ICS C120: INTRODUCTION TO PROGRAMMING

Item	Value
Curriculum Committee Approval Date	11/17/2023
Top Code	070700 - Computer Software Development
Units	3 Total Units
Hours	72 Total Hours (Lecture Hours 54; Lab Hours 18)
Total Outside of Class Hours	0
Course Credit Status	Credit: Degree Applicable (D)
Material Fee	No
Basic Skills	Not Basic Skills (N)
Repeatable	No
Open Entry/Open Exit	No
Grading Policy	Standard Letter (S), • Pass/No Pass (B)

Course Description

This course provides students with an introductory exploration of the fundamental concepts and practices in computer programming. Designed for students with little or no prior programming experience, this course provides a solid foundation in problem-solving, algorithmic thinking, and coding skills. Students will learn how to design, write, and debug computer programs using a structured and logical approach. Through hands-on exercises and projects, students will gain the necessary skills to become proficient programmers. ADVISORY: CIS C111. Transfer Credit: CSU.

Course Level Student Learning Outcome(s)

1. Develop a foundational understanding of programming concepts, including variables, data types, and basic control structures.
2. Apply fundamental control structures, including loops and conditional statements, to control program flow.
3. Demonstrate file input/output operations to read and write data from and to external files.

Course Objectives

- 1. Cultivate students' problem-solving skills by demonstrating how to apply programming techniques to analyze and solve real-world problems.
- 2. Introduce algorithmic thinking and the ability to design clear, step-by-step procedures to solve problems.
- 3. Define foundational programming concepts, including variables, data types, and basic control structures.
- 4. Provide students with the knowledge to gain proficiency in a programming language (e.g., Python, Java, C++) and be able to write, debug, and modify code effectively.
- 5. Describe the importance of writing clear, well-organized, and commented code for improved readability and maintainability.
- 6. Explore basic data structures such as arrays, lists, and dictionaries and demonstrate how to use them to store and manipulate data.
- 7. Reflect on the ethical and social implications of programming, including privacy, security, and responsible computing.
- 8. Demonstrate the use of fundamental control structures, including loops and conditional statements, and apply them to control program flow.
- 9. Demonstrate methods to identify and resolve common programming errors and debug code effectively.
- 10. Describe how to interact with users by receiving input and presenting output through various methods, such as console, files, or GUIs.

Lecture Content

Introducing Computer Programming What Is a Computer Program? What Do Programmers Do? The Software Development Life Cycle Client/Server Applications Getting Started Client/Server Design in Web Applications Working with Files and Folders File and Folder Addresses for various operating systems Program Design From Requirements to Algorithms What Are Instructions? Common Characteristics of Instructions Sequence, Selection and Repetition Structures A Programming Example Developing an Algorithm Basics of Markup Creating a User Interface Introducing HTML Tags Ignoring White Space Introducing HTML Tables Introducing Style Sheets Creating a Working Program Basics of PHP Working with HTML and PHP How a .php File is Processed Important Features of Client/Server Programs Receiving Input from a Form Persistence Saving and Retrieving Data The Difference Between Persistent and Transient Data Files and Databases Working with a Text File (closing, reading data from) PHP Functions to Read/Write Data from a Text File Programs that Choose Introducing Selection Structure Introducing IF and IF..ELSE Structures Introducing Flow Charts Boolean Expressions and Relational Operators Selection Using the IF Structure Testing Threshold Values Multiple Selection, Nesting, ANDs and ORs Validating User Input Validation Rules for the Wage Application Using A Nested Selection Structure to Validate Input Designing Applications with Nested Selection Structures Programs that Count Harnessing the Power of Repetition Controlling a Loop by Counting Coding a FOR Loop in PHP General Syntax of a FOR Loop Using a Variable to Control the Loop Condition While NOT End-Of-File Introducing Event-Controlled Loops Characteristics of WHILE Loops The Structure of WHILE Loops An Algorithm to Process Files of Unknown Length Using a WHILE Loop to Process a File of Scores Structured Data Working with Arrays What Is an Array? Working with Array Elements Extending an Array Arrays of Strings Associative Arrays and Web Session Using a Variable to Reference the Key of an Associative Array Using Associative Arrays as Lookups Using the array() Function to Create Associative Arrays Associative Arrays and the FOREACH Loop Program Modularity Working with Functions Using Functions Understanding Function Arguments Receiving Values from a Function Reasons to Use Pre-Defined Functions Connecting to a Database Working with MySQL What Is a Relational Database? The Relational Database Management System (RDBMS) Structured Query Language MySQL Configuring MySQL for Use with This Textbook Introduction to Object-Oriented Programming What is an Object? Creating and Using Instances of a Class Using Employee Objects in an Application Defining an Object Creating and Using Instances of an Object Class More About PHP Textbook Conventions Code Comments Location of Curly Braces HTML and PHP Files

Lab Content

Create an Input, Processing, Output (IPO) chart Design a user interface Create an HTML form Create an HTML form to obtain user input Read

and write data to an external text file Compare strings (e.g., testing a password) Program to create a bar chart (e.g., loops within loops) Program to asssociative arrays as lookups

Method(s) of Instruction

- Lecture (02)
- DE Live Online Lecture (02S)
- DE Online Lecture (02X)
- Lab (04)
- DE Live Online Lab (04S)
- DE Online Lab (04X)

Instructional Techniques

This course will utilize a combination of lecture, hands-on guided laboratory assignments, classroom/discussion student interactions, problem solving, quizzes, tests, and troubleshooting assignments to achieve the goals and objectives of this course. All instructional methods are consistent across all modalities.

Reading Assignments

Students will read from the required textbook and any additional reading resources provided.

Writing Assignments

For each module's programming demonstration, students will be tasked to write a minimum number of sentences describing the errors encountered. Student will be asked to include a screenshot in their documentation. Students will discuss the circumstances leading to any errors encountered. Speculate as to the nature of the bug (or error), and ask what can be done to remedy it.

Out-of-class Assignments

Watch video demonstrations, trying each line of code typed by the presenter. Be sure that the Integrated Development Environment (IDE) is working. While working through the demonstration code independently, students will be expected to take note of any errors experienced and post them to the discussion, where other students and the instructor will provide additional description to understand it further. At the close of the module, the source code created will be copied/pasted into a word processing document. A screenshot of the output the program made, including any errors will be provided by the student.

Demonstration of Critical Thinking

Students will be asked to read and review academic concepts in the textbook while experiencing how working code behaves under different conditions. Thus, practical programming techniques are taught through personal understanding, hands-on discovery, active learning, and discussing concepts and their results with others.

Required Writing, Problem Solving, Skills Demonstration

While trying programming techniques, when bugs and challenges are experienced, the student solves problems by writing clearly, explaining how academic concepts are applied successfully or unsuccessfully, and works iteratively to solve problematic code.

Eligible Disciplines

Computer science: Master's degree in computer science or computer engineering OR bachelor's degree in either of the above AND master's degree in mathematics, cybernetics, business administration, accounting or engineering OR bachelor's degree in engineering AND master's degree in cybernetics, engineering mathematics, or business administration OR bachelor's degree in mathematics AND master's degree in cybernetics, engineering mathematics, or business administration OR bachelor's degree in any of the above AND a master's degree in information science, computer information systems, or information systems OR the equivalent. Note: Courses in the use of computer programs for application to a particular discipline may be classified, for the minimum qualification purposes, under the discipline of the application. Master's degree required.

Textbooks Resources

1. Required O'Kane, M.. A Web-Based Introduction to Programming: Essential Algorithms, Syntax, and Control Structures Using PHP, HTML, and MariaDB/MySQL, 5th ed. Carolina Academic Press, 2021

Other Resources

1. Technology related white papers and articles are available at no charge to all students at multiple sites as recommended by the instructor. 2. Coastline Library