STRANDS AND STANDARDS
COMPUTER PROGRAMMING 1

Learning that works for Utah
CTE®

Course Description

An introductory course in program engineering and applications. The course introduces students to the fundamentals of computer programming. Students will learn to design, code, and test their own programs while applying mathematical concepts. Teachers introduce basic coding concepts and problem-solving skills.

Intended Grade Level	9-12
Units of Credit	0.5
Core Code	35.02.00.00.030
Concurrent Enrollment Core Code	35.02.00.13.030
Prerequisite	Digital Literacy, Computer Science Principles or Teacher Approval
Skill Certification Test Number	820
Test Weight	0.5
License Area of Concentration	CTE and/or Secondary Education 6-12
Required Endorsement(s)	
Endorsement 1	Intro to Computer Science
Endorsement 2	Programming & Software Development
Endorsement 3	Web Development
Endorsement 4	Information Technology Systems

ADA Compliant: April 2022

STRAND 1

Students will be familiar with and use a programming language IDE (Integrated Development Environment).

Standard 1

Demonstrate concept knowledge of different languages.

- Describe the difference between an interpreted language vs a compiled language
- Identify characteristics of high-level and low-level languages

Standard 2

Demonstrate the ability to use an IDE.

- Use an IDE to develop, compile, and run programs
- Understand the difference between syntax, run-time, and logic errors
- Use the debugger to identify errors

Performance Skills

Use an IDE to create a solution to solve a problem.

STRAND 2

Students will understand program development methodology and best practices.

Standard 1

Demonstrate the ability to use good programming style.

- Demonstrate proper use of white space (between lines and indentation)
- Use appropriate naming conventions for identifiers (variables, methods, functions, and file names)
- Understand the appropriate use of constants versus variables in programming style
- Construct identifiers with meaningful format; camelCase and underscore
- Implement appropriate output formatting (decimal places, dollar signs, and correct placement of variable data in a sentence)

Standard 2

Understand the ordered software development life cycle.

- Requirements Analysis: Identify specifications and understand requirements to create a solution to a problem
- Planning/Design: Design an algorithm to solve the problem using appropriate documentation (UML diagrams and pseudocode).
  - Define an algorithm
  - Break the problem down into its subcomponents using top-down design
- Implementation: Write the code, with comments, to implement the algorithm
- Testing: Test program for verification of errors and proper functionality
- Release and Maintenance: Release the solution and provide updates when necessary

Performance Skills

Demonstrate knowledge of program development methodology through a project.

STRAND 3

Students will understand and implement key programming concepts.

Standard 1

Understand and implement input and output commands.

- Understand the difference between input and output
- Understand there are different types of input (file, keyboard, mouse, microphone)
- Understand there are different types of output (speakers, monitor, printer, file)
- Write a program that receives input from a keyboard and produces output to the display

Standard 2

Understand and implement data types and variables.

- Differentiate between primitive data types (boolean, integer, float and string)
- Identify proper use of primitive data types (when to use one versus another)
- Declare a variable and assign it a value using the assignment operator
- Understand the difference between declaring and initializing a variable

Standard 3

Understand and implement operators and operands.

- Use basic arithmetic operators (modulus, multiplication, integer division, float division, addition, subtraction)
- Use basic comparison operators (<, >, ==, >=, <=)
- Use basic assignment operator (=)
- Understand order of operations for all operators
  - Parenthesis
  - Exponent
  - Multiplication
  - Division
  - Modulus
  - Addition
  - Subtraction
- Use basic logical operators (AND, OR, NOT)
- Use operands in conjunction with arithmetic, relational, and logical operators

Standard 4

Understand and implement expressions in a program.

- Understand how operators and operands are used to form expressions
- Identify and implement syntactically correct expressions
  - Possible examples: A OR B, 5==6, x != 3.142, x = 4, y + 7

Standard 5

Understand and implement functions.

- Understand and properly define scope, local variable, and global variable
- Understand what functions are and what are they used for (readability, reusability, modularity, abstraction)
- Understand the difference between a built-in function and user defined function
- Utilize built-in functions
- Understand that functions may or may not require arguments (input(s))
- Understand that functions may or may not return value(s) (output(s))

Standard 6
Understand and implement complex data types.

- Understand the difference between a simple and complex data types
- Declare a string variable in a program

Performance Skills
Write one or more programs that demonstrate effective use of the key programming concepts defined in Strand 3.

STRAND 4
Understand and implement control structures.

Standard 1
Understand and implement IF statements in a program.

- IF
    - Understand when to use an IF statement
    - Demonstrate correct use of an IF statement
- ELSE-IF
    - Understand when to use an ELSE-IF statement
    - Demonstrate correct use of ELSE-IF statements
- ELSE
    - Understand when to use an ELSE statement
    - Demonstrate proper use of an ELSE statement
- Nesting IF statements
    - Understand when to use a nested IF statement
    - Demonstrate proper use of a nested IF statement

Standard 2
Understand and implement basic loop structures in programs.

- For-loops
    - Understand when to use a for-loop
    - Understand the three components of a for-loop
        - An initial value (i = 0)
        - A condition (i < 7)
        - An update expression (i = i + 1)
    - Demonstrate proper use of for-loops
- While-loops
    - Understand when to use a while-loop
    - Demonstrate proper use of a while-loop
- Nested loops
    - Understand when to use nested loops
    - Demonstrate proper use of nested loops
- Identify the various ways that loops can end (break, met condition, condition fail)
- Design loops so they iterate the correct number of times
- Understand what causes an infinite loop

Standard 3
Understand and implement expressions and complex conditions in control structures.
- Create expressions using relational operators
  - Example: (a > 6, x != 7, y > 4)
- Form complex conditions using logical operators
  - Example: (a > 6 AND x != 7 OR y > 4)
- Incorporate complex conditions in loop structures
  - Example: While a player's health is greater than 50 and player is not dead

Performance Skills
Write one or more programs that demonstrate effective use of control structures.

STRAND 5
Students will be aware of career opportunities in the Computer Programming/Software Engineering industry and ethical applications.

Standard 1
Investigate career opportunities, trends, and requirements related to computer programming/software engineering careers.
- Identify the members of a computer programming/software engineering team:
  - team leader
  - analyst
  - senior developer
  - junior developer
  - client/subject matter expert
- Describe work performed by each member of the computer programming/software engineering team
- Investigate trends and traits associated with computer programming/software engineering careers (creativity, technical, leadership, collaborative, problem solving, design, etc.)
- Discuss related career pathways

Standard 2
Understand current ethical issues dealing with computer programming and information in society.
- Explain the impact software can have on society (i.e., privacy, piracy, copyright laws, ease of use, etc.)
- Explain the ethical reasons for creating reliable and robust software
- Describe how computer-controlled automation affects a workplace and society

Performance Skills
Develop awareness of career opportunities in the computer programming/software engineering industry ethical applications.

Workplace Skills
Workplace Skills taught:
- Communication
- Problem Solving
- Teamwork
- Critical Thinking
- Dependability
- Accountability
- Legal requirements / expectations

Skill Certification Test Points by Strand

Test Name	Test #	Number of Test Points by Strand										Total Points	Total Questions
		1	2	3	4	5	6	7	8	9	10		
Computer Programming	820	4	7	16	14	3						44	40

Computer Programming 1 – VOCABULARY

Strand 1 - Students will be familiar with and use a programming language IDE (Integrated Development Environment).	
IDE (Integrated Development Environment	Software for building applications that combines common developer tools in a single interface.
Interpreted Language	Source code is read and executed by an interpreter
Compiled Language	Source code is translated into machine code, and the machine code is stored in a separate file.
High-Level Language	Programming Language that enables a programmer to write programs that are closer to human language.
Low-Level Language	Programming language that contains basic instructions recognized by a computer.
Syntax Error	Error which is detected and prevents the program from running.
Run-Time Error	Error in the code that occurs while the program is running.
Logic Error	Mistake in the code that produces incorrect results.
Debugging	Finding and fixing problems in an algorithm or program.
Software Development Life Cycle	A process that produces software with the highest quality and lowest cost in the shortest time possible
Strand 2 - Students will understand program development methodology and best practices.	
White Space	Blank lines and extra spacing to improve readability of code.
Identifiers	Names given to variables, constants, methods, and functions.
Variable	A named value within a program.
Function	A named group of programming instructions.
Constant	Data values that stay the same every time a program is executed.
Camel Case	Naming convention where the first letter of name is lowercase, and each new word is capitalized. (camelCase)

Pascal Case	Naming convention where the first letter of each compound word is capitalized. (PascalCase)
Snake Case	Naming convention where spaces are replaced with underscores. (snake_case)
Software Development Life Cycle	1. Requirements Analysis - Identify specifications and understand requirements to create a solution. 2. Planning/Design - Design an algorithm to solve the problem using appropriate documentation (UML diagrams and pseudocode). 3. Implementation - Write the code 4. Testing - Test program for verification of errors and proper functionality. 5. Release & Maintenance - Release the solution and provide updates when necessary.
Algorithm	A finite set of instructions that accomplish a task.
Strand 3 - Students will understand and implement key programming concepts.	
Scope	Determines the accessibility (visibility) of variables.
Local Variable	Only recognized inside the function in which it is declared.
Global Variable	Recognized from anywhere inside a program.
Input	The information computers get from users, devices, or other computers.
Output	The information computers give to users, devices, or other computers.
String	An ordered sequence of characters.
Integer	A data type that is used for a whole number
Boolean	A data type that is either true or false.
Float	A data type that is used for fractional values in decimal format.
Declaration	Stating the name and data type of a variable.
Initialization	Assignment of an initial value for a variable.
Arithmetic Operators	Includes addition, subtraction, multiplication, division, and modulus operators.
Comparison Operators	<, >, ≤, ≥, ==, ≠ indicate a Boolean expression.

Order of Operations	Parenthesis, exponents, multiplication, modulus, division, addition, subtraction (PEMMDAS).
Logical Operators	NOT, AND, and OR, which evaluate to a Boolean value.
Expression	A combination of operators, constants, and variables.
Integer Division	Division in which the fractional part (remainder) is discarded.
Float Division	Division in which the fractional part (remainder) is included with a certain number of values after the decimal.
Function	A named group of programming instructions
Readability	The ease with which the code is read and understood.
Reusability	Capability of being used again or repeatedly.
Modularity	Enables reusability and minimizes duplication.
Abstraction	Hiding unnecessary details from the user.
Built-In Function	Any function that is provided as part of a high-level language and can be executed by a simple reference with specification of arguments.
User-Defined Function	A function created by the user.
Arguments	The variables given to the function for execution.
Parameters	The names listed in the method/function's definition.
Return	A value that is sent back to the user by a method/function.
Void Return	Indicates that the function does not return a value.
Simple Data Types	char, string, integer, float, double, boolean.
Complex Data Types	enumeration, array, list, object.
Strand 4 - Understand and implement control structures	
Conditional Statement	Decision making based on a Boolean value.
Nested IF Statement	An if statement placed inside another if statement.
For Loop	Initial Value Condition

	Increment/Decrement
While Loop	Loops through a block of code as long as a specified condition is true.
Nested Loop	A loop placed inside another loop.
Break	Statement used to immediately terminate a loop.
Met Condition	Expression evaluates to true.
Failed Condition	Expression evaluates to false.
Iterate	Each cycle through a loop.
Infinite Loop	A loop that, due to a logic error, will continue endlessly.
Complex Condition	Formed by combining multiple conditions with logical operators.
Exit Condition	Used to exit a loop.
Strand 5 - Students will be aware of career opportunities in Computer Programming/Software Engineering industry and ethical applications.	
Computer Programming/Software Engineering Team	Team Leader Analyst Senior Developer Junior Developer Client/Subject-Matter Expert

Computer Programming 1 – Skills Reference Sheet

Assignment, Display, and Input	
`a = expression`	Evaluates `expression` and then assigns a copy of the result to the variable **a**.
`DISPLAY(expression)`	Displays the value of `(expression)` in the console window.
`INPUT( )`	Accepts a value from the user and returns the input value.
Arithmetic Operators and Numeric Procedures	

`a + b`
`a - b`
`a * b`
`a / b`
	The arithmetic operators `+`, `-`, `*`, and `/` are used to perform arithmetic on `a` and `b`.
	
	For example, `17 / 5` evaluates to `3.4`.
	
	The order of operations used in mathematics applies when evaluating expressions.

`a MODULUS b`
`  -or-`
`a MOD b`
	Evaluates to the remainder when `a` is divided by `b`.
	
	For example, `17 MOD 5` evaluates to `2`.
	
	`MODULUS (MOD)` has the same precedence as the `*` and `/` operators.

Relational and Boolean Operators	
`NOT condition`	Evaluates to `true` if `condition` is `false`; otherwise evaluates to `false`.
`condition1 AND condition2`	Evaluates to `true` if both `condition1` and `condition2` are `true`; otherwise evaluates to `false`.
`condition1 OR condition2`	Evaluates to `true` if `condition1` is `true` or if `condition2` is `true` or if both `condition1` and `condition2` are `true`; otherwise evaluates to `false`.

`FOR(condition)`
`{`
`    <block of statements>`
`}`
	The code in `<block of statements>` is executed a certain number of times.

`WHILE(condition)`
`{`
`    <block of`
`statements>`
`}`
	The code in `<block of statements>` is repeated until the `(condition)` evaluates to `false`.

`IF(condition1)`
`{`
`    <first block of`
`statements>`
`{`
`ELSE IF(condition2)`
`{`
`    <second block of`
`statements>`
`}`
`ELSE`
`{`
`    <third block of`
`statements>`
`}`
	If `(condition1)` evaluates to `true`, the code in `<first block of statements>` is executed; if `(condition1)` evaluates to `false`, then `(condition2)` is tested; if `(condition2)` evaluates to `true`, the code in `<second block of statements>` is executed; if both `(condition1)` and `(condition2)` evaluate to `false`, then the code in `<third block of statements>` is executed.

Procedures and Procedure Calls	

`PROCEDURE procName( )`
`{`
`    <block of`
`statements>`
`}`
	Defines `procName` as a procedure that takes no arguments. The procedure contains `<block of statements>`.
	
	The procedure `procName` can be called using the following notation:
	
	`procName( )`