homework</td>
<td></td>
</tr>
<tr>
<td>Labs</td>
<td></td>
</tr>
<tr>
<td>Summary and Reminders</td>
<td></td>
</tr>
<tr>
<td>Unit 2</td>
<td></td>
</tr>
<tr>
<td>Objectives</td>
<td></td>
</tr>
<tr>
<td>Readings</td>
<td></td>
</tr>
<tr>
<td>Key Concepts That Must Be Covered in Class</td>
<td></td>
</tr>
<tr>
<td>Teaching Tips for This Unit</td>
<td></td>
</tr>
<tr>
<td>Homework</td>
<td></td>
</tr>
<tr>
<td>Labs</td>
<td></td>
</tr>
<tr>
<td>Courseware Components</td>
<td>Reviewed</td>
</tr>
<tr>
<td>----------------------------------------</td>
<td>----------</td>
</tr>
<tr>
<td>Summary and Reminders</td>
<td></td>
</tr>
<tr>
<td><strong>Unit 3</strong></td>
<td></td>
</tr>
<tr>
<td>Objectives</td>
<td></td>
</tr>
<tr>
<td>Readings</td>
<td></td>
</tr>
<tr>
<td>Key Concepts That Must Be Covered in Class</td>
<td></td>
</tr>
<tr>
<td>Teaching Tips for This Unit</td>
<td></td>
</tr>
<tr>
<td>Homework</td>
<td></td>
</tr>
<tr>
<td>Labs</td>
<td></td>
</tr>
<tr>
<td>Summary and Reminders</td>
<td></td>
</tr>
<tr>
<td><strong>Unit 4</strong></td>
<td></td>
</tr>
<tr>
<td>Objectives</td>
<td></td>
</tr>
<tr>
<td>Readings</td>
<td></td>
</tr>
<tr>
<td>Key Concepts That Must Be Covered in Class</td>
<td></td>
</tr>
<tr>
<td>Teaching Tips for This Unit</td>
<td></td>
</tr>
<tr>
<td>In-Class Assessment</td>
<td></td>
</tr>
<tr>
<td>Homework</td>
<td></td>
</tr>
<tr>
<td>Labs</td>
<td></td>
</tr>
<tr>
<td>Summary and Reminders</td>
<td></td>
</tr>
<tr>
<td><strong>Unit 5</strong></td>
<td></td>
</tr>
<tr>
<td>Objectives</td>
<td></td>
</tr>
<tr>
<td>Readings</td>
<td></td>
</tr>
<tr>
<td>Key Concepts That Must Be Covered in Class</td>
<td></td>
</tr>
<tr>
<td>Teaching Tips for This Unit</td>
<td></td>
</tr>
<tr>
<td>Homework</td>
<td></td>
</tr>
<tr>
<td>Labs</td>
<td></td>
</tr>
<tr>
<td>Summary and Reminders</td>
<td></td>
</tr>
<tr>
<td><strong>Unit 6</strong></td>
<td></td>
</tr>
<tr>
<td>Objectives</td>
<td></td>
</tr>
<tr>
<td>Readings</td>
<td></td>
</tr>
<tr>
<td>Key Concepts That Must Be Covered in Class</td>
<td></td>
</tr>
<tr>
<td>Teaching Tips for This Unit</td>
<td></td>
</tr>
<tr>
<td>In-Class Assessment</td>
<td></td>
</tr>
<tr>
<td>Homework</td>
<td></td>
</tr>
<tr>
<td>Labs</td>
<td></td>
</tr>
<tr>
<td>Summary and Reminders</td>
<td></td>
</tr>
<tr>
<td><strong>Unit 7</strong></td>
<td></td>
</tr>
<tr>
<td>Objectives</td>
<td></td>
</tr>
<tr>
<td>Readings</td>
<td></td>
</tr>
<tr>
<td>Key Concepts That Must Be Covered in Class</td>
<td></td>
</tr>
<tr>
<td>Teaching Tips for This Unit</td>
<td></td>
</tr>
<tr>
<td>Homework</td>
<td></td>
</tr>
<tr>
<td>Labs</td>
<td></td>
</tr>
<tr>
<td>Summary and Reminders</td>
<td></td>
</tr>
<tr>
<td><strong>Unit 8</strong></td>
<td></td>
</tr>
</tbody>
</table>
## Courseware Components

<table>
<thead>
<tr>
<th>Component</th>
<th>Reviewed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Objectives</td>
<td></td>
</tr>
<tr>
<td>Readings</td>
<td></td>
</tr>
<tr>
<td>Key Concepts That Must Be Covered in Class</td>
<td></td>
</tr>
<tr>
<td>Teaching Tips for This Unit</td>
<td></td>
</tr>
<tr>
<td>Homework</td>
<td></td>
</tr>
<tr>
<td>Labs</td>
<td></td>
</tr>
<tr>
<td>Summary and Reminders</td>
<td></td>
</tr>
</tbody>
</table>

### Unit 9

| Objectives                                     |          |
| Readings                                       |          |
| Key Concepts That Must Be Covered in Class     |          |
| Teaching Tips for This Unit                    |          |
| Homework                                       |          |
| Labs                                           |          |
| Summary and Reminders                          |          |

### Unit 10

| Objectives                                     |          |
| Readings                                       |          |
| Key Concepts That Must Be Covered in Class     |          |
| Teaching Tips for This Unit                    |          |
| Lab Practicum                                  |          |
| Summary and Reminders                          |          |

### Unit 11

| Course Review                                  |          |
| Unit Exam                                      |          |
| Lab Practicum                                  |          |

### Appendix A: Test and Answer Keys

| Unit Exam 1                                    |          |
| Unit Exam 2                                    |          |
| Unit Exam 3                                    |          |

### Appendix B: Project for This Course

N/A

### Appendix C: Lab Solutions

N/A

### Appendix D

N/A

### Appendix E

N/A

### Appendix F

N/A

### Appendix G: Homework Solutions

N/A

### Appendix H: Lab Practicum

N/A
Syllabus: Introduction to Computer Programming

Instructor: ________________________________________
Office hours: ________________________________________
Class hours: ________________________________________

Course Description
This course serves as a foundation for understanding the logical function and process of computer programming in a given language environment. Basic computer programming knowledge and skills in logic and syntax will be covered. Coding convention and procedures will be discussed relevant to the given programming language environment.

Major Instructional Areas
1. Fundamental concepts of computer programming
2. Memory allocation and variables
3. Problem specification and analysis using flowcharting and pseudocode
4. Conditional statements and repetition structures
5. Coding conventions and procedures

Course Objectives
1. Describe the fundamental concepts in computer programming by applying mathematical concepts and operations.
2. Create basic software program designs.
3. Create various modules in computer programs.
4. Create various functions in computer programs.
5. Create computer programs that can make decisions.
6. Create computer programs that can do repetitive processing.
7. Create computer programs that can do input validation.
8. Create program software that incorporates modules (and/or functions), conditional logic, looping, and input validation.
9. Test program software.

SCANS Objectives
SCANS is an acronym for Secretary’s Commission on Achieving Necessary Skills. The committee, created by the National Secretary of Labor in the early 1990s, created a list of skills and competencies that the committee feels are necessary for employees to function in a high-tech job market.
1. Identify relevant facts and analyze information in a logical manner after locating and verifying information using resources and computers.
2. Identify common goals and examine all possible options for problem solving.
3. Identify problems, create and implement solutions, and revise solutions, as required.
4. Allocate time and energy for completing projects in a timely manner.
5. Exert a high level of effort and perseverance toward attaining goals.
6. Recognize problems and devise and implement a plan of action.
7. Demonstrate the ability to utilize authentic resources available, including the Internet, knowledge libraries, or other sources.
8. Locate, understand, and interpret information obtained from a variety of sources.
9. Identify the need for data; select, retrieve, and analyze information; and communicate the results of information analysis in written, graphical, and pictorial formats.
10. Compare and contrast two theories or alternatives to arrive at the best solution.
11. Apply procedures, tools, and equipment—including computers and related technologies—whenever required.
12. Evaluate alternatives and choose the best for a situation.

Course Outline

Note: All graded activities, except the Lab Practicum and Unit Exams, are listed below in the pattern of <Unit Number>.<Assignment Number>. For example, Lab 1.1 refers to the 1st lab activity in Unit 1.

<table>
<thead>
<tr>
<th>Unit</th>
<th>Activities</th>
</tr>
</thead>
</table>
| 1— Fundamental Concepts   | • Content Covered: Starting Out with Programming Logic & Design  
                               o Chapter 1, “Introduction to Computers and Programming”  
                               • Labs: 1.1  
                               • Assignments: 1.1 |
| 2— Software Program Design| Read from Starting Out with Programming Logic & Design:  
                               o Chapter 2, “Input, Processing, and Output,” pp. 29-55  
                               • Labs: 2.1  
                               • Assignments: 2.1 |
| 3— Software Program Design (con’t.) | Read from Starting Out with Programming Logic & Design:  
                               o Chapter 2, “Input, Processing, and Output,” pp. 56-68  
                               • Labs: 3.1  
                               • Assignments: 3.1 |
| 4— Program Modules        | Read from Starting Out with Programming Logic & Design:  
                               o Chapter 3, “Modules,” pp. 75-90  
                               • Unit Exams: 1  
                               • Labs: 4.1 |
<table>
<thead>
<tr>
<th>Unit</th>
<th>Activities</th>
</tr>
</thead>
<tbody>
<tr>
<td>5— Program Modules and Functions</td>
<td>• Assignments: 4.1</td>
</tr>
</tbody>
</table>
| | Read from *Starting Out with Programming Logic & Design:*  
| | o Chapter 3, “Modules,” pp. 90-108  
| | • Labs: 5.1  
| | • Assignments: 5.1 |
| 6— Decisions I | Read from *Starting Out with Programming Logic & Design:*  
| | o Chapter 4, “Decision Structures and Boolean Logic,” pp. 115-142  
| | • Unit Exams: 2  
| | • Labs: 6.1  
| | • Assignments: 6.1 |
| 7— Decisions II | Read from *Starting Out with Programming Logic & Design:*  
| | o Chapter 4, “Decision Structures and Boolean Logic,” pp. 142-155  
| | • Labs: 7.1  
| | • Assignments: 7.1 |
| 8— Repetitive Processing I | Read from *Starting Out with Programming Logic & Design:*  
| | o Chapter 5, “Repetition Structures,” pp. 163-183 and pp. 196-201  
| | • Labs: 8.1  
| | • Assignments: 8.1 |
| 9— Repetitive Processing II | Read from *Starting Out with Programming Logic & Design:*  
| | o Chapter 5, “Repetition Structures,” pp. 183-211  
| | • Labs: 9.1  
| | • Assignments: 9.1 |
| 10— Repetitive Processing III and Comprehensive Lab | Read from *Starting Out with Programming Logic & Design:*  
| | o Chapter 7, “Input Validation”  
| | • Lab Practicum (Part I) |
| 11— Course Review and Comprehensive Lab | • Course Review  
| | • Lab Practicum (Part II)  
| | • Unit Exams: 3 |

**Instructional Methods**

The curriculum is designed to promote a variety of teaching strategies that support the outcomes described in the course objectives and that foster higher cognitive skills. Delivery makes use of various media and delivery tools in the classroom.

Multiple styles, such as lectures, collaborative learning options, and hands-on laboratory activities, will be used to deliver content and inspire and engage students. Your progress will be regularly assessed using various accessible methods and tools. This course uses
various learning strategies such as exams, assignments, lab exercises, and a lab practicum to help you understand the concepts. Assignments are based on the concepts covered in different units. Units 4, 6, and 11 each have an exam. These exams will analyze your learning and help you recall the concepts already taught.

Classroom practices will create a climate of high values with respect to both diversity and inclusiveness. An open communication environment will help to ensure useful interactions between students and the instructor and among students themselves. Lesson plans, course materials, notes, or other information resources will be made available and be made flexible to all students as needed.

**Instructional Materials and References**

**Student Textbook Package**

- Python/Raptor software CD (included with textbook)

**References**

**ITT Tech Virtual Library**

Log on to the ITT Tech Virtual Library at http://www.library.itt-tech.edu/ to access online books, journals, and other reference resources selected to support ITT Tech curricula.

**Books**

You may click “Books” or use the “Search” function on the home page to find the following books.

ITT Tech Virtual Library> Main Menu> Books> Books24x7>

Other References

The following resources may be found outside of the ITT Tech Virtual Library, whether online or in hard copy.

Web sites

- Algorithms
  http://courses.cs.vt.edu/~csOnline/Algorithms/Lessons/index.html
  Self-paced lessons on understanding and using algorithms
- Number Systems
  http://courses.cs.vt.edu/~csOnline/NumberSystems/Lessons/index.html
  Self-paced lessons on understanding and using number systems, including binary, hexadecimal, and octal
- Programming Languages
  http://courses.cs.vt.edu/~csOnline/ProgrammingLanguages/Lessons/index.html
  Self-paced lessons on understanding and using various programming languages
- Programming Tutorials
  http://www.programmingtutorials.com/python.aspx
  A collection of links to online tutorials available on Python software
- Python
  http://www.python.org
  The official Web site of the Python Software Foundation, with news, documentation, and downloads

Books

  Available on FreeTechBooks.com:
  http://freetechbooks.com/about134.html&highlight=allen+downey
  This downloadable book teaches readers to think like a computer scientist regarding the use of Python.

All links to Web references outside of the ITT Tech Virtual Library are always subject to change without prior notice.
Course Evaluation and Grading

Evaluation Criteria Table

The final grades will be based on the following categories:

<table>
<thead>
<tr>
<th>CATEGORY</th>
<th>WEIGHT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Labs</td>
<td>30%</td>
</tr>
<tr>
<td>Assignments</td>
<td>15%</td>
</tr>
<tr>
<td>Unit Exam 1</td>
<td>15%</td>
</tr>
<tr>
<td>Unit Exam 2</td>
<td>10%</td>
</tr>
<tr>
<td>Unit Exam 3</td>
<td>15%</td>
</tr>
<tr>
<td>Lab Practicum</td>
<td>15%</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>100%</strong></td>
</tr>
</tbody>
</table>

Note: Students are responsible for abiding by the Plagiarism Policy.

Grade Conversion Table

The final grades will be calculated from the percentages earned in the course, as follows:

<table>
<thead>
<tr>
<th>Grade</th>
<th>Percentage</th>
<th>GPA</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>90–100%</td>
<td>4.0</td>
</tr>
<tr>
<td>B+</td>
<td>85–89%</td>
<td>3.5</td>
</tr>
<tr>
<td>B</td>
<td>80–84%</td>
<td>3.0</td>
</tr>
<tr>
<td>C+</td>
<td>75–79%</td>
<td>2.5</td>
</tr>
<tr>
<td>C</td>
<td>70–74%</td>
<td>2.0</td>
</tr>
<tr>
<td>D+</td>
<td>65–69%</td>
<td>1.5</td>
</tr>
<tr>
<td>D</td>
<td>60–64%</td>
<td>1.0</td>
</tr>
<tr>
<td>F</td>
<td>&lt;60%</td>
<td>0.0</td>
</tr>
</tbody>
</table>

(End of Syllabus)
Instructor Guide

Important Notes before You Start

1. The activities and assignments provided in each unit are suggested. You may choose to provide replacement items where deemed necessary and appropriate. The replacement items must map directly to the learning objectives and the prescribed Evaluation Criteria Table in the syllabus.

2. You should inspire, motivate, and direct student usage of the ITT Tech Virtual Library resources by applying the graded learning activities designed in the course.

3. Distribute the evaluation criteria of any graded assignment to students.

4. When you begin each unit:
   a. Ask students if they have any questions related to the previous unit’s content, homework assignment, and/or lab activities.
   b. Collect the previous unit’s assignments.
   c. Provide a short overview of the previous unit’s content and relate that content to the current unit being taught. This will provide continuity for the students.

5. At the end of each unit:
   a. Briefly review the content that was covered in the unit, and highlight the most important aspects. Relate the content covered to the assignments given for that unit.
   b. Remind the students of the next unit’s reading and any deliverables that will be due. Suggested homework assignments have been provided.
   c. When explaining the assignments, clearly state your expectation and explain how their work will be graded.

6. Assessments:
   a. *Homework:* The exercises given in this course are smaller assignments designed to have the students apply the topics covered in the textbook. The students should submit their homework assignments in a typed word processing document. Note: No homework assignment is given during Units 10 and 11. Homework solutions can be found in Appendix G.
   b. *Unit Exams:* Three unit exams are administered throughout the course. Each exam will contain multiple-choice questions. The exams and answers are in Appendix A.
   c. *Lab Practicum:* The Lab Practicum is given during Units 10 and 11. The students are to design, create, and debug program software that implements input validation, decision making, and repetition. The program will be designed using pseudocode and flowcharts, and then implemented in the Python programming language. The Lab Practicum details can be found in Appendix H.
Course Overview

While lectures are important, the core focus of the course should be hands-on laboratory activities so that students can apply concepts covered in the lecture. The Comprehensive Lab Practicum at the end of the course optimizes this approach and is meant to provide a cumulative review of all the concepts learned in the course. Examinations in the course are divided into three smaller exams to provide a more accurate assessment of a student’s progress throughout the course and help identify problem areas early. You are encouraged to provide as many hands-on examples as practical and tailor labs and homework to suit your students’ needs, interests, and areas of specialization. It is highly recommended that the comprehensive practical lab should not be generic but tailored to student needs. Students should also be encouraged to research topics in the ITT Tech Virtual Library and make full use of the reference resources available there.

Throughout the course, focus on building analytical skills and logical thinking to solve programming problems. Do not emphasize learning the finer points of programming in Python. Point out to students that Python is being used merely as a tool to illustrate programming concepts and that this is not a “Programming in Python” course.

Instructor Resources

Resources from Publisher

Instructor resources may be available to support your textbook from Pearson. To find out if instructor resources are available:

1. Go to http://www.pearsoncustom.com/ITTinstructor, click the Access button, select your course from the dropdown list, and then click the Go button. You will see the student package for your course.
2. Click on Supplements in the left column to see if supplements are available. For digital resources:
   a. Click the file name link you want to download. Enter your login name and password.
   b. If you do not already have a login name and password, contact Instructor.ITT@Pearson.com to request an access code.
   c. Follow the steps below to download:
      i. Click the Log In button.
      ii. A new page will appear requesting you to acknowledge Pearson's terms of acceptance to ensure all users understand the proprietary nature of their products and the importance of keeping instructor resources in the hands of educators. Click I Accept to begin the download process.
      iii. The download process starts immediately.
   For printed or hard-media resources:
      a. Click on Order in the left column to access the Order Form to order printed or hard-media instructor resources. If you have not ordered using this form before, you will be asked to create a profile (your mailing
information), then given an instructor code. This code will enable you to quickly order material in the future; please save it.

b. Enter your instructor code.
c. You will be taken to an order page where you will find all the available resources. Indicate the copy type and quantity you need, then click the **Place Order** button at the bottom.
d. You will receive an on-screen confirmation.

Online help is available wherever downloads are available. For assistance in downloading instructor resources, click the link “Help with Downloading Instructor Resources.”

If you have any further questions, please contact Instructor.ITT@Pearson.com.

**Internal Curriculum Download Site**

The following resources are available through the **internal curriculum download site**: N/A

**Classroom and Lab Setup**

**Classroom Setup**

The theory portion of this course must be taught in a theory classroom with standard classroom equipment.

- A projection system
- Tables, chairs, whiteboard
- Instructor computer with the following configuration:
  - Desktop computer with any Microsoft Windows operating system
  - At least 20 GB hard disk for a typical installation
  - CD-ROM drive
  - Local Area Network (LAN) connection
  - Internet connection
  - Python compiler
  - Python IDE
  - Raptor

**Lab Setup**

The labs are to be conducted in the standard computer lab rooms. The computers should have the following configuration specific for this course:

- CD-ROM drive
- Local Area Network (LAN) connection
- Internet connection
- Python compiler
- Python IDE
- Raptor
Your Feedback

After you teach this course, answer the following questions to document your feedback. Your answers will help improve any future versions of the course. Therefore, be as descriptive as possible. If you have any suggestions for improvements, mention them as well.

Note: Please submit your feedback to your Program Chair / Associate Dean. All of the feedback submitted for this will be collected and forwarded to the Curriculum Department to be used in the revision cycle.

Date: _________________________

1. Comment on the appropriateness and level of the textbook with reference to the level of the students and the purpose of the course.

2. Do the course objectives cover the scope of the course comprehensively? If no, please provide specific examples. Comment on the difficulty level of the objectives.

3. Were the assignments and labs too difficult or too easy for the students? Describe the positive and negative aspects of assignments and labs. Please provide specific examples.

4. Are you satisfied with the quality of exam and/or quiz questions and their answers? If not, mention the specific areas of dissatisfaction, giving examples.

5. Which components in the syllabus, instructor guide and unit plan did you find most useful and which components redundant? Explain, giving reasons.

6. Explain what you like the most in the course, what challenges you encountered, and how you solved them while teaching this course.

7. Is there any other feedback that you would like to provide?

(End of Instructor Guide)
Unit 1: Fundamental Concepts

Objectives

1. Describe the fundamental concepts in computer programming.
   1.1: Describe the role of software for computers.
   1.2: Identify the hardware associated with a computer.
   1.3: Describe how computers store data.
   1.4: Explain how programs work.
   1.5: Differentiate among machine language, assembly language, and high-level languages.
   1.6: Differentiate between compilers and interpreters.
   1.7: Identify the different types of software.

Content Covered


• Chapter 1, pp. 1-22

Key Concepts That Must Be Covered in Class

The following key concepts must be covered in class in order to achieve the course objectives.

1. Role of software and how it works
2. How data is stored
3. Types of software, programming languages
4. Compilers vs. interpreters

Teaching Tips for This Unit

The teaching tips are provided for your reference. Please choose carefully according to your teaching style and students’ learning style.

• This unit gives you an excellent opportunity to provide students with a sense of why they are taking this course and why it is so important. Typically, for programming students this is not really an issue, but for other students it is not necessarily so clear why they should be taking this course.

Briefly introduce this course to the students with some of the following examples:

○ Without software programs to run networking devices and applications that require a network, computer networks would not be required or
even possible. Computer network administrators need to have solid programming skills to write scripts and troubleshoot problems with the applications that generate most of their network traffic.

- Without software programs, computer graphics and computer games would not be possible. Advanced computer-aided design and game design techniques require scripting and a solid knowledge of programming concepts.

- Without software programs, modern robotic devices and embedded systems would not be possible. Programming of embedded systems and “smart” devices require solid programming skills.

Explain to the students that without software programs to execute, computers are just expensive paperweights and if you want to tell the computer what to do, you need to learn how to program—provide them with instructions to execute. Also stress to the students that this is not a course about a particular programming language but rather about solid programming skills. (This tip refers to key concept #1—Role of software and how it works.)

- Review the basic components of the computer: CPU, main memory, and secondary storage. Review how a CPU can only execute binary instructions and how all secondary storage memory must be first transferred into main memory in order to be used by the CPU. (This tip refers to key concept #1—Role of software and how it works.)

- Discuss the Fetch-Decode-Execute cycle as illustrated in Figure 1-15 on p. 13 and Figure 1-16 on p. 14 of the textbook. If necessary, reference the discussion of physical memory and secondary storage from the TB143 Introduction to Personal Computers, and IT103 Operating Systems courses. (This tip refers to key concept #1—Role of software and how it works.)

- Use the analogy of a byte being represented as a series of 8 switches provided on pp. 7-10 of the textbook to discuss binary data storage. This will help to focus the students on the storage of data in binary form. Stress to the students that they will not be required to do binary calculations. It will not be necessary for you to cover binary arithmetic. (This tip refers to key concept #2—How data is stored.)

- Introduce ASCII and Unicode encoding schemes for the representation of text data. Use Figure 1-13 on p. 10 and Appendix A on p. 545 of the textbook to aid in your discussion. Explain the storage of different types of data and how different types of data may require different amounts of storage. (This tip refers to key concept #2—How data is stored.)

- Describe and discuss the differences between system software and application software. Refer the students to pp. 20-22 of the textbook for the discussion. Point
out that the same programming principles apply to creating or modifying either systems software or applications software. (This tip refers to key concept #3—Types of software, programming languages.)

- Describe the differences among machine language, assembly language, and high-level languages. Explain to the students that machine and assembly languages are usually CPU-dependent and not directly transferable between different types of computers, let alone operating systems. This non-transferability means a lot of costly rewriting of program code if a company decides to switch hardware vendors or operating system platforms. (This tip refers to key concept #3—Types of software, programming languages.)

- Use Table 1-1 on p. 16 of the textbook to discuss high-level languages such as C++, Java, Visual Basic, and Python. These languages are not only easier to use to create programs, but are also not as platform-dependent as low-level languages. Discuss that this flexibility comes at the price of speed and higher system requirements. (This tip refers to key concept #3—Types of software, programming languages.)

- Explain that CPUs require instructions in a very strict format and high-level languages must be converted into the strict format and sequence required for accurate machine language. The instructions must be converted using strict syntax rules by another piece of software: either an interpreter or a compiler. Discuss how a compiler creates an executable binary file while an interpreter creates binary instructions that are executed immediately. Precompiled binary files run faster but must be compiled ahead of time for the specific platform on which they are used. (This tip refers to key concept #4—Compilers vs. interpreters.)

- For interpreted programs, the translation to binary commands is typically executed by an interpreter on the client’s machine. Use Figures 1-18 and 1-19 on p. 18 of the textbook for this discussion. Interpreted programs are slower, but in some cases you cannot predict ahead of time what system the client will be using. Use the example of a JavaScript-enabled web browser used to run web page scripts—as long as the client has the JavaScript interpreter, it can run the code. Both interpreters and compilers are typically run inside software packages called Integrated Development Environments (IDEs) for ease of use. Figure 1-20 on p. 19 of the textbook is an illustration of the Visual Studio IDE. (This tip refers to key concept #4—Compilers vs. interpreters.)

- Explain to the students that the purpose of the homework assignment is to review the key concepts discussed in the class. Students should submit their answers in the next class. Answers can be found in Appendix G. (This tip refers to key concepts #1—Role of software and how it works, #2—How data is stored, #3—Types of software, programming languages, and #4—Compilers vs. interpreters).
Homework

The following homework is designed to cover the course objectives for this unit.

Assignment 1.1: Answer the following questions in Chapter 1 of your textbook, *Starting Out with Programming Logic & Design*:

- Multiple Choice Review Questions 5, 14, 15, 21, and 24, starting on p. 23
- Short Answer Review Questions 1 and 6 on p. 26

Submit your answers in a word-processed document to your instructor in Unit 2.
Labs

Lab 1.1: Fundamental Concepts of Programming Case Study

What is the purpose?

The purpose of this lab is to reinforce the fundamental concepts concerning different programming languages and compiled vs. interpreted programs.

What are the steps?

- **Task 1: Group Discussion**
  
  **Procedure**
  1. In small groups, discuss proposals for which type of software would be best suited for each of the scenarios below. Determine if the software program should be written in a low-level language, high-level language compiled, or high-level language interpreted.
     - Program software that monitors vehicle performance for an onboard computer in a car
     - Order processing software that will run in users’ Web browsers when they access your company’s Web site
     - Software that runs inside a $20 USB drive to allow the USB drive to talk to Windows (XP or later) computers’ operating systems only
     - Gaming software for a fast-paced action game that runs on a single computer (i.e., not a multi-player game)
     - Software to detect and track complex network intrusion attacks on Linux networks only

- **Task 2: Written Proposal**
  
  **Procedure**
  1. As a group, summarize your proposal for the software of your choice for each scenario. Your proposal should include the following:
     - Clear, concise list of reasons why you selected that type of software program
     - Clear, concise list of reasons why you did not choose other options

  Use the ITT Tech Virtual Library to research the various software programs. Proposals should properly reference the textbook and any other references used from the ITT Tech Virtual Library. Submit your typed proposal to your instructor for grading.

- **Task 3: Introduction to Pseudocode**
  
  **Procedure**
  1. Writing programs requires a lot of planning. A programmer needs to plan out data required by the program, the process that needs to be
conducted on the input data, and finally the output required by the program. Such considerations are very important—if the program does not store the necessary data in memory before it is needed, the program will not be able to do the processing it needs to do in order to generate the required output. For the following example, consider a calculator that stores data in up to 30 different memory storage locations (M1, M2, M3, …M30) as follows:

- **Example #1:** A program to calculate the perimeter of a square using the calculator
  - **Input:** Store length of a side in memory location M1.
  - **Process:** Calculate perimeter: M1*4
  - **Output:** Display perimeter of the square.

- **Example #2:** A program to calculate order totals for buying shirts using the calculator
  - **Input:** Store price of shirt in memory location M1.
  - **Process:** Calculate total before tax and store in memory location M4: M4 = M1 * M2
  - **Output:** Display order total before tax, total sales tax and total including tax.

2. Map out the Input, Process, and Output steps for the following programs:

- **Problem #1:** A program to calculate order totals for buying five different model laptops using the calculator. Assume the different models all have different prices but the same sales tax rate.

- **Problem #2:** A program to calculate personal income tax using the calculator. Assume you can get up to five different income tax deductions.

**Did it work?**

- Did your group select an appropriate software package for each of the scenarios listed?
- Did you use the resources of the ITT Tech Virtual Library to research the various software packages?
• Did you properly reference any resources used in your proposals?
• Were you able to map out the Input, Process, and Output steps for the two problems?
• Did you submit your proposals to your instructor for grading?
Summary and Reminders

Summary

- This unit covers the role of software and how it works.
- The concepts of how data is stored and types of software were discussed.
- The differences among programming languages were discussed.
- The differences between compilers and interpreters were explained.

Reminders

- Students should prepare for the next unit by reading Chapter 2, pp. 29-55, of their textbook prior to coming to class.
- Assignment 1.1 will be due in the next class period.
- The content covered in Unit 1 will be assessed in Unit Exam 1, which will be administered in Unit 4.

(End of Unit 1)
Unit 2: Software Program Design

Objectives

2. Create basic software program designs.
   2.1: Determine program input, processing, and output stages.
   2.2: Create the necessary flowcharts to describe a program’s structure.
   2.3: Use pseudocode to define a program’s structure.
   2.6: Formulate solution algorithms for calculations by properly following the order of operations.

Readings


- Chapter 2, pp. 29-55

Key Concepts That Must Be Covered in Class

The following key concepts must be covered in class in order to achieve the course objectives.

1. Determining input, process, and output
2. Creating flowcharts and pseudocode
3. Formulating algorithms

Teaching Tips for This Unit

The teaching tips are provided for your reference. Please choose carefully according to your teaching style and students’ learning style.

- Discuss how all programs basically consist of a three-step process: input, process, and output. Use the payroll calculation program example in Figure 2-3 on p. 34 of the textbook to illustrate this example. Ask the students to think of and share other examples that follow this three-step process. (This tip refers to key concept #1—Determining input, process, and output.)

- Discuss how the necessary input and process to achieve the output are not always very easy to determine, and require a disciplined approach to solve programming problems efficiently. Describe both syntax errors and logic errors and explain how logic errors can be the hardest to detect and solve. Explain the debugging process
and the software/program development life cycle using Figure 2-1 on p. 30 to illustrate this process. Stress to the students that understanding the requirements for the program is very important to project success and developing a successful solution algorithm. (This tip refers to key concepts #1—Determining input, process, and output and #3—Formulating algorithms.)

- Use the employee payroll example from pp. 32-33 of the textbook to illustrate pseudocode and flowcharting. Show the students the following examples in addition to the textbook example. This will help stress the importance of these two techniques in designing programs.
  - Converting from decimal to binary numbers
  - Calculating mileage values
  - Encrypting information

Stress to the students that pseudocode is not real code and that the programming language we will be using uses different syntax than that used in the pseudocode. Refer students to Appendix B on p. 546 of the textbook for further reference on flowchart symbols. (This tip refers to key concept #2—Creating flowcharts and pseudocode.)

- Emphasize to the students the importance of the sequence of steps in an algorithm, and that flowcharts describe sequence structures. You can use Figure 2-6 on p. 36 of the textbook to illustrate sequence structures. Discuss the use of variables in calculations, variable naming rules, and why variables are necessary to grab user input and perform calculations. (This tip refers to key concept #3—Formulating algorithms.)

- Review the correct syntax for variable assignments by reviewing the Note boxes on pp. 44 and 45 of the textbook as examples. Use Table 2-1 on p. 46 and Table 2-2 on p. 49 of the textbook to review arithmetic operators and order of precedence of operations. Use the Calculating an Average example on pp. 50-52 of the textbook to illustrate algorithm formulation. (This tip refers to key concept #3—Formulating algorithms.)

- Explain to the students that the purpose of the homework assignment is to review the key concepts discussed in the class. Students should submit their answers in the next class. Answers can be found in Appendix G. (This tip refers to key concepts #1—Determining input, process, and output, #2—Creating flowcharts and pseudocode, and #3—Formulating algorithms.)
Homework

The following homework is designed to cover the course objectives for this unit.

Assignment 2.1: Answer the following questions from Chapter 2 of your textbook, *Starting Out with Programming Logic & Design*:

- Multiple Choice Review Questions 1, 3, 4, and 8, starting on p. 68
- Algorithm Workbench Review Questions 1-3 on p. 71
- Programming Exercises Question 8 on p. 73

Submit your answers in a word-processed document to your instructor in Unit 3.
Labs

Lab 2.1: Input, Processing, and Output:

   Algorithms
   Pseudocode
   Flowcharts

What is the purpose?

In this lab, you will gain practice in identifying program input, process, and output and in designing solution algorithms for programs. You will also practice designing programs by using pseudocode and flowcharts.

What are the steps?

- **Task 1: Lab 1.1—Algorithms**
  **Procedure**
  2. Submit your answers to Steps 2-4 to your instructor for grading.

- **Task 2: Lab 1.2—Pseudocode**
  **Procedure**
  1. Complete Lab 1.2—Pseudocode on pp. 3-6 of the *Lab Manual to Accompany Starting Out with Programming Logic & Design*.
  2. Submit your answers to Steps 1-6 on pp. 4-6 to your instructor for grading.

- **Task 3: Lab 1.3—Flowcharts**
  **Procedure**
  1. Complete Lab 1.3—Flowcharts on pp. 7-10 of the *Lab Manual to Accompany Starting Out with Programming Logic & Design*.
  2. Submit your flowchart from Step 10 to your instructor for grading.

Did it work?

- Were you able to find the logic errors in Tasks 1 and 2?
- Were you able to correct the error found in Tasks 1 and 2?
- Were you able to use Raptor correctly to create a flowchart?
- Were you able to create your flowchart properly?
- Did you submit your work to your instructor for grading?
Summary and Reminders

Summary

- This unit introduced the basic parts of a software program: input, process, and output.
- The fundamental techniques used to design software were introduced, including creating flowcharts, writing pseudocode, and formulating algorithms.

Reminders

- Students should prepare for the next unit by reading Chapter 2, pp. 56-68, of their textbook prior to coming to class.
- Assignment 2.1 will be due in the next class period.
- The content covered in Unit 2 will be assessed in Unit Exam 1, which will be administered in Unit 4.

(End of Unit 2)
Unit 3: Software Program Design (con’t.)

Objectives

2. Create basic software program designs.
   2.4: Describe the use of variables (declaration, naming, assignment, and initialization) in program designs.
   2.5: Use the correct data type for variables in program designs.
   2.7: Describe the usefulness of properly commenting code.

Readings


- Chapter 2, pp. 56-68

Key Concepts That Must Be Covered in Class

The following key concepts must be covered in class in order to achieve the course objectives.

1. Variable declaration, naming, and initialization
2. Data types
3. Comments

Teaching Tips for This Unit

The teaching tips are provided for your reference. Please choose carefully according to your teaching style and students’ learning style.

- Briefly review the input, process, and output stages of a program from the previous unit. Explain that variables store the data necessary for a program to execute. Before data can be stored, the operating system needs to be told how much memory will be required and how to find the data once it has been stored. Relate how this is accomplished by a variable declaration, which must occur before the variable can be used. Stress to the students the necessity of a unique variable name if the program is going to be able to find the data stored by the variable.

Discuss how different types of data require different amounts of memory storage. Explain that depending upon the programming language used, data type may be specified in the variable declaration. Use Program 2-11 and Program 2-12 on pp.
57-58 of the textbook to illustrate this concept. Be sure to mention that not all programming languages require data types in variable declarations. For example, Python does not require data types in variable declarations. (This tip refers to key concepts #1—Variable declaration, naming, and initialization and #2—Data types.)

- Briefly discuss the importance of variable initialization and how using variables in calculations before they have been initialized can lead to errors. Use the modified version of Program 2-12 found on the bottom of p. 59 of the textbook to demonstrate this error. (This tip refers to key concept #1—Variable declaration, naming, and initialization.)

- Use the examples on pp. 62-63 of the textbook to introduce the concept of constants, and how constants must be initialized immediately upon declaration and cannot be changed after declaration. (This tip refers to key concept #1—Variable declaration, naming, and initialization.)

- Students must understand variables in order to be successful in implementing algorithms in program code. Be sure to spend an adequate amount of time explaining variables to the students. (This tip refers to key concept #1—Variable declaration, naming, and initialization.)

- Briefly discuss the errors that are possible with incompatible data types and data truncation. Examples can be found on pp. 60-61 of the textbook. Be sure to mention that some programming languages are more forgiving than others with data types and data type conversion. (This tip refers to key concept #2—Data types.)

- Stress the importance of proper indentation and formatting while providing examples of pseudocode and program code to the class. These are key skills that must be developed by the students in order to create easily readable program code. Lead the students into a discussion of comments and internal documentation. (This tip refers to key concept #3—Comments.)

- Discuss the importance of internal documentation (i.e., comments) for programmers. Explain that for future program maintenance, comments are essential. The textbook mentions both block and line comments (p. 65), but only shows an example of C++ style line comments. You may want to provide an example of a block comment for illustration, but it is not necessary to dwell too long on block comments since they are not supported by Python. Be sure to point out that different languages use different symbols to denote where a line comment begins. For example, Python uses the # symbol as opposed to the two forward slashes (//{{} used in the textbook. (This tip refers to key concept #3—Comments.)

- Demonstrate how variables are applied by working through the hand tracing example found in Figures 2-13 and 2-14 on pp. 63-64 of the textbook. Afterwards,
work through a full program example, Program 2-13 on pp. 66-67, that uses variables and comments. Use Figure 2-15 on p. 67 to demonstrate a program flowchart. Encourage as much class participation in the discussion of the flow charts as time permits. (This tip refers to key concepts #1—Variable declaration, naming, and initialization, #2—Data types, and #3—Comments.)

- To prepare for the lab portion of the class using Python code, demonstrate how parts of Program 2-13 from pp. 66-67 of the textbook may be translated into Python. Ignore the constants and variable declarations as follows:

```
#Display the amount of rise in five years
fiveYears = 1.5 * 5
print ‘The ocean level will rise’, fiveYears, “millimeters in seven years.”
```

Demonstrate a simple example program that inputs both string and numeric data to do a calculation. Explain that the rawInput( ) function is used to accept string data and the input( ) function is used to input numeric data. For example:

```
#Program to compute percentile scores.

# Collect data
firstName=’’
firstName = rawInput(‘Enter your first name’)
totalPossible = input(‘Enter total marks possible’)
actualScore = input(‘Enter actual test score’)

# Compute percentile score
percentileScore = actualScore / totalPossible * 100
print firstName, ‘ your score was’, percentileScore, ‘%’
```

Make sure to mention some key rules for Python programming. Emphasize that Python is case-sensitive, ignores white space, and will take the end of a statement to be the physical end of the line (unless the \ line continuation character is used). Make sure to stress as well the importance of giving variables meaningful names that follow proper naming conventions. Refer the students to p. 39 of the textbook for variable naming rules. (This tip refers to key concepts #1—Variable declarations, naming, and initialization, #2—Data types, and #3—Comments.)

- Be sure to discuss the concept of concatenation as an operation for joining two character strings. (This tip refers to key concepts #1—Variable declarations, naming, and initialization, #2—Data types, and #3—Comments.)

- Explain to the students that the purpose of the homework assignment is to review the key concepts discussed in the class. Students should submit their answers in the next class. Answers can be found in Appendix G. (This tip refers to key
concepts #1—Variable declarations, naming, and initialization, #2—Data types, and #3—Comments.)
Homework

The following homework is designed to cover the course objectives for this unit.

Assignment 3.1: Answer the following questions from Chapter 2 of your textbook, *Starting Out with Programming Logic & Design*:

- Multiple Choice Review Questions 15-18 and 20 on p. 70
- Algorithm Workbench Review Questions 11-12 on p. 72
- Programming Exercises Question 7 on p. 73

Submit your answers in a word-processed document to your instructor in Unit 4.
Labs

Lab 3.1: Input, Processing, and Output:

Python Code
Programming Challenge 1: Team Average
Programming Challenge 2: Pedometer Calculator

What is the purpose?

In this lab, you will identify program input, process, and output components and implement them in a properly commented program code using variables.

What are the steps?

• Task 1: Lab 1.4—Python Code

  Procedure
  2. When creating your Python program in Step 6, save your file as Lab3_1.py.
  3. After ensuring your program code is complete and runs properly, submit your program code to your instructor for grading.

• Task 2: Lab 1.5—Programming Challenge 1: Team Average

  Procedure
  2. Create a solution algorithm to calculate the average.
  3. Write the pseudocode.
     o Determine the required program input, process, and output.
     o Determine the variables and variable data types required.
     o Create the pseudocode for variable declaration and data input, calculation, and program output.
  4. Using the proper flowchart symbols create a flowchart using the pseudocode as a guide.
  5. Provide the Python program code.
     o Start the IDLE environment, and save the file as Lab3_2.py.
     o Add header comments to the program.
     o Using the pseudocode you created, enter the required input statements, calculation statement, and output statement. After each statement is created, check the program to make sure it runs.
6. Before submitting your final program to your instructor for grading, make sure the program runs and supplies the proper output.

- **Task 3: Lab 1.6—Programming Challenge 2: Pedometer Calculator Procedure**
  2. Create a solution algorithm to calculate the average.
  3. Write the pseudocode.
     - Determine the required program input, process, and output.
     - Determine the variables and variable data types required.
     - Create the pseudocode for variable declaration and data input, calculation, and program output.
  4. Using the proper flowchart symbols, create a flowchart using the pseudocode as a guide.
  5. Provide the Python program code.
     - Start the IDLE environment, and save the file as Lab3_3.py.
     - Add header comments to the program.
     - Using the pseudocode you created, enter the required input statements, calculation statement, and output statement. After each statement is created, check the program to make sure it runs.
  6. Before submitting your final program to your instructor for grading, make sure the program runs and supplies the proper output.

**Did it work?**

- Were you able to create an operational program free of syntax errors?
- Were you able to create a program that produces the proper output?
- Did you submit your work to your instructor for grading?
Summary and Reminders

Summary

- This unit introduced some of the key concepts and techniques related to data storage in software programs: variable declaration, variable initialization, and data types.
- The importance of internal documentation in programs was also emphasized in this unit.
- All of the concepts introduced in the unit were implemented in program code to create working software programs.

Reminders

- Students should prepare for the next unit by reading Chapter 3, pp. 75-90, of their textbook prior to coming to class.
- Assignment 3.1 will be due in the next class period.
- Remind students that the content covered in Units 1, 2, and 3 will be assessed on Unit Exam 1, which will be administered in the next class. Material for the exam will come from the readings and notes for Chapters 1 and 2.

(End of Unit 3)
Unit 4: Program Modules

Objectives

3. Create various modules in computer programs.
   3.1: Explain the importance of separating code into modules for efficiency.
   3.2: Use flowcharting as a tool to create program modules.
   3.3: Describe the impact of program modules on variables—variable scope.

Readings


- Chapter 3, pp. 75-90

Key Concepts That Must Be Covered in Class

The following key concepts must be covered in class in order to achieve the course objectives.

1. Importance of using modules
2. Flowcharting modules
3. Variable scope-local variables

Teaching Tips for This Unit

The teaching tips are provided for your reference. Please choose carefully according to your teaching style and students’ learning style.

- During this unit, you will administer Unit Exam 1, which covers Units 1, 2, and 3. You may wish to conduct a brief review of the previous material and answer any of your students’ questions before administering the Unit Exam.

- To set the stage for Unit 4, explain to the students why modules should be used, especially since program code does not need to be written using modules at all in order to run. Explain that writing program code that uses modules is a powerful technique to structure code so that it can be created, tested, and modified in an efficient, cost-effective manner. Refer the students to p. 76 of the textbook when discussing the divide-and-conquer technique, which assists in writing modules. Be sure to discuss the benefits of program code structured this way. These benefits can be found on pp. 76-77 of the textbook. (This tip refers to key concept #1—Importance of using modules.)
• Discuss the importance of naming modules properly using descriptive names and following variable naming conventions. If necessary, review pp. 39-40 of the textbook. Remind students that Python is case-sensitive and that case matters for module names. Discuss the general structure of modules and the transfer of control when modules are called or terminated.

Use Program 3-1 on pp. 79-80 and Figures 3-2, 3-3, and 3-4 on pp. 80-81 of the textbook as an example. Note that in Python, proper indentation of program statements inside a module is critical. Additionally, the initial module does not have to be called main() but this is a good convention to follow. Show a Python implementation of the message program example as follows:

```python
def main( ):
    print ‘I have a message for you.’
    showMessage( )
    print “That’s all, folks!”

def showMessage( ):
    print ‘Hello world’

main( )
```

Do not dwell too long on the Python code example and get sidetracked into discussions about the Python environment. State that you are merely using the sample to illustrate a point. (This tip refers to key concept #2—Flowcharting modules.)

• Emphasize to the students that structuring modules correctly is essential in program design and that tools such as flowcharts can be very helpful in determining that structure. Use Figure 3-5 on p. 82 of the textbook to discuss the symbols used to depict a module. Discuss Figure 3-6 on p. 82 as an example of flowcharting modules. (This tip refers to key concept #2—Flowcharting modules.)

• Discuss the flow of control and the interaction of modules. Mention that top-down design (a.k.a. stepwise refinement) is a common technique to break a solution algorithm up into individual modules. Refer to Figure 3-7 on p. 83 of the textbook to discuss hierarchy (a.k.a. structure) charts and to depict module relationships. Use the ACME dryer repair program example found on pp. 84-88 to illustrate how an involved algorithm can be broken up into individual algorithms and constructed using a hierarchy chart (Figure 3-8 on p. 84), pseudocode (Program 3-2 on pp. 85-86), and individual flowcharts for each algorithm (Figure 3-9 on p. 87 and Figure 3-10 on p. 88). (This tip refers to key concept #2—Flowcharting modules.)
• Discuss the concept of variable scope and local variables. Use Program 3-3 on p. 89 of the textbook as an example. Use the example on p. 90 to discuss how local variables of different scope can have the same name but variables of the same scope cannot have the same name. (This tip refers to key concept #3—Variable scope-local variables.)

• Explain to the students that the purpose of the homework assignment is to review the key concepts discussed in the class. Students should submit their answers in the next class. Answers can be found in Appendix G. (This tip refers to key concepts #1—Importance of using modules, #2—Flowcharting modules, and #3—Variable scope-level variables.)

In-Class Assessment

Unit Exam 1:

Administer Unit Exam 1. Refer to Appendix A for the exam questions and answers.
Homework

The following homework is designed to cover the course objectives for this unit.

Assignment 4.1: Answer the following questions from Chapter 3 of your textbook,
Starting Out with Programming Logic & Design:

• Multiple Choice Review Questions 2, 3, 6, 8, and 9 starting on p. 109
• Short Answer Review Questions 3-4 on p. 111

Submit your answers in a word-processed document to your instructor in Unit 5.
Labs

Lab 4.1: Modules and Local Variables

What is the purpose?
In this lab, you will gain practice designing solution algorithms for programs using modules and local variables. You will also practice designing programs by using pseudocode and flowcharts, and then implementing them in the Python programming language.

What are the steps?

• Task 1: Formulating Pseudocode
  Procedure
  1. In pseudocode, design a program that consists of two modules: main() and welcomeMessage(). The main() module just calls the welcomeMessage( ) module that outputs the following messages: “Welcome to my program using functions” and “My name is Joe Student”.
  2. Add a line to the main() module to call up a module goodbyeMessage( ). Add a module goodbyeMessage( ) that prints the following message “Good bye!”
  3. Submit your pseudocode to your instructor for grading.

• Task 2: Flowcharting
  Procedure
  1. Using the Raptor application, create a flowchart for the main( ), welcomeMessage( ), and goodbyeMessage( ) modules. (Refer to Lab 2.3 on pp. 25-28 as a reference for how to create flowcharts of modules using Raptor).
  2. Save your file as Lab4-1-1.rap.
  3. Print out your flowcharts and submit your work to your instructor for grading.

• Task 3: Lab 2.4—Python Code and Functions
  Procedure
  2. Save your file as Lab4_2.py.
  3. Submit your Python program code to your instructor for grading.
• **Task 4: Formulating Pseudocode**  
  **Procedure**  
  1. In pseudocode, design a program that consists of two modules: main( ) and inputName( )  
  2. The main( ) module first outputs the following message: “Welcome to the variable program”  
  3. The main( ) module declares a string variable called name and calls the inputName( ) module to assign a value to name as follows:

        name = inputName( )

  Be sure to declare the variable name properly, first using the correct data type.  
  4. The main( ) module then uses the name variable to output a message saying hello to the name that is being stored in the name variable.  
  5. Submit your pseudocode to your instructor for grading.

• **Task 5: Flowcharting**  
  **Procedure**  
  1. Using the Raptor application, create a flowchart for the main( ) and inputName( ) modules (refer to Lab 2.3, pp. 25-28, as a reference for how to create flowcharts of modules using Raptor).  
  2. Save your file as Lab4-1-2.rap.  
  3. Print out your flowcharts and submit your work to your instructor for grading.

• **Task 6: Lab 2.5—Python Code and Variables**  
  **Procedure**  
  2. Save your file as Lab4_6.py.  
  3. Submit your Python program code to your instructor for grading.

**Did it work?**  
• Were you able to create the pseudocode in Tasks 1 and 4?  
• Were you able to create flowcharts using Raptor in Tasks 2 and 5?  
• Were you able to create the Python code in Tasks 3 and 6?  
• Did you submit your work to your instructor for grading?
Summary and Reminders

Summary

- This unit introduced using modules to structure code that can be more easily debugged and modified.
- The concepts of local variables and variable scope were also introduced in this unit.

Reminders

- Students should prepare for the next unit by reading Chapter 3, pp. 90-108, and Chapter 6, pp. 217-218 and pp. 225-231, of their textbook prior to coming to class.
- Assignment 4.1 will be due in the next class period.
- The content covered in Unit 4 will be assessed in Unit Exam 2, which will be administered in Unit 6.

(End of Unit 4)
Unit 5: Program Modules and Functions

Objectives

3. Create various modules in computer programs.
   3.4: Describe the necessity of having compatible arguments in module parameters.
   3.5: Compare and contrast the usefulness of argument passing by value and by reference.
   3.6: Evaluate the various program modules.

4. Create various functions in computer programs.
   4.1: Describe the usefulness of functions in computer programs.
   4.2: Use flowcharting as a tool to create functions.
   4.3: Describe functions using pseudocode.
   4.4: Evaluate the various functions created for use in computer programs.

Readings


- Chapter 3, pp. 90-108
- Chapter 6, pp. 217-218 and pp. 225-231

Key Concepts That Must Be Covered in Class

The following key concepts must be covered in class in order to achieve the course objectives.

1. Module parameters and necessity of argument and compatibility
2. Pass by value vs. pass by reference

Teaching Tips for This Unit

The teaching tips are provided for your reference. Please choose carefully according to your teaching style and students’ learning style.

- Explain the importance of matching the number, the order, and the data type of input arguments. Discuss the pseudocode in Figures 3-12 and 3-13 on p. 93 of the textbook to present the concept of argument compatibility.

  Discuss multiple argument passing using Program 3-6 and Figure 3-14 on p. 94.
Discuss the implications of mixing up the order of arguments (Note: The example on p. 94 works regardless of argument order so you need another example.) For example, if name and age are not specified in the correct order in the calling statement for this module, then the output will not make sense:

Module printBirthdayMessage(String name, Integer age)
// display birthday message
Display “Happy “, age , “th Birthday “, name
End Module

(This tip refers to key concept #1—Module parameters and necessity of argument and compatibility.)

- Explain to the students that even though Python is not strongly typed, you can still get syntax and logic errors in your Python code if your module expects one data type but receives a conflicting data type, such as input( ) instead of rawInput( ). (This tip refers to key concept #1—Module parameters and necessity of argument and compatibility.)

- Use Program 3-8 on p. 98 of the textbook to discuss the differences between passing arguments by value and by reference. Also reference the pseudocode example found in Program 3-9 on pp. 99-100. Note that pass by value is safer than pass by reference, but that pass by reference is more efficient in terms of memory usage and coding if the value of the argument is to be changed.

If time permits, use the program listings for the cup conversion program to illustrate how pass by reference may be used to clean up code. The original is Program 3-7 on p. 96, and is illustrated in Figure 3-15 on p. 95 and Figure 3-16 on p. 97. The revised code can be found in Program 3-10 on p. 102 and uses pass by reference. The revised program is illustrated in Figure 3-17 on p. 101 and Figure 3-18 on p. 103. Note that Python passes arguments by value. (This tip refers to key concept #2—Pass by value vs. pass by reference.)

- Explain that while most variables have a local scope, they can also have global scope. Use Program 3-11 on p. 104 of the textbook to illustrate global variables. Note that global variables can make programs difficult to debug and that passing argument values using local variables is preferred. Briefly mention that global variables are often constants and refer to Program 3-12 on pp. 106-107 as an example. Note that Python can declare global variables using the global keyword. (This tip refers to key concepts #1—Module parameters and necessity of argument and compatibility and #2—Pass by value vs. pass by reference.)

- Explain that functions are really just modules that return a value. Discuss the pseudocode in Figure 6-7 on p. 226 and Program 6-6 on p. 227 of the textbook. Note input( ) and rawInput( ) are examples of functions. Mention that for Python,
no special notation is required specifying the data type of the returned value. (This tip refers to key concept #1—Module parameters and necessity of argument and compatibility.)

- Explain to the students that the purpose of the homework assignment is to review the key concepts discussed in the class. Students should submit their answers in the next class. Answers can be found in Appendix G. (This tip refers to key concepts #1—Module parameters and necessity of argument and compatibility and #2—Pass by value vs. pass by reference.)
Homework

The following homework is designed to cover the course objectives for this unit.

Assignment 5.1: Answer the following questions from Chapter 3 of your textbook, *Starting Out with Programming Logic & Design*:

- Multiple Choice Review Questions 12-15 on p. 110
- Algorithm Workbench Review Question 2, 3, and 6 starting on p. 111

Submit your answers in a word-processed document to your instructor in Unit 6.
Labs

Lab 5.1: Modules

What is the purpose?
In this lab, you will gain practice designing solution algorithms for programs using modules stressing argument passing. You will also practice designing programs by using pseudocode and flowcharts, and then implementing them in the Python programming language.

What are the steps?

- **Task 1: Lab 2.1—Algorithms**
  **Procedure**
  2. Submit your answers to Step 2 to your instructor for grading.

- **Task 2: Lab 2.2—Pseudocode and Modules**
  **Procedure**
  2. Submit your answers to Steps 1-3 to your instructor for grading.

- **Task 3: Lab 2.3—Flowcharts**
  **Procedure**
  2. Submit your flowchart to your instructor for grading.

- **Task 4: Lab 2.6—Writing a Complete Program**
  **Procedure**
  2. Submit your completed program to your instructor for grading.

- **Task 5: Lab 2.7—Programming Challenge 1: Retail Tax**
  **Procedure**
  2. Submit your program code to your instructor for grading.
Did it work?

- Were you able to obtain the answers to Step 2 in Lab 2.1?
- Were you able to declare the variables in Step 1 of Lab 2.2?
- Were you able to declare the modules and create the pseudocode in Steps 2 and 3 of Lab 2.2?
- Were you able to create flowcharts using Raptor?
- Were you able to create the Python code in Labs 2.6 and 2.7?
- Did you submit your work to your instructor for grading?
Summary and Reminders

Summary

- This unit continued to introduce the use of modules to structure code, with special emphasis placed upon argument passing.
- The concepts of global variables and functions were also introduced in this unit.

Reminders

- Students should prepare for the next unit by reading Chapter 4, pp. 115-142, of their textbook prior to coming to class.
- Assignment 5.1 will be due in the next class period.
- Remind students that the content covered in Units 4 and 5 will be assessed on Unit Exam 2, which will be administered in the next class. Material for the exam will come from the readings and notes for Chapters 3 and 6.

(End of Unit 5)
Unit 6: Decisions I

Objectives

5. Create computer programs that can make decisions.
   5.1: Use pseudocode to represent Boolean conditions.
   5.2: Use flowcharts as a tool to represent Boolean conditions.
   5.3: Apply the concept of nesting conditions to computer programs.
   5.4: Use if-then, if-then-else, and case structures in a computer program.
   5.5: Compare strings using the program language.

Readings


- Chapter 4, pp. 115-142

Key Concepts That Must Be Covered in Class

The following key concepts must be covered in class in order to achieve the course objectives.

1. Flowcharting and pseudocode to represent Boolean conditions
2. Nesting of conditions
3. String comparisons
4. If-then and if-then-else

Teaching Tips for This Unit

The teaching tips are provided for your reference. Please choose carefully according to your teaching style and students’ learning style.

- During this unit you will administer Unit Exam 2, which covers Units 4 and 5. You may wish to conduct a brief review of the previous material and answer any of your students’ questions before administering the exam.

- Explain that decision structures enable programs to provide different responses to different conditions and consequently make them much more flexible and useful. In essence, decision structures allow software programs to appear to “think.” Use Figure 4-1 on p. 116, Figure 4-2 on p. 117, and Figure 4-3 on p. 118 of the textbook to demonstrate how simple decision structures can be represented in flowcharts. (This tip refers to key concept #1—Flowcharting and pseudocode to represent Boolean conditions.)
• Discuss the relational operators found in Tables 4-1 and 4-2 on p. 119 of the textbook. Take care to differentiate between the assignment operator = and the relational operator ==. (See the warning text on p. 120—Python behaves like C++ and Java in this regard). Show how the decision structure in Figure 4-5 on p. 121 can be represented by an if-then statement in pseudocode. Note that properly written conditions are essential to make an if-then statement work. (This tip refers to key concepts #1—Flowcharting and pseudocode to represent Boolean conditions and #4—If-then and if-then-else.)

• Discuss if-then statements in further detail. Explain that indenting if-then statements in pseudocode makes it much easier to determine what code belongs inside the if-then block. Use the pseudocode in Figure 4-6 on p. 122 of the textbook as an example. Explain that proper indenting in Python if-then statements is essential to determine what is inside the if-then block because there is no End If statement. Use a Python version of the pseudocode in Figure 4-6 on p. 122 as an example:

```python
if sales > 50000:
    bonus = 500.00
    commissionRate = 0.12
    print “You’ve met your sales quota!”
```

(This tip refers to key concepts #1—Flowcharting and pseudocode to represent Boolean conditions and #4—If-then and if-then-else.)

• Discuss if-then-else (dual alternative) statements. Use the pseudocode in Figure 4-9 on p. 126 of the textbook as an example and provide the Python version of the code as follows:

```python
if temperature < 40:
    print “A little cold, isn’t it?”
else:
    print “Nice weather we’re having.”
```

Finally, discuss the more involved pseudocode example in Program 4-2 on pp. 127-128 and discuss how it is represented in both a hierarchy chart (Figure 4-10 on p. 127) and flowcharts (Figures 4-11 on p. 129 and 4-12 on p. 130). (This tip refers to key concepts #1—Flowcharting and pseudocode to represent Boolean conditions and #4—If-then and If-then-else.)

• Explain that, just like numeric values, strings may be compared and used in conditions. Use Program 4-3 on p. 131 as an example. Explain that strings are compared based on the ASCII code values of each character in the string (Figures 4-13 and 4-14 on p. 133). Note that string comparisons in Python are case-sensitive. (This tip refers to key concept #3—String comparisons.)
• Discuss how complex comparisons can be made using nested structures and if-then-else-if statements. Use the flowchart in Figure 4-17 on p. 139 of the textbook to illustrate a grading program that requires a series of decisions to be made. Use the pseudocode in Program 4-6 on pp. 139-140 to show how this may be implemented in a nested structure. Use the pseudocode on p. 141 to illustrate how the same program can be written using an if-then-else-if structure. Note that the if-then-else-if code is simpler to read and requires less indenting. Finally, illustrate how the grading program’s if-then-else-if code can be implemented in Python:

```python
if score < 60:
    print "Your grade is F."
elif score < 70:
    print "Your grade is D."
elif score < 80:
    print "Your grade is C."
elif score < 90:
    print "Your grade is B."
else:
    print "Your grade is A."
```

(This tip refers to key concepts #1—Flowcharting and pseudocode to represent Boolean conditions, #2—Nesting of conditions, and #4—If-then and if-then-else.)

• Explain to the students that the purpose of the homework assignment is to review the key concepts discussed in the class. Students should submit their answers in the next class. Answers can be found in Appendix G. (This tip refers to key concepts #1—Flowcharting and pseudocode to represent Boolean conditions, #2—Nesting of conditions, #3—String comparisons, and #4—If-then and if-then-else.)

**In-Class Assessment**

**Unit Exam 2:**

Administer Unit Exam 2. Refer to Appendix A for the exam questions and answers.
Homework

The following homework is designed to cover the course objectives for this unit.

Assignment 6.1: Answer the following questions from Chapter 4 of your textbook, *Starting Out with Programming Logic & Design*:

- Multiple Choice Review Questions 3, 5, 7, and 8 on p. 156
- Algorithm Workbench Review Questions 3, 4, and 5 starting on p. 158

Submit your answers in a word-processed document to your instructor in Unit 7.
Labs

Lab 6.1: Decisions and Boolean Logic

What is the purpose?

In this lab, you will gain practice designing decision making structures using single alternative (if-then) and dual alternative (if-then-else) decision structures. In addition, nested if-then-else and if-then-else-if structures will also be covered. You will design decision-making structures using pseudocode and flowcharts, and then implement them in the Python programming language.

What are the steps?

- **Task 1: Lab 3.1—Evaluating Conditions**
  
  **Procedure**
  
  2. Submit your answers to Steps 2 and 3 to your instructor for grading.

- **Task 2: Lab 3.2—Pseudocode and Decisions**
  
  **Procedure**
  
  2. Submit your answers to Steps 1, 2, and 3 to your instructor for grading.

- **Task 3: Lab 3.3—Flowcharts**
  
  **Procedure**
  
  2. Submit your completed flowcharts to your instructor for grading.

- **Task 4: Lab 3.4—Python Code**
  
  **Procedure**
  
  2. Submit your completed code to your instructor for grading.

- **Task 5: Lab 4.3—Pseudocode: Nested Decision Structures**
  
  **Procedure**
  
2. Submit your pseudocode to your instructor for grading.

- **Task 6: Lab 4.4—Flowcharts**
  Procedure
  2. Submit your completed flowcharts to your instructor for grading.

- **Task 7: Lab 4.5—Python Code**
  Procedure
  2. Submit your completed code to your instructor for grading.

**Did it work?**

- Were you able to obtain the answers in Steps 2 and 3 of Lab 3.1?
- Were you able to declare the variables in Lab 3.2?
- Were you able to name the modules needed in Step 2 of Lab 3.2?
- Were you able to complete the pseudocode in Step 3 of Lab 3.2?
- Were you able to create flowcharts using Raptor?
- Were you able to create the Python code in Labs 3.4 and 4.5?
- Did you submit your work to your instructor for grading?
Summary and Reminders

Summary

• This unit introduced decision making structures using single alternative (if-then) and dual alternative (if-then-else) decision structures as well as more complex nested if-then-else and if-then-else-if structures.
• Construction of Boolean conditions, including string comparisons, was also investigated in this unit.

Reminders

• Students should prepare for the next unit by reading Chapter 4, pp. 142-155, of their textbook prior to coming to class.
• Assignment 6.1 will be due in the next class period.
• The content covered in Unit 6 will be assessed in Unit Exam 3, which will be administered in Unit 11.

(End of Unit 6)
Unit 7: Decisions II

Objectives

5. Create computer programs that can make decisions.
   5.1: Use pseudocode to represent Boolean conditions.
   5.2: Use flowcharts as a tool to represent Boolean conditions.
   5.3: Apply the concept of nesting conditions to computer programs.
   5.4: Use if-then, if-then-else, and case structures in a computer program.
   5.5: Compare strings using the program language.
   5.6: Use Boolean variables and logical operators in computer programs.
   5.7: Evaluate compound logical conditions.

Readings


- Chapter 4, pp. 142-155

Key Concepts That Must Be Covered in Class

The following key concepts must be covered in class in order to achieve the course objectives.

1. Case structures
2. Boolean variables and logical operators
3. Compound logical conditions

Teaching Tips for This Unit

The teaching tips are provided for your reference. Please choose carefully according to your teaching style and students’ learning style.

- Briefly review the decision making structures discussed in the previous class. Continue this discussion by introducing case (switch) structure. Indicate that case structures are multiple alternative decision structures that can be used instead of nested structures or if-then-else-if structures. Note that while case structures may not be as flexible as their alternatives, they can be much easier to read and debug. Use the case structure represented in Figure 4-18 on p. 142 and the pseudocode on pp. 143-144 of the textbook.

Note the alternative nested structure for the same problem on Figure 4-19 on p. 144 and the following pseudocode:
If month == 1 Then
    Display “January”
Else
    If month == 2 Then
        Display “February”
    Else
        If month == 3 Then
            Display “March”
        Else
            Display “Error: Invalid Month”
        End If
    End If
End If

Point out to the students that Python does not have a case structure; however, you can still do any decision logic that is done in a case structure in Python using a nested structure or an if-then-else-if structure. (This tip refers to key concept #1—Case structures.)

• Introduce the logical operators AND, OR, and NOT, which are listed in Table 4-3 on p. 147 of the textbook. Students typically have difficulty deciding whether to use AND or OR operators. Review thoroughly Table 4-5 on p. 148 and Table 4-6 on p. 149. In addition, review Table 4-7 on p. 150 for the operation of the NOT operator. Note that Python does not use Java/C++ notation for logical operators—instead it uses AND, OR, and NOT. (This tip refers to key concept #2—Boolean variables and logical operators.)

• Illustrate how compound Boolean conditions using the logical operators can simplify code by eliminating the need for nested structures. Review the loan qualifier program using pseudocode found on p. 150 of the textbook or Program 4-5 on p. 136 that uses a nested structure. Review the similar program shown in Program 4-8 on p. 151 that uses compound Boolean conditions and no nesting. (This tip refers to key concepts #2—Boolean variables and logical operators, and #3—Compound logical conditions.)

• Demonstrate how using compound Boolean conditions can also be very useful to specify ranges of values as shown in the pseudocode on p. 153 of the textbook. (This tip refers to key concepts #2—Boolean variables and logical operators and #3—Compound logical conditions.)

• Use the pseudocode examples on pp. 154-155 of the textbook to demonstrate the value of Boolean variables as flag variables to signal if a certain condition has been met in a program. (This tip refers to key concept #2—Boolean variables and logical operators.)
• Explain to the students that the purpose of the homework assignment is to review the key concepts discussed in the class. Students should submit their answers in the next class. Answers can be found in Appendix G. (This tip refers to key concepts #1—Case structures, #2—Boolean variables and logical operators, and #3—Compound logical conditions.)
Homework

The following homework is designed to cover the course objectives for this unit.

Assignment 7.1: Answer the following questions from Chapter 4 of your textbook, *Starting Out with Programming Logic & Design*:

- Multiple Choice Review Questions 10 and 12-15 on p. 157
- Algorithm Workbench Review Questions 6-8 and 10 on p. 159

Submit your answers in a word-processed document to your instructor in Unit 8.
Labs

Lab 7.1: Decisions and Boolean Logic

What is the purpose?

In this lab, you will practice designing decision-making structures using case structures, compound Boolean expressions, and Boolean variables. You will design decision-making structures using pseudocode and flowcharts, and then implement them in the Python programming language.

What are the steps?

- **Task 1: Lab 3.5—Programming Challenge 1: Guess the Secrets**
  Procedure
  2. Submit your work to your instructor for grading.

- **Task 2: Lab 4.1—Logical Operators and Dual Alternative Decisions**
  Procedure
  2. Submit your answer for Steps 2 and 3 to your instructor for grading.

- **Task 3: Lab 4.2—Pseudocode: Dual Alternative Decisions**
  Procedure
  2. Submit your pseudocode to your instructor for grading.

- **Task 4: Lab 4.6—Programming Challenge 1: Tip, Tax, and Total**
  Procedure
  2. Submit your work to your instructor for grading.
Did it work?

- Were you able to create the pseudocode, flowchart, and Python code in Lab 3.5?
- Where you able to obtain the answers to Steps 2 and 3 in Lab 4.1?
- Where you able to complete the pseudocode in Lab 4.2?
- Were you able to create the pseudocode, flowchart, and Python code in Lab 4.6?
- Did you submit your work to your instructor for grading?
Summary and Reminders

Summary

- This unit introduced decision-making structures using case structures, compound Boolean statements, and Boolean variables.

Reminders

- Students should prepare for the next unit by reading Chapter 5, pp. 163-183 and pp. 196-201, of their textbook prior to coming to class.
- Assignment 7.1 will be due in the next class period.
- The content covered in Unit 7 will be assessed in Unit Exam 3, which will be administered in Unit 11.

(End of Unit 7)
Unit 8: Repetitive Processing I

Objectives

6. Create computer programs that can do repetitive processing.
   6.1: Use pseudocode to represent repetition structures.
   6.2: Use flowcharts as a tool to represent repetition structures.
   6.3: Create the While, Do-While, and Do-Until conditional loops.
   6.4: Describe the implications of an infinite loop.

Readings


- Chapter 5, pp. 163-183 and pp. 196-201

Key Concepts That Must Be Covered in Class

The following key concepts must be covered in class in order to achieve the course objectives.

1. Flowcharting and pseudocode to represent loops
2. While, Do-While, and Do-Until loops
3. Infinite loop implications

Teaching Tips for This Unit

The teaching tips are provided for your reference. Please choose carefully according to your teaching style and students’ learning style.

- Since this unit introduces the concept of using loops to do repetitive processing, you should first explain why such structures are useful. A repetition structure has benefits over just coding the same statements many times over in terms of ease of coding, modification, and readability. Use the disadvantages identified on the top of p. 164 of the textbook for your explanation. (This tip refers to key concepts #1—Flowcharting and pseudocode to represent loops, #2—While, Do-While, and Do-Until loops, and #3—Infinite loop implications.)

- Introduce the two basic types of loops: counter-controlled and condition-controlled. Explain that under certain conditions, you know exactly how many times to repeat something ahead of time, making counter-controlled loops preferable in this case. On the other hand, in some cases (for example, displaying a user menu until the user selects the command to exit the menu), it cannot be
determined exactly how many repetitions you will need. You can, however, determine what condition should stop the repetition, making condition-controlled loops preferable in this case. Explain that the majority of this unit covers condition-controlled loops. (This tip refers to key concept #2—While, Do-While, and Do-Until loops.)

- Use Figure 5-1 on p. 165 of the textbook to discuss how While loops are represented in flowcharts. Using the code outlined on p. 165, go over the general format of While loops in pseudocode. Explain that a While loop is a pretest loop where the test condition is tested at the start of the loop. Stress that if the condition for the loop is initially false, not a single iteration of the loop will be made. Review the sales commission program, Program 5-1 on p. 166, Figure 5-2 on p. 167, and Figure 5-3 on p. 168. Provide a Python code implementation of the While loop in Figure 5-2 on p. 167 as follows:

```python
while keepGoing = "y":
    #Get the amount of sales.
    print "Enter the amount of sales."
    sales = input()
    #Calculate the commission
    commission = sales * CommissionRate
    #Display the commission
    print "The commission is $", commission
    print "Do you want to calculate another"
    keepGoing = input("commission? (Enter y for yes).")
```

Note that in Python there is no End While statement and that indentation is what determines the while block. (This tip refers to key concepts #1—Flowcharting and pseudocode to represent loops and #2—While, Do-While, and Do-Until loops.)

- Discuss the need for coming up with a way to terminate loops so an infinite loop is not created. Stress that infinite loops not only waste system resources, they also prevent the program from continuing with any new processing. Use the revised sales commission program in Program 5-3 on p. 172 of the textbook as an example of an infinite loop. (This tip refers to key concept #3—Infinite loop implications.)

- Discuss how putting most of the code that is repeated inside the body of a loop in a module simplifies the loop code and makes it easier to catch any looping errors. Use the modular version of the commission program in Program 5-4 on p. 172, Figure 5-5 on p. 173, and Figure 5-6 on p. 174 of the textbook as examples. (This tip refers to key concepts #1—Flowcharting and pseudocode to represent loops, and #2—While, Do-While, and Do-Until loops.)
• Explain that the Do-While loop is a posttest structure and consequently will always complete at least one loop iteration—even if the looping condition is false. Use the Do-While version of the commission program in Program 5-5 on p. 176 and Figure 5-8 on p. 177 of the textbook as an example. Explain that there are no Do-While loops in Python, but that since Do-While loops can also be expressed as While loops, this is not an issue. (This tip refers to key concepts #1—Flowcharting and pseudocode to represent loops and #2—While, Do-While, and Do-Until loops.)

• Discuss another posttest loop—the Do-Until loop. Mention that a Do-Until loop is similar to a Do-While loop but continues looping until the condition becomes true instead of while the condition is true. Mention that any Do-While condition can become a Do-Until condition by using the NOT operator. Review the flowchart in Figure 5-10 on p. 180 of the textbook and use the general pseudocode on p. 181 to discuss the structure of the Do-Until loop. Discuss the password program in Program 5-7 on p. 181 and Figure 5-11 on p. 182 as an example of a Do-Until loop. Explain that there are no Do-Until loops in Python but since Do-Until loops can be expressed as While loops, this is not an issue. (This tip refers to key concepts #1—Flowcharting and pseudocode to represent loops and #3—Infinite loop implications.)

• Discuss that although a posttest loop such as a Do-While or Do-Until has the advantage in some cases of automatically allowing at least one loop iteration, all condition-controlled loops can be written as While loops. (This tip refers to key concepts #1—Flowcharting and pseudocode to represent loops and #2—While, Do-While, and Do-Until loops.)

• Introduce the concept of using a counter variable with a While loop to create counter-controlled loops. Remind students that counter-controlled loops are very useful if dealing with situations that require a set number of iterations—for instance, processing payroll for a given number of days in a pay period. Use the pseudocode on p. 196 of the textbook to explain the mechanics of initializing and incrementing a counter variable inside a While loop. Note that failing to increment a counter variable properly will cause an infinite loop (see the warning text on p. 197), as will improper variable initialization or condition specification. Run through Program 5-14 on p. 197 and go over the critical pieces of the While loop as shown in Figure 5-17 on p. 197. (This tip refers to key concepts #1—Flowcharting and pseudocode to represent loops and #3—Infinite loop implications.)

• Briefly go over incrementing counters by values other than 1. Note that sometimes this is desirable to do things such as coloring every other square on a game board or counting backwards. Review Program 5-10 on p. 191 and Program 5-11 on p. 192 of the textbook. (This tip refers to key concepts #1—Flowcharting and pseudocode to represent loops and #3—Infinite loop implications.)
• Explain to the students that the purpose of the homework assignment is to review the key concepts discussed in class. Students should submit their answers in the next class. Answers can be found in Appendix G. (This tip refers to key concepts #1—Flowcharting and pseudocode to represent loops, #2—While, Do-While, and Do-Until loops, and #3—Infinite loop implications.)
Homework

The following homework is designed to cover the course objectives for this unit.

Assignment 8.1: Answer the following questions from Chapter 5 of your textbook, *Starting Out with Programming Logic & Design*:

- Multiple Choice Review Questions 1-5 and 7 starting on p. 211
- Algorithm Workbench Review Questions 1, 2, 7, and 8 starting on p. 213

Submit your answers in a word-processed document to your instructor in Unit 9.
Labs

Lab 8.1: Repetition Structures

What is the purpose?

In this lab, you will practice designing condition-controlled loop structures, including While and Do-While loops. You will design condition-controlled loop structures using pseudocode and flowcharts, and then implement them in the Python programming language.

What are the steps?

- **Task 1: Lab 5.1—Repetition Structures Pseudocode: Condition-controlled Loops**
  
  **Procedure**
  2. Submit your pseudocode to your instructor for grading.

- **Task 2: Lab 5.2—Repetition Structures Pseudocode: Counter-controlled Loops**
  
  **Procedure**
  2. Submit your pseudocode to your instructor for grading.

- **Task 3: Lab 5.3—Flowcharts**
  
  **Procedure**
  2. Submit your completed flowchart to your instructor for grading.

- **Task 4: Lab 5.4—Python Code**
  
  **Procedure**
  2. Submit your completed code to your instructor for grading.

- **Task 5: Lab 5.5—Programming Challenge: Yum Yum Burger Joint**
  
  **Procedure**
2. Submit your work to your instructor for grading.

Did it work?

- Were you able to create the pseudocode in Lab 5.1?
- Were you able to create the pseudocode in Lab 5.2?
- Were you able to create the flowchart using Raptor in Lab 5.3?
- Were you able to create the Python code in Lab 5.4?
- Did you submit your work to your instructor for grading?
Summary and Reminders

Summary

- This unit introduced repetition structures and focused primarily on condition-controlled loop structures.
- While, Do-While, and Do-Until loops were covered in this unit, as well as modularization of loop block code and the dangers of infinite loops.

Reminders

- Students should prepare for the next unit by reading Chapter 5, pp. 183-211, of their textbook prior to coming to class.
- Assignment 8.1 will be due in the next class period.
- The content covered in Unit 8 will be assessed in Unit Exam 3, which will be administered in Unit 11.

(End of Unit 8)
Unit 9: Repetitive Processing II

Objectives

6. Create computer programs that can do repetitive processing.
   6.5: Evaluate the counter-controlled For loops.
   6.6: Use sentinel values in creating computer programs.
   6.7: Evaluate nested loops.

Readings


- Chapter 5, pp. 183-211

Key Concepts That Must Be Covered in Class

The following key concepts must be covered in class in order to achieve the course objectives.

1. For loops
2. Sentinel values
3. Nested loops
4. Designing comprehensive program software with input validation, decision making, and repetition

Teaching Tips for This Unit

The teaching tips are provided for your reference. Please choose carefully according to your teaching style and students’ learning style.

- Review the usefulness of counter-controlled repetition structures for programs that require a set number of reiterations of a process. Introduce the concept of a For loop using a counter variable. Review the three-step process listed on p. 184 of the textbook that a For loop iteration goes through. Use Figure 5-12 on p. 184 and Figure 5-13 on p. 185 to illustrate the For loop process. (This tip refers to key concept # 1—For loops.)

- Present the pseudocode for a For loop using the sample code found on p. 186 of the textbook. Discuss the “Hello world” program as illustrated in Program 5-8 on p. 186 and Figure 5-14 on p. 187. Present what the “Hello world” program could look like implemented in Python, as follows:
for counter in range(5):
    print 'Hello World'

Note that just like a While loop in Python, indentation of the loop body is essential because there is no End For statement. Illustrate another example of the same loop in Python specifying a starting value for the range function:

for counter in range(1,6):
    print 'Hello World'

Note that when a starting value is specified in range( ), the loop ends just before the ending value. Finally, illustrate that this loop could also have been written by explicitly specifying the counter values:

for counter in [1,2,3,4,5]:
    print 'Hello World'

Note that all three versions of the Python code are equivalent, but the later versions give the programmer increasingly more control over the counter values. (This tip refers to key concept # 1—For loops.)

• Discuss looping by increments other than one in a For loop. Remind students that this is useful for such tasks as processing every other item in a list or counting backwards. Review the basic structure of the pseudocode For loop that uses a “Step” option as illustrated in the pseudocode at the bottom of p. 190 and Program 5-10 on p. 191 of the textbook. Review the pseudocode to decrement the counter (see example at the bottom of p. 193). Illustrate the Python version of the countdown example:

for counter in range(10,0,-1):
    print counter

(This tip refers to key concept # 1—For loops.)

• Use Program 5-18 on pp. 202-203, Figure 5-18 on p. 202, and Figure 5-19 on p. 204 of the textbook as illustrations of some of the applications of For loops. Note that although the example uses a For loop, a While loop could also be used. As a review of While loops, get the students to convert the For loop of Program 5-19 on pp. 206-207 into a While loop. Note that for any situation that a For loop is used, a While loop can be used to replace it. (This tip refers to key concept #1—For loops.)

• Introduce the concept of sentinel values for use in exiting While loop iterations. Take care to stress that sentinel values must be unique and must never be a regular value. Use the property tax program, Program 5-19 on pp. 206-207, and
Figure 5-20 on p. 208 of the textbook to illustrate the use of a sentinel value. (This tip refers to key concept #2—Sentinel values.)

- Introduce the concept of nested loops. As this is the most complex topic that the students have come across so far, take extra time to discuss this if necessary. Use the pseudocode on p. 209 and Figure 5-21 on p. 210 of the textbook to trace the operation of the clock simulator program. Note that nested loops can be any combination of While and For loops. If time permits, have the students convert some of the For loops into While loops in the clock simulator program to illustrate the point. (This tip refers to key concept #3—Nested loops.)

- **IMPORTANT** The students will begin their Comprehensive Lab Practicum in the next unit. Photocopy and distribute the details of the Comprehensive Lab Practicum that can be found in Appendix H. Note: The students will not begin their Comprehensive Lab Practicum until the next unit. However, at this time discuss the project, and divide the class into teams. It is recommended that you divide the class into groups of three students—each of the three modules the student teams can choose from have three components and this will allow each student to perform one of the tasks in the module. It is recommended that you consider grouping students of differing abilities.

At this time share with the students your evaluation criteria for the Comprehensive Lab Practicum. Be sure to answer any student concerns regarding how the team Comprehensive Lab Practicum will be assessed.

After you have divided the class into teams, allow the teams to have approximately 15 minutes to brainstorm about which module they will select and which team member will do which task, and answer any questions that they have. Encourage students to select project topics that relate to their field of study and even design their own projects. Stress that all topics must be approved by you before the teams continue with their project and must include input validation, conditional logic, and repetition. You should carefully check that student projects meet all requirements and are not too ambitious.

During this brainstorming session, and the actual lab practicum, be sure to keep the students focused on the tasks in the descriptions. Some students may wish to go beyond the requirements, and consequently may not have time to finish their project. (This tip refers to key concept #4—Designing comprehensive program software with input validation, decision making, and repetition.)

- Explain to the students that the purpose of the homework assignment is to review the key concepts discussed in the class. Students should submit their answers in the next class. Answers can be found in Appendix G. (This tip refers to key concepts #1—For loops, #2—Sentinel values, and #3—Nested loops.)
Homework

The following homework is designed to cover the course objectives for this unit.

Assignment 9.1: Answer the following questions from Chapter 5 of your textbook, *Starting Out with Programming Logic & Design*:

- Multiple Choice Review Questions 6, 9, and 10 starting on p. 211
- Algorithm Workbench Review Questions 3, 4, 6, 9, and 10 starting on p. 213

Submit your answers in a word-processed document to your instructor in Unit 10.
Labs

Lab 9.1: Repetition Structures

What is the purpose?
In this lab, you will practice designing condition-controlled and counter-controlled loop structures using While and For loops. You will design While and For loop structures using pseudocode and flowcharts, and then implement them in Python.

What are the steps?

• Task 1: Lab 6.1—For Loop and Pseudocode
  Procedure
  2. Submit your work to your instructor for grading.

• Task 2: Lab 6.2—For Loop and Flowcharts
  Procedure
  2. Submit your flowchart to your instructor for grading.

• Task 3: Lab 6.3—Python Code
  Procedure
  2. Submit your completed code to your instructor for grading.

• Task 4: Lab 6.4—Programming Challenge 1: Average Test Scores
  Procedure
  2. Submit your work to your instructor for grading.

Did it work?

• Were you able to obtain the answers in Lab 6.1?
• Were you able to create the flowchart using Raptor in Lab 6.2?
• Where you able to create the Python code in Lab 6.3?
• Were you able to create the Python code in Lab 6.4?
• Were you able to create the pseudocode, flowchart, and Python code in Lab 6.5?
• Did you submit your work to your instructor for grading?
Summary and Reminders

Summary

- This unit introduced counter-controlled repetition with For loops.
- In addition, sentinel values and nested loop structures were investigated in this unit.

Reminders

- Students should prepare for the next unit by reading Chapter 7, pp. 257-266, of their textbook prior to coming to class.
- Assignment 9.1 will be due in the next class period.
- The content covered in Unit 9 will be assessed in Unit Exam 3, which will be administered in Unit 11.

(End of Unit 9)
Unit 10: Repetitive Processing III and Comprehensive Lab

Objectives

7. Create computer programs that can do input validation.
   7.1: Explain the importance of input validation.
   7.2: Code input validation loops.

8. Create program software that incorporates modules (and/or functions), conditional logic, looping, and input validation.

9. Test program software.
   9.1: Demonstrate problem-solving techniques used in testing software.
   9.2: Identify methods used to debug program software.

Readings


- Chapter 7, pp. 257-266

Key Concepts That Must Be Covered in Class

The following key concepts must be covered in class in order to achieve the course objectives.

1. Importance of input validation
2. How to implement input validation using loops
3. Designing comprehensive program software with input validation, decision making, and repetition

Teaching Tips for This Unit

The teaching tips are provided for your reference. Please choose carefully according to your teaching style and students’ learning style.

- Discuss the importance of proper input for a software program to provide proper output. Use Program 7-1 on pp. 257-258 of the textbook to illustrate the garbage-in-garbage-out (GIGO) concept. Discuss the need for input validation. (This tip refers to key concept #1—Importance of input validation.)
• Use Figure 7-1 on p. 259 and the pseudocode on pp. 259-260 of the textbook to introduce the concept of an input validation loop (error trap or error handler). Use Program 7-2 on p. 261 as an example to discuss the importance of a priming read to get the initial input to test and the usefulness of compound Boolean conditions to test conditions. Discuss pseudocode examples of yes/no input and password validation code found on pp. 264-265 to illustrate how strings can be validated. (This tip refers to key concept #2—How to implement input validation using loops.)

• Introduce the concept of defensive programming and creating programs that avoid anticipated errors. Discuss the importance of determining if input is of the correct type, not empty, and accurate. Discuss in class the examples of input validation listed in the textbook on p. 266. (This tip refers to key concept #2—How to implement input validation using loops.)

• During the previous unit, the student teams for the Comprehensive Lab Practicum brainstormed about which project they would be completing and which team member would be responsible for which component. Before having the student teams begin working on their Comprehensive Lab Practicum, clarify any remaining questions the students may have. Again, remind the students to stay focused on the tasks identified in the Comprehensive Lab Practicum details. (This tip refers to key concept #3—Designing comprehensive program software with input validation, decision making, and repetition.)
Comprehensive Lab Practicum

Allow the students to begin working on their Comprehensive Lab Practicum. Refer to Appendix H for the Lab Practicum details.
Summary and Reminders

Summary

- This unit introduced the importance of input validation in programs and how input validation can be implemented using loops.
- In addition, the design of program software using input validation, decision making, and repetition was also discussed in this unit.

Reminders

- Remind students that the content covered in Units 6 through 10 will be assessed on Unit Exam 3, which will be administered in the next class. Material for the exam will come from the readings and notes for Chapter 4, 5, and 7.
- Remind the students that they will continue to work on their Comprehensive Lab Practicum in the next class.

(End of Unit 10)
Unit 11: Course Review and Comprehensive Lab

Course Review

Now that this course is nearing completion, it’s time to review the important concepts with the students. Here is an approach to do this. Feel free to adopt any other approach that works for you:

a. Start the class by congratulating them for successful participation in the course. You may want to recognize the commitment and hard work of some students.

b. Ask the students to enumerate the takeaways from each unit. Give them a chance to speak about their experiences—the concepts that were easy or difficult for them.

c. For each unit, summarize the key concepts and learning so that any important point is not missed. You can refer to the unit summaries for a list of points that should be covered.

In-Class Assessment

Unit Exam 3:

Administer Unit Exam 3. Refer to Appendix A for the exam questions and answers.

Comprehensive Lab Practicum

Allow the students to complete their Comprehensive Lab Practicum. Refer to Appendix H for the Lab Practicum details.

(End of Unit 11)
Appendix A: Test and Answer Keys

Unit Exam 1

DATE: __________________________________________
STUDENT NAME: ________________________________
COURSE NUMBER: ______________________________
INSTRUCTOR: ________________________________
SCHOOL LOCATION: __________________________

General Instructions:

1. This is a closed-book, closed-notes exam. No reference material (including assignments and labs) will be permitted for use during the exam session.
2. The exam contains the following types of questions:
   o Multiple choice
3. Please use the separate answer sheet provided to you for marking your answers.
Unit Exam 1 Questions

1. In order for a software program to run, its instructions and data must be translated into the _______ format.
   a. ASCII  
   b. Unicode  
   c. binary  
   d. tertiary

2. Programs that make a computer useful for everyday tasks such as word processing are known as _________.
   a. system software  
   b. application software  
   c. operation systems  
   d. None of the above

3. Which of the following is not an example of operating system software?
   a. Microsoft Word  
   b. Windows Vista  
   c. Linux  
   d. Mac OS X

4. Compared to an interpreted program, a compiled program usually _______.
   a. executes slower  
   b. executes faster  
   c. translates more easily to other platforms  
   d. requires more CPU processing time

5. In a high-level programming language, an interpreter _______ the instructions.
   a. translates  
   b. compiles and translates  
   c. executes  
   d. translates and executes
6. Consider the following pseudocode (Note: Roman numerals are line numbers):

   i. Display taxDue
   ii. Set taxDue = price * quantity * taxRate
   iii. Input price
   iv. Input quantity
   v. Set taxRate = 0.17

   The above pseudocode is ______.
   
   a. correct
   b. incorrect—the order of statements should be iii, iv, i, v, and ii
   c. incorrect—the order of statements should be iii, iv, v, ii, and i
   d. incorrect—you do not need line ii

7. Consider the following pseudocode (Note: Roman numerals are line numbers):

   i. Set quantity = 5
   ii. Set quantity = 7
   iii. Set price = 10
   iv. Set price * quantity = total
   v. Display price, quantity, total

   Which of the following statements is true?

   a. Line iv is incorrect.
   b. Line ii is illegal.
   c. Lines i, ii, and iii are incorrect.
   d. The pseudocode is correct.

8. The term used for a set of rules that must be strictly followed when writing a program is known as ______.

   a. semantics
   b. syntax
   c. key words
   d. punctuations
9. Consider the following pseudocode (Note: Roman numerals are line numbers):

   i. Input month
   ii. Display month
   iii. Declare Integer month

   In the above pseudocode, ______.

   a. line ii should be before line i
   b. line iii should be before line i
   c. Both a and b above
   d. None of the above

10. The following is an example of an instruction written in which computer language?

    10110000

    a. C++
    b. C#
    c. Assembly language
    d. Machine language

11. A name that represents a value that cannot be changed during a program’s execution is called ______.

    a. a real value
    b. an uninitialized value
    c. a fixed value
    d. a named constant

12. The coding scheme that contains a set of 128 numeric codes used to represent characters in a computer’s memory is ______.

    a. ASCII
    b. ENIAC
    c. binary numbering
    d. Unicode

13. The smallest storage location in a computer’s memory is known as a _____.

    a. bit
    b. byte
    c. word
    d. letter
14. The hard disk drive is a secondary storage device that stores data by _______________ encoding it onto circular disks.

a. electrically  
b. magnetically  
c. optically  
d. None of the above

15. Consider the following pseudocode (Note: Roman numerals are line numbers):

i. Declare Integer result
ii. Set result = 6 + 15 / 3
iii. Display “Result is “, result

The output from the above pseudocode is:

a. Result is 7  
b. Result is 11  
c. Compiler error—line iii incorrect  
d. Compiler error—line ii should go before line i

16. A parallelogram in a flowchart is used to denote __________.

a. input and output  
b. start and end  
c. assignments  
d. None of the above

17. Which error produces incorrect results but does not prevent the program from running?

a. syntax  
b. logic  
c. grammatical  
d. human

18. The informal language that programmers use to create models of programs that have no syntax rules and are not meant to be compiled or executed is called __________.

a. algorithm  
b. program  
c. flowchart  
d. pseudocode
19. The diagram that graphically depicts the steps that take place in a program is a ________.
   a. algorithm  
b. program  
c. flowchart  
d. pseudocode

20. A set of statements that execute in the order they appear is a ______ structure.
   a. control  
b. repetition  
c. decision  
d. sequence

21. Which mathematical operator is used to raise five to the second power?
   a. *  
b. ^  
c. /  
d. ~

22. The value of the expression 12 - 4 * 3 / 2 + 9 is ________.
   a. 21  
b. 15  
c. -6  
d. 2.18

23. Consider the following pseudocode:

   Declare Real score

   The above pseudocode is an example of a ____ where Real is a _____ and score is a _____.
   a. data declaration, variable name, data value  
b. variable initialization, data value, variable name  
c. data initialization, data value, variable name  
d. variable definition, data type, variable name
24. The following is an example of what type of statement?

Set rate = 5.75

a. Declaration  
b. Assignment  
c. Input  
d. Output

25. What symbol is used to mark the beginning and end of a string?

a. Slash  
b. Asterisk  
c. Quotation Mark  
d. Comma
Unit Exam 1 Answer Sheet

DATE: ______________________________________
STUDENT NAME: ____________________________
COURSE NUMBER: ____________________________
INSTRUCTOR: ________________________________
SCHOOL LOCATION: __________________________

General Instructions:

To answer each question, please darken the circle corresponding to your choice of answer by using a pencil. If you decide to change your answer, please erase your original answer completely and darken your new answer.

<table>
<thead>
<tr>
<th>Question No.</th>
<th>(a)</th>
<th>(b)</th>
<th>(c)</th>
<th>(d)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Question No.</td>
<td>(a)</td>
<td>(b)</td>
<td>(c)</td>
<td>(d)</td>
</tr>
<tr>
<td>-------------</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
</tr>
<tr>
<td>21</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>22</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>23</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>24</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>25</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Unit Exam 1 - ANSWER KEY

<table>
<thead>
<tr>
<th>Question Number</th>
<th>Answer Options</th>
<th>Correct Answer</th>
<th>Course Objective Tested</th>
<th>Reference in Course</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>abcd</td>
<td>c</td>
<td>1</td>
<td>Introduction to Computer Programming</td>
</tr>
<tr>
<td>2.</td>
<td>abcd</td>
<td>b</td>
<td>1</td>
<td>Introduction to Computer Programming</td>
</tr>
<tr>
<td>3.</td>
<td>abcd</td>
<td>a</td>
<td>1</td>
<td>Introduction to Computer Programming</td>
</tr>
<tr>
<td>4.</td>
<td>abcd</td>
<td>b</td>
<td>1</td>
<td>Introduction to Computer Programming</td>
</tr>
<tr>
<td>5.</td>
<td>abcd</td>
<td>d</td>
<td>1</td>
<td>Introduction to Computer Programming</td>
</tr>
<tr>
<td>6.</td>
<td>abcd</td>
<td>c</td>
<td>2</td>
<td>Unit 2 – Software Program Design Lab 2.1: Task 2</td>
</tr>
<tr>
<td>7.</td>
<td>abcd</td>
<td>a</td>
<td>2</td>
<td>Unit 1—Functional Concepts Lab 1.1: Task 3</td>
</tr>
<tr>
<td>8.</td>
<td>abcd</td>
<td>b</td>
<td>1</td>
<td>Introduction to Computer Programming</td>
</tr>
<tr>
<td>9.</td>
<td>abcd</td>
<td>b</td>
<td>2</td>
<td>Unit 2 – Software Program Design Lab 2.1: Task 2</td>
</tr>
<tr>
<td>10.</td>
<td>abcd</td>
<td>d</td>
<td>1</td>
<td>Introduction to Computer Programming</td>
</tr>
<tr>
<td>Question Number</td>
<td>Answer Options</td>
<td>Correct Answer</td>
<td>Course Objective Tested</td>
<td>Reference in Course</td>
</tr>
<tr>
<td>-----------------</td>
<td>----------------</td>
<td>----------------</td>
<td>-------------------------</td>
<td>---------------------</td>
</tr>
<tr>
<td>11.</td>
<td>a b c d</td>
<td>d</td>
<td>2</td>
<td>Input, Processing, and Output</td>
</tr>
<tr>
<td>12.</td>
<td>a b c d</td>
<td>a</td>
<td>1</td>
<td>Introduction to Computer Programming</td>
</tr>
<tr>
<td>13.</td>
<td>a b c d</td>
<td>a</td>
<td>1</td>
<td>Introduction to Computer Programming</td>
</tr>
<tr>
<td>14.</td>
<td>a b c d</td>
<td>b</td>
<td>1</td>
<td>Introduction to Computer Programming</td>
</tr>
<tr>
<td>15.</td>
<td>a b c d</td>
<td>b</td>
<td>2</td>
<td>See Unit 3 Lab Task 2</td>
</tr>
<tr>
<td>16.</td>
<td>a b c d</td>
<td>a</td>
<td>2</td>
<td>Input, Processing, and Output</td>
</tr>
<tr>
<td>17.</td>
<td>a b c d</td>
<td>b</td>
<td>2</td>
<td>Input, Processing, and Output</td>
</tr>
<tr>
<td>18.</td>
<td>a b c d</td>
<td>d</td>
<td>2</td>
<td>Input, Processing, and Output</td>
</tr>
<tr>
<td>19.</td>
<td>a b c d</td>
<td>c</td>
<td>2</td>
<td>Input, Processing, and Output</td>
</tr>
<tr>
<td>20.</td>
<td>a b c d</td>
<td>d</td>
<td>2</td>
<td>Input, Processing, and Output</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Chapter/Unit Name</th>
<th>Page #</th>
<th>Topic Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>62 Named Constants</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10 Storing Characters</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7 How Computers Store Data</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5 Secondary Storage Devices</td>
<td></td>
<td></td>
</tr>
<tr>
<td>See Unit 3 Lab Task 2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>32 Flowcharts</td>
<td></td>
<td></td>
</tr>
<tr>
<td>32 Designing a Program</td>
<td></td>
<td></td>
</tr>
<tr>
<td>30 Pseudocode</td>
<td></td>
<td></td>
</tr>
<tr>
<td>32-33 Flowcharts</td>
<td></td>
<td></td>
</tr>
<tr>
<td>36 Sequence Structure</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Question Number</td>
<td>Answer Options</td>
<td>Correct Answer</td>
</tr>
<tr>
<td>-----------------</td>
<td>----------------</td>
<td>----------------</td>
</tr>
<tr>
<td>21.</td>
<td>a b c d</td>
<td>b</td>
</tr>
<tr>
<td>22.</td>
<td>a b c d</td>
<td>b</td>
</tr>
<tr>
<td>23.</td>
<td>a b c d</td>
<td>d</td>
</tr>
<tr>
<td>24.</td>
<td>a b c d</td>
<td>b</td>
</tr>
<tr>
<td>25.</td>
<td>a b c d</td>
<td>c</td>
</tr>
</tbody>
</table>
Unit Exam 2

DATE: ____________________________

STUDENT NAME: ____________________________

COURSE NUMBER: ____________________________

INSTRUCTOR: ____________________________

SCHOOL LOCATION: ____________________________

General Instructions:

1. This is a closed-book, closed-notes exam. No reference material (including assignments and labs) will be permitted for use during the exam session.
2. The exam contains the following types of questions:
   ○ Multiple choice
3. Please use the separate answer sheet provided to you for marking your answers.
Unit Exam 2 Questions

1. “Function” is another name for _____.
   a. a module
   b. a variable
   c. a reference
   d. a sub class

2. Consider the following pseudocode:

Module main()
    Declare String first
    Declare String last
    first = “ ”
    last = “ ”
    load(first, last)
    Display “Name is “, first, last
End Module

Module load(String first, String last)
    first = “Joe”
    last = “Black”
End Module

The output should be:
   a. Compiler error
   b. Name is Joe Black
   c. Name is
   d. None of the above
3. Consider the following pseudocode:

Module main( )
    Call getScore( )
    Display “Your score is” score
End Module

Module getScore( )
    Declare Integer score
    score = 99
End Module

What will be the output?

a. Your score is 99
b. Module
   Your score is 99
   End Module
c. Your score is score
d. None of the above

4. The module definition comprises the module header and the module _____.

   a. statements
   b. arguments
   c. parameters
   d. body

5. The following is an example of a module _________.

   Call showNetPay( )

   a. definition
   b. display
   c. execution
   d. header

6. A function is a module that _________.

   a. can never accept arguments
   b. must always return a value
   c. cannot have local variables
   d. cannot have global variables
7. A local variable is a ______.
   a. variable that is declared inside a module
   b. variable that can be used in all modules
   c. variable that must be constant
   d. None of the above

8. If module arguments are passed by value, ______.
   a. only a copy of the data is passed to the parameter
   b. the parameter can be used to change the original variable’s data
   c. variable scope is global
   d. All of the above

9. The statement that appears immediately after the module call is known as the ________.
   a. return address
   b. return point
   c. resume point
   d. resume address

10. In a flowchart, a rectangle with vertical bars depicts ______.
    a. a hierarchy chart
    b. a module call
    c. global variable scope
    d. local variable scope

11. The starting point of a module is indicated by the ________.
    a. declaration
    b. definition
    c. header
    d. body

12. What tool would a programmer use to visualize the relationship between modules?
    a. Flowchart
    b. Hierarchy chart
    c. Pseudo chart
    d. Modular chart
13. What is the term used for the variable that receives an argument that is passed into a module?
   a. Global  
   b. Local  
   c. Parameter  
   d. Constant

14. Which type of variable is visible to every module and the entire program?
   a. Local  
   b. Pass by value  
   c. Reference  
   d. Global

15. Which type of variable is not recommended to be used in programs because it makes programs hard to understand and debug?
   a. Local  
   b. Pass by value  
   c. Reference  
   d. Global

16. When an argument is passed by ________, it is not affected by a change of the content of the parameter variable.
   a. value  
   b. constant  
   c. reference  
   d. variable

17. Variables that are useful for establishing two-way communication between modules are known as ______ variables.
   a. value  
   b. constant  
   c. reference  
   d. I/O

18. What phrase is placed in the starting terminal symbol of a module in a flowchart?
   a. Start  
   b. Begin  
   c. Module name  
   d. Continue
19. Which of the following is not a benefit of using modules?

a. Simpler code  
b. Faster development  
c. Code reuse  
d. All of the above

20. The technique used by programmers to break down an algorithm into modules is called _________________.

a. Bottom-Up design  
b. Top-Down design  
c. Sequential design  
d. Random design
### Unit Exam 2 - ANSWER KEY

<table>
<thead>
<tr>
<th>Question Number</th>
<th>Answer Options</th>
<th>Correct Answer</th>
<th>Course Objective Tested</th>
<th>Reference in Course</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A B C D</td>
<td>A</td>
<td>3</td>
<td>Modules 76</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Introduction to Modules</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>A B C D</td>
<td>C</td>
<td>3</td>
<td>Lab 4.1: Task 4</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Formulating Pseudocode</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>A B C D</td>
<td>D</td>
<td>3</td>
<td>Modules 89</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Local Variables</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>A B C D</td>
<td>D</td>
<td>3</td>
<td>Modules 78</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Defining and Calling a Module</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>A B C D</td>
<td>C</td>
<td>3</td>
<td>Modules 79</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Defining and Calling a Module</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>A B C D</td>
<td>B</td>
<td>4</td>
<td>Functions 217</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Introduction to Functions: Generating Random Numbers</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>A B C D</td>
<td>A</td>
<td>3</td>
<td>Modules 88</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Local Variables</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>A B C D</td>
<td>A</td>
<td>3</td>
<td>Modules 97</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Passing Arguments by Value and by Reference</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>A B C D</td>
<td>B</td>
<td>3</td>
<td>Modules 81</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Defining and Calling a Module</td>
<td></td>
</tr>
<tr>
<td>Question Number</td>
<td>Answer Options</td>
<td>Correct Answer</td>
<td>Course Objective Tested</td>
<td>Reference in Course</td>
</tr>
<tr>
<td>-----------------</td>
<td>----------------</td>
<td>----------------</td>
<td>-------------------------</td>
<td>---------------------</td>
</tr>
<tr>
<td>10.</td>
<td>A B C D</td>
<td>B</td>
<td>3</td>
<td>Modules</td>
</tr>
<tr>
<td>11.</td>
<td>A B C D</td>
<td>C</td>
<td>3</td>
<td>Modules</td>
</tr>
<tr>
<td>12.</td>
<td>A B C D</td>
<td>B</td>
<td>3</td>
<td>Modules</td>
</tr>
<tr>
<td>13.</td>
<td>A B C D</td>
<td>C</td>
<td>3</td>
<td>Modules</td>
</tr>
<tr>
<td>15.</td>
<td>A B C D</td>
<td>D</td>
<td>3</td>
<td>Modules</td>
</tr>
<tr>
<td>16.</td>
<td>A B C D</td>
<td>A</td>
<td>3</td>
<td>Modules</td>
</tr>
<tr>
<td>17.</td>
<td>A B C D</td>
<td>C</td>
<td>3</td>
<td>Modules</td>
</tr>
<tr>
<td>18.</td>
<td>A B C D</td>
<td>C</td>
<td>3</td>
<td>Modules</td>
</tr>
<tr>
<td>19.</td>
<td>A B C D</td>
<td>D</td>
<td>3</td>
<td>Modules</td>
</tr>
<tr>
<td>Question Number</td>
<td>Answer Options</td>
<td>Correct Answer</td>
<td>Course Objective Tested</td>
<td>Reference in Course</td>
</tr>
<tr>
<td>-----------------</td>
<td>----------------</td>
<td>----------------</td>
<td>-------------------------</td>
<td>---------------------</td>
</tr>
<tr>
<td>20.</td>
<td>A B C D</td>
<td>B</td>
<td>3</td>
<td>Modules</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>82</td>
</tr>
</tbody>
</table>
Unit Exam 3

DATE: ____________________________
STUDENT NAME: ____________________________
COURSE NUMBER: ____________________________
INSTRUCTOR: ____________________________
SCHOOL LOCATION: ____________________________

General Instructions:

1. This is a closed-book, closed-notes exam. No reference material (including assignments and labs) will be permitted for use during the exam session.
2. The exam contains the following types of questions:
   - Multiple choice
3. Please use the separate answer sheet provided to you for marking your answers.
Unit Exam 3 Questions

1. Typically, in conditions that require checking if something is equivalent, the ______ operator is used instead of the assignment operator _____.
   a. =, ==
   b. ==, =
   c. !=, =
   d. <>, =

2. Which structure is a logical design that controls the order in which a set of statements executes?
   a. Control
   b. Sequence
   c. Module
   d. Terminal

3. Consider the following statement:
   A store is giving a discount of 30% for all purchases of over $100. Which of the following is not the appropriate structure to use to program the statement?
   a. Control
   b. Decision
   c. Selection
   d. Sequence

4. The flowchart symbol that indicates that some condition must be tested is _____________.
   a. a rectangle
   b. an oval
   c. a square
   d. a diamond

5. What type of operator can be used to determine whether a specific relationship exists between two values?
   a. Boolean
   b. Relational
   c. Logical
   d. Mathematical
6. A case structure is a _____ alternative decision structure.
   a. single  
   b. dual  
   c. multiple  
   d. single or dual

7. Consider the following pseudocode:

   Declare String choice
   Declare Real product

   Set choice = ‘y’
   Set product = 4.00

   If choice != “n” OR cost < 5.00 Then
      Display “Order product.”
   Else
      Display “Do not order product.”
   End If

   The expected output would be:
   a. Do not order product.  
   b. Order product.  
   c. Syntax error, should use == to compare choice. 
   d. Syntax error, should use = to compare choice.

8. A counter-controlled repetition structure is preferable when _______.
   a. it is undecided whether a loop is required or not  
   b. the number of loop iterations is not known ahead of time 
   c. the number of loop iterations is known ahead of time 
   d. None of the above

9. The _____ operator is a unary operator because it works with only one operand.
   a. AND  
   b. NOT  
   c. OR  
   d. EITHER
10. Consider the following pseudocode:

Declare Integer count

For count = 1 To 3 Step 2
    Display count
End For

What would the output be?

a. 1
   3
b. 2
   3
c. 1
   2
d. 1
   2
   3

11. Which of the following are good examples of a posttest loop?

   a. Do…While and Do…Until loops
   b. While loops
   c. For loops
   d. None of the above

12. If the expression is false, the ____ operator will return true.

   a. AND
   b. NOT
   c. OR
   d. EITHER
13. Consider the following pseudocode:

Declare Integer count
Set count = 5
While count < 5
  If count > 3 Then
    count = count – 1
  Else
    Display count
  End If
  count = count + 1
End While

What would the output be?

a.  
   1
   2
   3
   4

b.  
   1
   2
   3
   3
   3
   3
   3

 c.  
   5

d. None of the above

14. What type of operators are the following?

>  <  >=  <=  ==  !=

a. Boolean
b. Logical
c. Relational
d. Mathematical
15. Which operator would make the following expression false?

   True _____ False

   a. AND
   b. NOT
   c. OR
   d. EITHER

16. When testing multiple consecutive conditions _____.

   a. a nested decision structure must be used
   b. an if-then-else-if decision structure must be used
   c. Either a or b
   d. None of the above

17. Which structure causes a statement or set of statements to execute repeatedly?

   a. Start over
   b. Sequence
   c. Decision
   d. Repetition

18. Which pair of loops causes a statement or set of statements to repeat as long as a condition is true?

   a. While and Do-While
   b. While and Do-Until
   c. Do-While and Do-Until
   d. Do-Until and For

19. Which loop repeats a statement or set of statements as long as the Boolean expression is false?

   a. Do-While
   b. Do-Until
   c. For
   d. While
20. The statements that appear between the While and the End While clauses are called the ______.
   a. head of the loop
   b. loop
   c. body of the loop
   d. loop statements

21. Which loop statement does not contain an increment statement but automatically increments the counter at the end of each iteration?
   a. While
   b. For
   c. Do-While
   d. Do-Until

22. The following is an example of what type of loop?
   For k=7 to maxValue
   a. Count-controlled
   b. Condition-controlled
   c. While
   d. Do-While

23. The special value that marks the end of a list of values is a ______ value.
   a. sentinel
   b. stop
   c. end
   d. continue

24. How many times will the following loop iterate?
   Set k=1
   While k<=5
       Display k
   End While
   a. Two
   b. Three
   c. None
   d. Infinite
25. How many times will the following loop iterate?

Set k=1
While k>5
    Display k
End While

a. Two
b. Three
c. None
d. Infinite
# Unit Exam 3 - ANSWER KEY

<table>
<thead>
<tr>
<th>Question Number</th>
<th>Answer Options</th>
<th>Correct Answer</th>
<th>Course Objective Tested</th>
<th>Reference in Course</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>A</td>
<td>B</td>
<td>5</td>
<td>Decision Structures and Boolean Logic</td>
</tr>
<tr>
<td>1.</td>
<td>B</td>
<td>C</td>
<td>5</td>
<td>Writing a Decision Structure in Pseudocode</td>
</tr>
<tr>
<td></td>
<td>A</td>
<td>B</td>
<td>5</td>
<td>Introduction to Decision Structures</td>
</tr>
<tr>
<td>2.</td>
<td>A</td>
<td>C</td>
<td>5</td>
<td>Introduction to Decision Structures</td>
</tr>
<tr>
<td></td>
<td>D</td>
<td>D</td>
<td>5</td>
<td>Introduction to Decision Structures</td>
</tr>
<tr>
<td>3.</td>
<td>A</td>
<td>D</td>
<td>5</td>
<td>Introduction to Decision Structures</td>
</tr>
<tr>
<td>4.</td>
<td>C</td>
<td>B</td>
<td>5</td>
<td>Introduction to Decision Structures</td>
</tr>
<tr>
<td>5.</td>
<td>B</td>
<td>C</td>
<td>5</td>
<td>Introduction to Decision Structures</td>
</tr>
<tr>
<td>6.</td>
<td>A</td>
<td>C</td>
<td>5</td>
<td>Introduction to Decision Structures</td>
</tr>
<tr>
<td>7.</td>
<td>A</td>
<td>B</td>
<td>5</td>
<td>Unit 6—Decisions I</td>
</tr>
<tr>
<td>8.</td>
<td>A</td>
<td>C</td>
<td>6</td>
<td>Lab 6.1: Task 5</td>
</tr>
<tr>
<td></td>
<td>B</td>
<td>C</td>
<td>6</td>
<td>Pseudocode: Nested Decision Structures</td>
</tr>
<tr>
<td>9.</td>
<td>A</td>
<td>B</td>
<td>5</td>
<td>Repetition Structure</td>
</tr>
<tr>
<td>10.</td>
<td>A</td>
<td>B</td>
<td>5</td>
<td>Count-Controlled Loops and the For Statement</td>
</tr>
<tr>
<td></td>
<td>A</td>
<td>C</td>
<td>5</td>
<td>Logical Operators</td>
</tr>
<tr>
<td></td>
<td>A</td>
<td>A</td>
<td>6</td>
<td>For Loop and Pseudocode</td>
</tr>
<tr>
<td>Question Number</td>
<td>Answer Options</td>
<td>Correct Answer</td>
<td>Course Objective Tested</td>
<td>Reference in Course</td>
</tr>
<tr>
<td>-----------------</td>
<td>----------------</td>
<td>----------------</td>
<td>-------------------------</td>
<td>---------------------</td>
</tr>
<tr>
<td>11.</td>
<td>A B C D</td>
<td>A</td>
<td>6</td>
<td>Repetition Structure</td>
</tr>
<tr>
<td>12.</td>
<td>A B C D</td>
<td>B</td>
<td>5</td>
<td>Decision Structures and Boolean Logic</td>
</tr>
<tr>
<td>13.</td>
<td>A B C D</td>
<td>D</td>
<td>6</td>
<td>Lab 8—Repetitive Processing I</td>
</tr>
<tr>
<td>14.</td>
<td>A B C D</td>
<td>C</td>
<td>5</td>
<td>Decision Structures and Boolean Logic</td>
</tr>
<tr>
<td>15.</td>
<td>A B C D</td>
<td>A</td>
<td>5</td>
<td>Decision Structures and Boolean Logic</td>
</tr>
<tr>
<td>16.</td>
<td>A B C D</td>
<td>C</td>
<td>5</td>
<td>Decision Structures and Boolean Logic</td>
</tr>
<tr>
<td>17.</td>
<td>A B C D</td>
<td>D</td>
<td>6</td>
<td>Repetition Structure</td>
</tr>
<tr>
<td>18.</td>
<td>A B C D</td>
<td>A</td>
<td>6</td>
<td>Repetition Structure</td>
</tr>
</tbody>
</table>

© ITT Educational Services, Inc.

Date: 04/09/09
<table>
<thead>
<tr>
<th>Question Number</th>
<th>Answer Options</th>
<th>Correct Answer</th>
<th>Course Objective Tested</th>
<th>Reference in Course</th>
<th>Chapter/Unit Name</th>
<th>Page Number</th>
<th>Topic Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>19.</td>
<td>A B C D</td>
<td>B</td>
<td>6</td>
<td>Repetition Structure</td>
<td>180</td>
<td>The Do-Until Loop</td>
<td></td>
</tr>
<tr>
<td>20.</td>
<td>A B C D</td>
<td>C</td>
<td>6</td>
<td>Repetition Structure</td>
<td>165</td>
<td>Writing a While loop in Pseudocode</td>
<td></td>
</tr>
<tr>
<td>21.</td>
<td>A B C D</td>
<td>B</td>
<td>6</td>
<td>Repetition Structure</td>
<td>183</td>
<td>Deciding which loop to use</td>
<td></td>
</tr>
<tr>
<td>22.</td>
<td>A B C D</td>
<td>A</td>
<td>6</td>
<td>Repetition Structure</td>
<td>186</td>
<td>The For Statement</td>
<td></td>
</tr>
<tr>
<td>23.</td>
<td>A B C D</td>
<td>A</td>
<td>6</td>
<td>Repetition Structure</td>
<td>205</td>
<td>Sentinels</td>
<td></td>
</tr>
<tr>
<td>24.</td>
<td>A B C D</td>
<td>D</td>
<td>6</td>
<td>Unit 9—Repetitive Processing II</td>
<td>Lab 9.1: Task 1 and 2</td>
<td>For Loop and Pseudocode; For Loop and Flowcharts</td>
<td></td>
</tr>
<tr>
<td>25.</td>
<td>A B C D</td>
<td>C</td>
<td>6</td>
<td>Unit 9—Repetitive Processing II</td>
<td>Lab 9.1: Task 1 and 2</td>
<td>For Loop and Pseudocode; For Loop and Flowcharts</td>
<td></td>
</tr>
</tbody>
</table>
Appendix B: Project for This Course

Not applicable to this course
Appendix C: Lab Solutions

Solutions for the labs in this course may be downloaded from the Pearson Instructor Resource site:

http://www.pearsonhighered.com/educator/academic/product/0,3110,032147127X,00.html

Select the Resources tab, and select <View Downloadable Files> for the Lab Manual to Accompany Starting Out with Programming Logic & Design Solutions. Download both files; one is for Labs 1-8 and one is for Labs 9-10.
Appendix D

This page intentionally left blank
This page intentionally left blank
Appendix G: Homework Solutions

Unit 1

Multiple Choice:

5. C
14. B
15. C
21. C
24. A

Short Answer:

1. A CPU conducts all the “thinking,” that is, it executes all instructions.

6. A compiler creates an executable binary file while an interpreter creates binary instructions that are immediately executed by the CPU—no executable file is created. Put another way, the compiler just does program language translation while an interpreter conducts program translation and execution.

Unit 2

Multiple Choice:

1. C
3. D
4. B
8. A

Algorithm Workbench:

1. Set count = 27
2. Set total = 10 + 14
3. Set due = total - downpayment
Programming Exercise:

8.
Display “Enter Price of Meal”
Input price
Set salestax = price * 0.07
Set tip = price * 0.15
Set total = price + salestax + tip
Display “Your Sales Tax is “, salestax, “ your tip is “, tip, “ your total is “, total

Unit 3

Multiple Choice:

15. D
16. B
17. B
18. C
20. A

Algorithm Workbench:

11. 11
12. 5

Programming Exercise:

7.
Declare Real miles
Declare Real gallons
Declare Real mpg
Display “Enter miles driven”
Input miles
Display “Enter gallons used”
Input gallons
Set mpg = miles / gallons
Display “Your mpg is “, mpg

Unit 4

Multiple Choice:

2. A
3. D
Short Answer:

3. Flow of control returns to just after where the call statement for the module was made and the module’s local variables go out of scope.

4. A local variable is variable declared inside a module. Local variables can only be accessed by statements inside the module where they were declared.

**Unit 5**

Multiple Choice:

12. C
13. A
14. D
15. B

Algorithm Workbench:

2. Call showValue(12)
3. a = 3, b = 2, c = 1
6.
   Output will be:
   0 0
   1 3.4

**Unit 6**

Multiple Choice:

3. B
5. A
7. B
8. C
Algorithm Workbench:

3. If a < 10 Then
   Set b = 0
Else
   Set b = 99
End If

4. If score < 60 Then
   Display “Your grade is F.”
Else
   If score < 70 Then
      Display “Your grade is D.”
   Else
      If score < 80 Then
         Display “Your grade is C.”
      Else
         If score < 90 Then
            Display “Your grade is B.”
         Else
            Display “Your grade is A.”
         End If
      End If
   End If
End If

5. If amount1 > 10 Then
   If amount2 < 100 Then
      If amount1 > amount2 Then
         Display amount1
      Else
         Display amount2
      End If
   End If
End If

Unit 7

Multiple Choice:

10. B
12. A
13. B
14. C
15. A

Algorithm Workbench:

6. Select selection
   Case 1:
   Display “You selected A.”
   Case 2:
   Display “You selected 2.”
   Case 3:
   Display “You selected 3.”
   Case 4:
   Display “You selected 4.”
   Default:
   Display “Not good with numbers, eh?”
   End Select

7. If speed >= 24 AND speed <= 56 Then
   Display “Speed is normal”
   Else
   Display “Speed is abnormal”
   End If

8. If points >= 9 AND points <= 51 Then
   Display “Valid points.”
   Else
   Display “Invalid points.”
   End If

10. If minimum Then
    hours = 10
    End If

Unit 8

Multiple Choice:

1. B
2. D
3. D
4. A
5. B
7. C

**Algorithm Workbench:**

1. Declare Real product
   Set product = 0.0
   While product < 100.0
       Input product
       Set product = product * 10.0
   End While

2. Declare String continue
   Declare Real number1
   Declare Real number2
   Set continueYes = “y”
   Do
       Display “Enter a number”
       Input number1
       Display “Enter another number”
       Input number2
       Set sum = number1 + number2
       Display “Sum is”, sum
       Display “Do you want to continue (y = yes)?”
       Input continueYes
   While continueYes == “y” OR continueYes == “Y”

7. Declare Integer x
   Do
       Display “Enter a number.”
       Input x
   While x > 0

8. Declare String sure = “N”
   While sure != “Y” AND sure != “y”
       Display “Are you sure you want to quit?”
       Input sure
   End While
Unit 9

Multiple Choice:

6. A
9. D
10. A

Algorithm Workbench:

3.
Declare Integer counter
   For counter = 0 To 1000 Step 10
       Display counter
   End For

4.
Declare Integer counter
Declare Real value, total
Set total = 0
For counter = 1 To 10
    Display “Enter a number”
    Input value
    total = total + value
End For

6.
Declare Integer counter1, counter2
Declare String displayValue
For counter1 = 1 to 10
    displayValue = “”
For counter2 = 1 to 15
    Set displayValue = displayValue + “#”
End For
Display displayValue
End For

9.
Declare Integer count
For count = 0 To 49
    Display “The count is “, count
End For
10.

Declare Integer count = 1
While count <= 50
    Display count
    Set count = count + 1
End While
Appendix H: Lab Practicum

Lab Practicum

Purpose: As a team, you will be designing, creating, and debugging program software that implements input validation, decision making, and repetition. The software program will be designed using pseudocode and flowcharts, and then implemented in the Python programming language.

Deliverables, Requirements, and Timeline

1. Group Selection and Choosing a Project Topic:
   1. Select team members. All team members should be part of the program creation process.
   2. Choose a project topic. Either create your own or choose from one of the following topics:

   a) Programming a Robot Battlebot
      Write a program for a Robot Battlebot that satisfies these conditions:
      • The battlebot should have the following command menu selections: Fire a Weapon, Move Forward, Move Backward, Exit
      • If the Fire Weapon command is given, the user should be prompted to enter how far away (in feet) the opponent battlebot is. If the opponent is within 20 feet, a message should be displayed that the opponent is destroyed; if it is within 40 feet, the message should say it is partially disabled; and if it is over 40 feet away, the message should say it is unharmed. The battlebot has enough ammunition to fire its weapon five times in total. If it is out of ammunition, it should display a message that it cannot fire anymore when the Fire Weapon command is given.
      • If the Move Forward command is given, the user should be prompted to enter how far they want to go and then how far away (in feet) that any obstacles are in front of the battlebot. If an obstacle is blocking the battlebot’s path, then the battlebot should print a message saying it is blocked and how far it can move forward. If the battlebot is not blocked, it should display a message that it can move the entire distance and state the distance.
      • The Move Backward command should work the same way as the Move Forward command.
• The Battlebot should keep track of how far it travels. It has enough fuel to travel 200 feet in total. If it runs out of fuel, it should display a message that it is out of fuel when the Move Forward or Move Backward command is given.

b) Programming a Network Backup Routine
Write a program for a network backup that satisfies these conditions:

• The routine should have the following command menu selections: New Backup Item, Remove Backup Item, Exit

• The New Backup Item command will prompt the user for the size of the file to be backed up (in megabytes), its priority level, and the frequency per week. The routine should adjust the space needed for the item’s backup by the weekly frequency. For example, a 400MB file that is backed up 4x per week has a total weekly backup need of 400 x 4 = 1600 MB.

Total weekly limits for all backups are as follows:

<table>
<thead>
<tr>
<th>Priority Level</th>
<th>Weekly Limit</th>
</tr>
</thead>
<tbody>
<tr>
<td>High Priority</td>
<td>3000 MB</td>
</tr>
<tr>
<td>Medium Priority</td>
<td>5000 MB</td>
</tr>
<tr>
<td>Low Priority</td>
<td>9000 MB</td>
</tr>
</tbody>
</table>

• The routine should determine if there is enough space left to handle the requested backup and display a message stating either there is not enough storage space left or that the backup is being stored. If a new item is being backed up, the available storage space should be decreased by the amount of the backup.

• The Remove Backup Item command will prompt the user for the size of the item (in megabytes), its priority level, and the frequency per week. If the space being requested to be freed up is more than what has been used in storage, then a message should be displayed advising the user to recheck the input. Otherwise, the available storage should be increased by the amount of the backup and a message should be displayed showing the new available storage.

c) Programming a Personal Budget Program
Write a program for a personal budget that satisfies these conditions:

• The program should have the following command menu selections: Add New Expense, Remove Expense, Add New Income, Remove Income, Exit
• The Add New Expense command will prompt the user for the amount of the expense and the frequency of the expense per month. The total monthly expense of an item is its expense multiplied by its monthly frequency. The total initial budget for monthly expenses is $4,000. If the new expense exceeds what is left for monthly expenses, the program should display a message that the expense was rejected because the budget was exceeded. If the expense can be covered by the monthly budget, the program should display a message that the expense was accepted and state the budget left after the expense.

• The Remove Expense should prompt the user for the expense amount and its monthly frequency. If the expense amount to be removed exceeds whatever has been used of the current budget, then a message should be displayed to recheck the expense amounts. Otherwise the remaining budget will be increased by the amount of the expense reduction and a message displayed stating the amount of the currently available budget funds.

• The Add New Income option will prompt the user for the amount of the increase in monthly income and increase the monthly budget by that amount. A message will be displayed indicating the amount of the new available funds.

• The Remove Income option will prompt the user for the amount of reduction in monthly income. If the reduction exceeds the available funds, then print a message indicating the amount owing. Otherwise, set the budget to the difference and print the amount of funds available.

2. Create the Pseudocode:

1. You should consider the following accumulator variables in designing your pseudocode:

   a) **Programming a Robot Battlebot**
      
      ammunition = 5 (initially)
      fuel = 200 (initially)

   b) **Programming a Network Backup Routine**
      
      highPriority = 3000 (initially)
      medPriority = 5000 (initially)
      lowPriority = 9000 (initially)

   c) **Programming a Personal Budget Program**
      
      totalBudget = 4000 (initially)
2. Program code for the menu selections should be placed inside modules named as follows:

   a) Programming a Robot Battlebot
      fireWeapon( )
      moveForward( )
      moveBackward( )

   b) Programming a Network Backup Routine
      addBackupItem( )
      removeBackupItem( )

   c) Programming a Personal Budget Program
      addExpense( )
      removeExpense( )
      addRevenue( )
      removeRevenue( )

3. All projects should implement a While loop in their code to display the command selection menu. The While loop should use a variable called choice to capture the user’s menu selection. The While loop should test to see what value the variable choice has to determine which menu option was selected. Invalid selections should receive a message to input selection again. You can use the following pseudocode as an outline of the selection menu loop:

   a) Programming a Robot Battlebot
      //variable declarations:
      Declare Integer choice = 0
      Declare Integer ammunition = 5
      Declare Real fuel = 200.0

      //main selection menu
      While choice != 4
         //display menu
         Display “Menu Selections:”
         Display “1 –Fire Weapon”
         Display “2 –Move Forward”
         Display “3 –Move Backward”
         Display “4 –Exit”
         Display “Enter your selection”
         Input choice
         //check menu selection
         If choice == 1 Then
            fireWeapon( )
Else If choice == 2 Then
    moveForward( )
Else If choice == 3 Then
    moveBackward( )
Else If choice == 4 Then
    Display “Goodbye!”
Else
    Display “Invalid input – please try again.”
End If
End While

b) Programming a Network Backup Routine
 //variable declarations:
 Declare Integer choice = 0
 Declare Real highPriority = 3000.0
 Declare Real medPriority = 5000.0
 Declare Real lowPriority = 9000.0

//main selection menu
While choice != 3
    //display menu
    Display “Menu Selections:”
    Display “1 – Add Backup Item”
    Display “2 – Remove Backup Item”
    Display “3 – Exit”
    Display “Enter your selection”
    Input choice
    //check menu selection
    If choice == 1 Then
        addBackupItem( )
    Else If choice == 2 Then
        removeBackupItem( )
    Else If choice == 3 Then
        Display “Goodbye!”
    Else
        Display “Invalid input – please try again.”
    End If
End While

c) Programming a Personal Budget Program
 //variable declarations:
 Declare Integer choice = 0
 Declare Real totalBudget = 4000

//main selection menu
While choice != 5
//display menu
Display “Menu Selections:”
Display “1 – Add an Expense”
Display “2 – Remove an Expense”
Display “3 – Add Revenue”
Display “4 – Remove Revenue”
Display “5 – Exit”
Display “Enter your selection:”
Input choice
//check menu selection
If choice == 1 Then
    addExpense( )
Else If choice == 2 Then
    removeExpense( )
Else If choice == 3 Then
    addRevenue( )
Else If choice == 4 Then
    removeRevenue( )
Else If choice == 5 Then
    Display “Goodbye!”
Else
    Display “Invalid input – please try again.”
End If
End While

3. Create the flowchart of the pseudocode using Raptor.
4. Translate the pseudocode into Python program code.
5. Test that the program will only accept valid user input.

Deliverables: You will submit a Word document containing the following:
- Cover page with Project Name, Team Members, Instructor Name, and Course Number
- Pseudocode for the program
- Flowchart for the program from Raptor
- Python program code
- Samples of program code messages