C S 49: FOUNDATIONS OF COMPUTER PROGRAMMING

Foothill College Course Outline of Record

Heading	Value
Effective Term:	Summer 2023
Units:	4.5
Hours:	4 lecture, 2 laboratory per week (72 total per quarter)
Advisory:	Intermediate Algebra or equivalent.
Degree & Credit Status:	Degree-Applicable Credit Course
Foothill GE:	Non-GE
Transferable:	CSU/UC
Grade Type:	Letter Grade (Request for Pass/No Pass)
Repeatability:	Not Repeatable

Student Learning Outcomes

- A successful student will be able to write and debug computer programs which make use of the fundamental control structures and method-building techniques common to all programming languages. Specifically, the student will use data types, input, output, iterative, conditional, and functional components of the language in his or her programs.
- A successful student will be able to use object-oriented programming techniques to design and implement a clear, well-structured computer program. Specifically, the student will use and design classes and objects in his or her programs.

Description

Introduction to basic computer programming concepts using an object-oriented language. Topics include the software life-cycle, procedural vs. object-oriented programming, IDE and debugging, documentation, and coding conventions. Using an object-oriented computer language, students will explore data types, basic data structures and algorithms, control structure, console and file I/O, functions, error handling and testing.

Course Objectives

The student will be able to:

1. Demonstrate an understanding of the software life-cycle, including design, development, styles, documentation, testing and maintenance
2. Effectively use program design tools and programming environments
3. Compare and contrast procedural versus objected-oriented programming
4. Use data types, variables and expressions appropriately
5. Use control structures effectively
6. Write algorithms, including simple sorting and searching
7. Incorporate console and file input/output
8. Handle run-time errors appropriately
9. Make use of predefined Application Programming Interfaces
10. Write programmer-defined functions
11. Demonstrate comfort with applications used throughout course

Course Content

1. Software life-cycle, including design, development, styles, documentation, testing and maintenance
   a. Coding conventions
      i. Naming
      ii. Indentation
   b. Documentation
   c. Test-driven and iterative development methods
   d. Principles of testing and designing test data
2. Program design tools and programming environments
   a. Navigation through the operating system file structure through well-organized storage and retrieval of files
   b. Storage and retrieval of files to/from a server or repository
   c. Writing vs. running a program
   d. Use of editor, compiler and debugger
3. Procedural versus objected-oriented programming
   a. Survey of current languages
4. Data types, variables, expressions
   a. Primitive data
   b. Numeric data
   c. Character and string data
   d. Boolean data
   e. Constants
   f. Lists and arrays, including multi-dimensional arrays
   g. Creating and evaluating numeric, character, and boolean expressions
   h. Type conversions and casting
5. Control structure
   a. Selective structures: if and switch
   b. Repetitive structures: loops
   c. Code blocks
6. Algorithms, including simple sorting and searching
7. Console and file input/output
   a. Unformatted output
   b. Formatted output
   c. User input
   d. File and Stream I/O
8. Error handling
   a. Syntax errors
   b. Run-time errors
   c. Logic errors
9. Predefined Application Programming Interface
   a. Parameters
   b. Return values
10. Programmer-defined functions
    a. Parameters
    b. Local variables
    c. Return values
    d. Passing parameters by value and by reference
11. Applications used throughout course in selected areas

a. Math
b. Physics
c. Chemistry
d. Biology
e. Astronomy
f. Business and Finance
g. Internet
h. Internet of Things

Lab Content

1. Using an IDE to write source code for a project and run it
   a. Distinguish source code from a recording of the run of a program
   b. Include both the source code and a recording of the run in an electronic file(s) for submission
   c. Identify a program's errors as originating in the compiler, the program logic, the user's runtime behavior, or the organization of the project in the IDE
2. Using iterative development to progressively refine a project's features to fit a specification
   a. Write and test a program that implements just one of a project's required features
   b. Add the implementation of a second required feature to the project and test thoroughly
   c. Complete the project by implementing and testing the remaining features one by one
   d. Perform regression testing after the implementation of each new feature
3. Using test-driven development to speed up debugging
   a. Write test code first that does not run
   b. Implement the code required to make the test code run successfully
4. Developing programs that are well designed and easy to modify
   a. Outline a project first in English in an abstract way, and make this outline the project's documentation
   b. Separate data and computation in a program
   c. Use named constants to keep numbers out of a program
   d. Choose an appropriate data type for a program's storage
   e. Use a consistent and standard indentation style in the source code
5. Writing expressions to be evaluated by the computer
   a. Correctly translate an English description of a numeric calculation into an expression that the computer can evaluate
   b. Get data from the user in whole numbers and convert so that the calculation takes place with floating point operations and results
   c. Write a complex boolean expression
   d. Use string manipulations to achieve a specified result
6. Writing a program that interacts with the user
   a. Accept character data at runtime from the user to fill a program's variables with values
   b. Accept numerical data from the user at runtime to use in calculations
7. Controlling the order in which program statements are executed
   a. Use branches
   b. Use loops
   c. Use function or method calls and returns
   d. Enclose groups of statements into blocks to achieve a desired execution sequence
8. Using functions or methods to write code without repetition
   a. Write a function or method with no parameters and no returned value
   b. Write a function or method with both parameters and a returned value
   c. Use the scope of variables to keep data as local as possible
   d. Read an API to find the information needed to effectively call a function or method documented there
9. Read from and write to a file system

Special Facilities and/or Equipment

1. Access to a computer laboratory with the appropriate compilers.
2. When taught online, the college will provide a fully functional and maintained course management system through which the instructor and students can interact. Students must have ongoing access to computers with internet capabilities.

Method(s) of Evaluation

Methods of Evaluation may include but are not limited to the following:

Exams
Quizzes
Programming projects
Discussions
Class presentation

Method(s) of Instruction

Methods of Instruction may include but are not limited to the following:

Lectures which include motivation for syntax and use of the object-oriented language, APIs, functional programming, example programs, and analysis of these programs
Online labs (for all sections, including those meeting face-to-face/on-campus), consisting of:
1. A programming assignment webpage located on a college-hosted course management system or other department-approved internet environment. Here, the students will review the specification of each programming assignment and submit their completed lab work
2. A discussion webpage located on a college-hosted course management system or other department-approved internet environment. Here, students can request assistance from the instructor and interact publicly with other class members
Detailed review of programming assignments, which includes model solutions and specific comments on the student submissions
In-person or online discussion which engages students and instructor in an ongoing dialog pertaining to all aspects of designing, implementing and analyzing programs
When course is taught fully online:
1. Instructor-authored lecture materials, handouts, syllabus, assignments, tests, and other relevant course material will be delivered through a college-hosted course management system or other department-approved internet environment
2. Additional instructional guidelines for this course are listed in the attached addendum of CS department online practices

Representative Text(s) and Other Materials

Downey and Mayfield. Think Java: How to Think Like a Computer Scientist, 2nd ed.. 2019.

Horstmann and Necaise. Python for Everyon, 3rd ed.. 2019.

Sebesta, Robert. Concepts of Programming Languages, 11th ed.. 2019.

Types and/or Examples of Required Reading, Writing, and Outside of Class Assignments

1. Reading Assignments:
   a. Textbook assigned reading averaging 20 pages per week
   b. Reading the supplied handouts and modules averaging 10 pages per week
   c. Reading online resources as directed by instructor though links pertinent to programming
   d. Reading library and reference material directed by instructor through course handouts
2. Writing Assignments:
   a. Writing technical prose documentation that supports and describes the programs that are submitted for grades

Discipline(s)

Computer Science