COURSE OUTLINE: CSD110 - INTRO TO PROGRAMMING

Prepared: Rodney Martin
Approved: Martha Irwin, Dean, Business and Information Technology

Course Code: Title	CSD110: INTRODUCTION TO PROGRAMMING
Program Number: Name	2095: COMPUTER PROGRAMMING
Department:	COMPUTER STUDIES
Academic Year:	2024-2025
Course Description:	The ability to solve arbitrary problems using a computer programming language is a valuable skill for anyone. Accessible to all regardless of previous experience, the goal of this course is to give students a sense of how to solve computing problems using the fundamental constructs in all programming languages: values, types, operators, variables, lists, conditionals, loops, functions, input & output. Students gain an understanding of how to break problems into sub problems that can be solved using these fundamental constructs, and they learn how computers can \`understand\` and execute the instructions they write in their programs. Due to their low barrier to entry and wide adoption, Python and/or JavaScript will be used as the programming language of delivery.
Total Credits:	4
Hours/Week:	4
Total Hours:	56
Prerequisites:	There are no pre-requisites for this course.
Corequisites:	There are no co-requisites for this course.
This course is a pre-requisite for:	CSD121, CSD123
Vocational Learning Outcomes (VLO's) addressed in this course: **Please refer to program web page for a complete listing of program outcomes where applicable.**	**2095 - COMPUTER PROGRAMMING** VLO 2 Contribute to the diagnostics, troubleshooting, documenting and monitoring of technical problems using appropriate methodologies and tools. VLO 9 Support the analysis and definition of software system specifications based on functional and non-functional requirements. VLO 10 Contribute to the development, documentation, implementation, maintenance and testing of software systems by using industry standard software development methodologies based on defined specifications and existing technologies/frameworks. VLO 11 Apply one or more programming paradigms such as, object-oriented, structured or functional programming, and design principles, as well as documented requirements, to the software development process.
Essential Employability Skills (EES) addressed in this course:	EES 3 Execute mathematical operations accurately. EES 4 Apply a systematic approach to solve problems. EES 5 Use a variety of thinking skills to anticipate and solve problems.

EES 9 Interact with others in groups or teams that contribute to effective working relationships and the achievement of goals.

Course Evaluation:

Passing Grade: 50%, D

A minimum program GPA of 2.0 or higher where program specific standards exist is required for graduation.

Other Course Evaluation & Assessment Requirements:

Students are expected to be present to write all tests in class, unless otherwise specified. If a student is unable to write a test due to illness or a legitimate emergency, that student must contact the professor prior to class and provide reasoning. Should the student fail to contact the professor, the student shall receive a grade of zero on the test.

If a student is not present 10 minutes after the test begins, the student will be considered absent and will not be given the privilege of writing the test.
Students exhibiting academic dishonesty during a test will receive an automatic zero. Please refer to the College Academic Dishonesty Policy for further information.

In order to qualify to write a missed test, the student shall have:
a.) attended at least 75% of the classes to-date.
b.) provide the professor an acceptable explanation for his/her absence.
c.) be granted permission by the professor.

NOTE: The missed test that has met the above criteria will be an end-of-semester test.

Labs / assignments are due on the due date indicated by the professor. Notice by the professor will be written on the labs / assignments and verbally announced in advance, during class.

Labs and assignments that are deemed late will have a 10% reduction per academic day to a maximum of 5 academic days at 50% (excluding weekends and holidays). Example: 1 day late - 10% reduction, 2 days late, 20%, up to 50%. After 5 academic days, no late assignments and labs will be accepted. If you are going to miss a lab / assignment deadline due to circumstances beyond your control and seek an extension of time beyond the due date, you must contact your professor in advance of the deadline with a legitimate reason that is acceptable.

It is the responsibility of the student who has missed a class to contact the professor immediately to obtain the lab / assignment. Students are responsible for doing their own work. Labs / assignments that are handed in and are deemed identical or near identical in content may constitute academic dishonesty and result in a zero grade.

Students are expected to be present to write in-classroom quizzes. There are no make-up options for missed in-class quizzes.

Students have the right to learn in an environment that is distraction-free, therefore, everyone is expected to arrive on-time in class. Should lectures become distracted due to students walking in late, the professor may deny entry until the 1st break period, which can be up to 50 minutes after class starts or until that component of the lecture is complete.

Grade
Definition Grade Point Equivalent
A+ 90 - 100% 4.00
A 80 - 89%
B 70 - 79% 3.00
C 60 - 69% 2.00

D 50 - 59% 1.00
F (Fail) 49% and below 0.00

CR (Credit) Credit for diploma requirements has been awarded.
S Satisfactory achievement in field /clinical placement or non-graded subject area.
U Unsatisfactory achievement in field/clinical placement or non-graded subject area.
X A temporary grade limited to situations with extenuating circumstances giving a student additional time to complete the requirements for a course.
NR Grade not reported to Registrar`s office.
W Student has withdrawn from the course without academic penalty.

Books and Required Resources:

Think Python by Allen B. Downey
Publisher: O`Reilly Edition: 3
ISBN: 9781098155438
Freely available online: https://allendowney.github.io/ThinkPython/

Object Oriented Programming in Python
Publisher: readthedocs.org
Freely available online: https://python-textbok.readthedocs.io/en/stable/

Course Outcomes and Learning Objectives:

Course Outcome 1	**Learning Objectives for Course Outcome 1**
1. Describe the nature of computers and programming	1.1 Define computation, and explain how it relates computers and programming languages 1.2 Explain what a programming language is (and is not) 1.3 Distinguish between compiled and interpreted languages 1.4 Explain what is meant by a language`s syntax 1.5 Describe what happens in a computer when you run a program 1.6 Describe the basic elements of all computer programs 1.7 Use a REPL to execute instructions and experiment with programming ideas 1.8 Use a text editor and interpreter to create programs
Course Outcome 2	**Learning Objectives for Course Outcome 2**
2. Create variables and simple expressions and statements	2.1 Define the terms `value` and `type` 2.2 Determine the type of a value, and cast values from one type to another 2.3 Create values of various types, including integers, floating point numbers, and strings 2.4 Explain the use of null values 2.5 Use values, operators, and operands to create expressions 2.6 Explain operator precedence 2.7 Create useful code comments 2.8 Assign values to variables, and describe how this looks at the level of computer memory 2.9 Describe variable naming conventions 2.10 Distinguish between expressions and statements 2.11 Discuss the benefits of immutable data
Course Outcome 3	**Learning Objectives for Course Outcome 3**
3. Use and create functions	3.1 Describe the nature of functions

	3.2 Identify when to encapsulate instructions into a function 3.3 Create functions that may include parameters and/or yield values 3.4 Call functions using arguments 3.5 Use function results as values in expressions that may include function composition 3.6 Analyze flow of program execution when functions are involved 3.7 Discuss variable scope 3.8 Create recursive functions, and understand when they are useful 3.9 Define and use higher-order functions 3.10 Provide function documentation using conventional syntax 3.11 Explain function preconditions and postconditions 3.12 Discuss the benefits of pure vs impure functions
Course Outcome 4	**Learning Objectives for Course Outcome 4**
4. Control program flow using conditionals	4.1 Create boolean expressions using relational and logical operators 4.2 Explain when and how non-boolean values may be interpreted as boolean values 4.3 Use conditional statements to control program flow, including chained and nested conditionals 4.4 Use conditional statements to check function preconditions 4.5 Describe the limitations of equality comparisons with floating point numbers
Course Outcome 5	**Learning Objectives for Course Outcome 5**
5. Control program flow using loops	5.1 Use loops to repeat a set of instructions a fixed number of times 5.2 Use loops to repeat instructions depending on a dynamic condition 5.3 Describe and use counter variables and sentinel values 5.4 Explain infinite loops and understand how to avoid them 5.5 Use return, break and continue statements to end loops early
Course Outcome 6	**Learning Objectives for Course Outcome 6**
6. Use objects	6.1 Explain what an `object` is 6.2 Describe object methods and properties 6.3 Use the methods and properties of objects in working code 6.4 Distinguish between mutable and immutable types 6.5 Explain what a reference is, and describe the underlying model in terms of computer memory 6.6 Discuss the difference between object equality and object identity 6.7 Distinguish between aliasing assignment and object cloning
Course Outcome 7	**Learning Objectives for Course Outcome 7**
7. Use sequences (lists, tuples, strings) to store and track information	7.1 Describe what a data structure is 7.2 Explain what a sequence is, and how it pertains to lists, tuples, and strings

	7.3 Obtain one element of a sequence using indexing 7.4 Obtain subsections of a sequence using slicing 7.5 Determine if an element is in a sequence 7.6 Remove an element from a sequence 7.7 Traverse sequences using loops 7.8 Create strings to represent textual data 7.9 Process list/tuple/string objects using common methods 7.10 Create and use nested lists and/or tuples and understand when they are useful 7.11 Use destructuring assignment to obtain sequence element values
Course Outcome 8	**Learning Objectives for Course Outcome 8**
8. Use dictionaries to store and track information	8.1 Explain how dictionaries differ from sequences such as lists 8.2 Map a key to a value using a dictionary 8.3 Obtain a dictionary item using indexing 8.4 Determine if a dictionary key has already been set 8.5 Traverse dictionary data using loops 8.6 Process dictionary objects using common methods 8.7 Create and use dictionaries of lists/tuples 8.8 Implement common coding techniques using dictionaries
Course Outcome 9	**Learning Objectives for Course Outcome 9**
9. Employ basic software design techniques	9.1 Describe the purpose of modules 9.2 Reuse existing code by importing modules 9.3 Describe the terms \`encapsulation\` and \`generalization\`, and how they pertain to functions and modules 9.4 Explain what is meant by a function or module\`s interface 9.5 Use refactoring to improve existing code 9.6 Explain what an algorithm is and be able to implement simple algorithms
Course Outcome 10	**Learning Objectives for Course Outcome 10**
10. Handle input/output, and errors	10.1 Produce output using a print statement 10.2 Employ string formatting techniques 10.3 Obtain keyboard input from a user 10.4 Distinguish between absolute and relative file paths 10.5 Create paths to specific files 10.6 Read data from a file 10.7 Write data to a file 10.8 Write simple command-line scripts that accept arguments 10.9 Distinguish between syntax, runtime, and semantic errors 10.10 Throw appropriate errors in exceptional situations 10.11 Prevent program crashes due to errors using try..catch blocks 10.12 Use debugging tools to investigate errors

Evaluation Process and Grading System:

Evaluation Type	Evaluation Weight
Coding Assignments	40%
Oral Assessment	20%

Quizzes	10%
Test 1	15%
Test 2	15%

Date:	June 16, 2024
Addendum:	Please refer to the course outline addendum on the Learning Management System for further information.