Coding II

Level: 3
Career Cluster: Digital Technology
Program: Information Technology
Pathway(s): Coding

Course Catalog Description

Develops advanced programming skills through increasingly complex projects focused on problem analysis, algorithm design, and software implementation. Students strengthen logical thinking, debugging, modular design, and collaborative development practices while working with high-level programming concepts and real-world coding challenges.

Summary

Challenges students to develop advanced skills in problem analysis, algorithm design, and software implementation through programming projects of increasing complexity. Students apply computational thinking, data structures, control structures, functions, modular design, debugging, and collaborative development practices while working individually and in teams. Through short- and long-term coding projects, students strengthen logical reasoning, technical communication, and professional problem-solving skills while preparing for more advanced study and workplace-connected opportunities in software development. The course also emphasizes persistence, adaptability, and project-based learning as students refine solutions and build confidence with more sophisticated programming tasks. Students are encouraged to approach programming as both a technical discipline and a creative process for designing effective, user-centered solutions.

Notes

This revised document expands the course from a shorter list of broad programming competencies into a more detailed and intentionally sequenced set of student learning outcomes with clearly defined objectives for each section. The updated version strengthens the progression of skills by adding more explicit emphasis on computational thinking, modular design, software development workflow, collaboration, leadership, work-based learning, and a final assessment project. It also provides a clearer structure for instruction by breaking content into focused sections that more directly support planning, assessment, and alignment to current CTE expectations. At the same time, the revised course preserves the core programming content from the earlier version, including algorithms, data types, variables, control structures, functions, arrays, recursion, debugging, and software quality, while presenting those topics in a more comprehensive and measurable format.

Student Learning Outcomes

1. Career Exploration and Postsecondary Opportunities

Objective: *Students will analyze career pathways in information technology, exploring industry roles, workplace expectations, required education, certifications, professional organizations, and postsecondary opportunities available in Colorado in order to make informed academic and career decisions.*

1) Identify employment opportunities in various fields with a focus in the area of information technology
    a. Recognize the work typically performed, tools and technology used, and nature of work environments
    b. Identify potential certifications within the careers
    c. Find membership organizations associated with the careers
    d. Understand the necessary education associated within the careers
2) Define professionalism within the context of information technology
    a. Criticism and evaluation
    b. Presentation
    c. Assessment
3) Recognize laws, regulations, and ethics significant to the fields of information technology
4) Identify postsecondary opportunities offered in the state of Colorado
    a. Technical college, community college, 4-year institutions
    b. Certificates, associates, bachelor, and advanced degree opportunities

2. Advanced Computational Thinking and Algorithm Design

Objective: *Students will analyze complex programming problems and design algorithms that support efficient, organized software solutions.*

5) Analyze programming problems by identifying inputs, outputs, constraints, and expected results
6) Decompose complex tasks into smaller procedures or modules
7) Design algorithms that use sequence, selection, and iteration to solve multistep problems
8) Construct pseudocode, flowcharts, or logic maps before writing code
9) Evaluate multiple solution strategies and select the most effective approach for a given problem
10) Justify algorithm design choices using appropriate technical language
11) Revise algorithm plans based on testing results or feedback

3. Data Types, Variables, and Operators

Objective: *Students will apply a range of data types, operators, and variable structures to support more advanced computational tasks.*

12) Differentiate among integer, float, string, Boolean, list, array, and other relevant data types

13) Select appropriate data types based on program requirements
14) Apply arithmetic, relational, and logical operators in programming tasks
15) Analyze how operators behave with different data types
16) Explain how data is represented and stored in a computer system
17) Create meaningful variables and constants that follow accepted naming conventions
18) Trace values through a program to verify correct data handling

4. Control Structures and Program Flow

Objective: *Students will construct programs that use decision-making and repetition to manage program flow and solve increasingly sophisticated problems.*

19) Write programs that use conditionals to support multiple decision paths
20) Develop programs that use loops to automate repeated processes
21) Explain the flow of execution in a program
22) Analyze how nested or combined control structures affect outcomes
23) Select appropriate control structures based on the needs of a problem
24) Test control logic using valid, invalid, and edge-case inputs
25) Revise program flow to improve clarity, logic, and efficiency

5. Functions, Methods, and Modular Programming

Objective: *Students will design reusable, organized code by applying functions, methods, parameters, and modular structure.*

26) Define functions or methods that perform specific tasks within a program
27) Use system-defined and programmer-defined functions in code
28) Apply value and reference parameters when supported by the selected language
29) Explain the benefits of modular programming for readability and reuse
30) Organize programs into logical sections or reusable components
31) Evaluate when a task should be solved with a separate function or method
32) Refactor programs to improve modularity and maintainability

6. Scope, Lifetime, and Code Organization

Objective: *Students will examine how scope, lifetime, and structure influence program behavior, readability, and reliability.*

33) Explain how scope and lifetime rules affect variables in code
34) Differentiate between local and global data as supported by the programming language
35) Analyze how poor variable management can lead to logic errors
36) Organize code according to accepted style and documentation practices
37) Apply comments and documentation to clarify program behavior
38) Evaluate code organization for readability and maintainability
39) Revise existing code to better reflect coding standards

7. Arrays, Collections, Structures, and Object-Oriented Design

Objective: *Students will organize and manipulate collections of data using arrays, structures, classes, and object-oriented concepts when appropriate.*

40) Write programs that use arrays or other collection types to manage groups of data
41) Manipulate stored values using indexing, iteration, and update operations
42) Group related data using a structure, class, or equivalent feature
43) Explain the purpose of object-oriented programming in software development
44) Create basic classes or structured data models when appropriate to the language
45) Differentiate between object-oriented and procedural approaches
46) Apply object-oriented or structured design to solve a programming problem

9. Recursion, References, and Intermediate Programming Techniques

Objective: *Students will apply intermediate coding techniques to solve problems that require more advanced logic and program structure.*

47) Demonstrate recursion in a program where appropriate
48) Compare recursive and iterative approaches to solving a problem
49) Use references or pointers when supported by the programming language and environment
50) Explain how references affect data handling and memory use
51) Analyze when intermediate techniques improve a solution and when they do not
52) Test recursive or reference-based code for accuracy and reliability
53) Evaluate the tradeoffs of different implementation strategies

9. Testing, Debugging, and Software Quality

Objective: *Students will improve software reliability by testing, debugging, documenting, and refining code through a structured quality process.*

54) Test programs using valid, invalid, and edge-case inputs
55) Identify syntax, logic, and runtime errors in code
56) Debug programs systematically using error messages, tracing, and test evidence
57) Document bugs, revisions, and testing outcomes
58) Evaluate whether a program meets requirements and intended outcomes
59) Revise code to improve correctness, efficiency, and readability
60) Assess how software quality affects user experience and system reliability

10. Software Development Workflow and Collaborative Coding

Objective: *Students will apply a structured software development process while working individually and with others on programming tasks of increasing complexity.*

61) Describe the stages of the program development process
62) Plan software tasks using criteria, constraints, and timelines
63) Implement programs using analysis, design, coding, testing, and documentation practices
64) Collaborate with peers to develop, test, and improve code

65) Communicate programming decisions, challenges, and solutions clearly
66) Incorporate peer or instructor feedback into program revisions
67) Evaluate team workflows and project progress against goals

11. Work-Based Learning

Objective: *Students will connect advanced coding skills to real-world software tasks through workplace-connected learning experiences and applied development scenarios.*

68) Describe how intermediate and advanced coding skills are used in software and IT environments
69) Participate in project tasks that simulate client, user, or workplace expectations
70) Apply programming skills to complete authentic or workplace-connected development tasks
71) Follow project requirements, deadlines, and technical specifications
72) Document work completed, tools used, and revisions made during development
73) Explain how classroom coding skills transfer to real software development contexts
74) Reflect on performance and growth during workplace-connected experiences

12. Leadership in Software Development

Objective: *Students will demonstrate leadership, collaboration, and professional responsibility while contributing to software development projects.*

75) Demonstrate effective communication in team-based programming environments
76) Apply collaboration strategies to solve technical problems with peers
77) Take responsibility for assigned tasks and project deadlines
78) Support team decision-making through constructive feedback and problem solving
79) Manage time and resources effectively during individual and group work
80) Reflect on leadership strengths and areas for growth in project settings
81) Contribute to a positive and ethical development environment

13. Final Assessment Project

Objective: *Students will design, develop, test, document, and present a software project that demonstrates advanced coding skills and sound software development practices.*

82) Plan a final programming project using a clearly defined problem statement, criteria, and constraints
83) Design a solution using algorithms, modular structure, and appropriate data handling
84) Develop a working program that demonstrates intermediate coding concepts from the course
85) Integrate functions, control structures, collections, and other advanced programming features into a final product
86) Test and debug the program using structured evidence
87) Document the design process, code structure, testing, and revisions
88) Present the completed software project and justify technical decisions using appropriate vocabulary

89) Evaluate the effectiveness of the final product and propose next steps for improvement

14. Career and Technical Student Organizations

Objective: *Using the CTSO Resource Document, chapter activities will provide some of the best opportunities CTSO members will have to learn by doing. A successful program of work creates a positive learning atmosphere in the classroom. CTSO members learn how to accept responsibility, work as a team, manage a budget, and handle success and failure. Per the CTE Admin Handbook, CTSOs are embedded into curriculum.*

Professional Development

Prepare each member for entry into the workforce and provide a foundation for success in a career. Becoming a professional does not stop with acquiring a skill, but involves an increased awareness of the meaning of good citizenship and the importance of labor and management in the world of work.

Community Service

Promote and improve goodwill and understanding among all segments of the community through services donated by CTSO chapters, and to instill in its members a lifetime commitment to community service.

Business and Industry Connections

Increase awareness of quality job practices and attitudes, and increase the opportunities for employer engagement and eventual employment.

Financial Leadership

Plan and participate in activities to allow all members to carry out the chapter's projects. Educate members on their own personal finance for current and future success.

Public Relations

Make the general public aware of the good work that students in career and technical education are doing to better themselves and their community, state, nation, and world.

Social Activities

To increase cooperation in the school and community through activities that allow CTSO members to get to know each other in something other than a business or classroom setting.

Academic Standards Alignment

Colorado CTE courses are required to integrate academic and essential skills into course content. The following academic standards have been identified as aligned to this course and should be integrated throughout instruction.

CO CTE Academic Standards Resources (Rubrics, integration strategies, etc)
CDE Colorado Academic Standards Online (Online standards lookup tool)

Note: Mathematics and Science standards will be added in the future. Visit the Colorado Academic Standards page for the most current version.

Reading, Writing and Communicating Crosswalk

Course: Coding 2
Colorado Academic Standards – Reading, Writing, and Communicating

Standard	Prepared Graduate Competency	Description	Course Evidence / Alignment
Standard 1: Oral Expression and Listening	Prepared Graduate 1	Collaborate effectively as group members or leaders	Students work individually and in small groups on complex programming projects, requiring discussion, peer feedback, and shared problem-solving.
Standard 1: Oral Expression and Listening	Prepared Graduate 2	Deliver effective oral presentations	Students explain program logic, algorithms, and design decisions during code reviews, demonstrations, and discussions.
Standard 2: Reading for All Purposes	Prepared Graduate 4	Read a wide range of informational texts	Students read and interpret programming documentation, language references, project specifications, and technical resources.
Standard 2:	Prepared	Understand how language	Students apply

Reading for All Purposes	Graduate 5	functions in different contexts	precise programming language syntax and terminology and adapt explanations of technical concepts for different audiences.
Standard 3: Writing and Composition	Prepared Graduate 7	Craft informational/explanatory texts	Students write code comments, documentation, and explanations describing program structure, logic, and functionality.
Standard 3: Writing and Composition	Prepared Graduate 9	Demonstrate mastery of the writing process	Students iteratively refine code and documentation to produce clear, accurate, and error-free final programs.
Standard 4: Research Inquiry and Design	Prepared Graduate 10	Gather and ethically use information	Students research programming languages, libraries, and algorithms, evaluate source reliability, and follow ethical coding and documentation practices.

Free Resource Links

1. Career Exploration and Postsecondary Opportunities

- freeCodeCamp Learn to Code for Free (career pathways, certifications, project based learning)
  https://www.freecodecamp.org/
- Python.org Documentation (beginner guide, official documentation, Python learning path)
  https://www.python.org/doc/
- Code.org Free Curriculum Commitment (computer science pathways, curriculum access, equity in CS)
  https://code.org/en-US/about/code-org-free-curriculum-commitment

2. Advanced Computational Thinking and Algorithm Design

- Khan Academy The Building Blocks of Algorithms (sequencing, selection, iteration, algorithm design)
  https://www.khanacademy.org/computing/ap-computer-science-principles/algorithms-101/building-algorithms/a/the-building-blocks-of-algorithms
- freeCodeCamp JavaScript Algorithms and Data Structures (algorithms, logic, data structures, problem solving)
  https://www.freecodecamp.org/learn/javascript-algorithms-and-data-structures
- Programiz Learn to Code for Free (algorithm practice, coding examples, problem solving)
  https://www.programiz.com/
- Khan Academy Algorithms Course (computational thinking, efficiency, algorithm concepts)
  https://www.khanacademy.org/computing/ap-computer-science-principles/algorithms-101

3. Data Types, Variables, and Operators

- W3Schools Data Types Tutorial (data types, variables, operators)
  https://www.w3schools.com/python/python_datatypes.asp
- Khan Academy Intro to Variables (variables, data storage, expressions)
  https://www.khanacademy.org/computing/computer-programming/programming
- Programiz Data Types Guide (data types, variables, operators)
  https://www.programiz.com/python-programming/variables-datatypes
- freeCodeCamp Data Types Explained (variables, data types, programming fundamentals)
  https://www.freecodecamp.org/news/data-types-in-programming/

4. Control Structures and Program Flow

- W3Schools Control Flow (conditionals, loops, program flow)
  https://www.w3schools.com/python/python_conditions.asp

- Khan Academy Control Structures (loops, decision making, sequencing) https://www.khanacademy.org/computing/computer-programming/programming/logic
- Programiz Loops and Decisions (loops, conditionals, control flow) https://www.programiz.com/python-programming/if-elif-else
- GeeksforGeeks Control Flow Basics (program flow, conditional logic, loops) https://www.geeksforgeeks.org/control-structures/

5. Functions Methods and Modular Programming

- W3Schools Functions Tutorial (functions, parameters, modular code) https://www.w3schools.com/python/python_functions.asp
- GeeksforGeeks Functions in Programming (functions, modular design, reuse) https://www.geeksforgeeks.org/functions-programming/
- Programiz Python Functions (functions, parameters, return values) https://www.programiz.com/python-programming/function
- Khan Academy Writing Functions (modular programming, abstraction, reuse) https://www.khanacademy.org/computing/computer-programming/programming/functions

6. Scope Lifetime and Code Organization

- W3Schools Variable Scope (scope, local variables, global variables) https://www.w3schools.com/python/python_scope.asp
- GeeksforGeeks Variable Scope (scope, lifetime, memory behavior) https://www.geeksforgeeks.org/scope-of-variables-in-programming/
- Real Python Code Style Guide (code organization, readability, documentation) https://realpython.com/python-pep8/
- Programiz Scope and Lifetime (variables, scope, program behavior) https://www.programiz.com/python-programming/global-local-nonlocal-variables

7. Arrays Collections Structures and Object Oriented Design

- W3Schools Python Lists and Arrays (arrays, collections, data structures) https://www.w3schools.com/python/python_lists.asp
- GeeksforGeeks Object Oriented Programming (classes, objects, OOP concepts) https://www.geeksforgeeks.org/python-oops-concepts/
- Programiz Classes and Objects (OOP, classes, structures) https://www.programiz.com/python-programming/class
- Khan Academy Object Oriented Programming (objects, classes, design) https://www.khanacademy.org/computing/computer-programming/programming/objects

8. Recursion References and Intermediate Programming Techniques

- GeeksforGeeks Recursion Tutorial (recursion, algorithms, problem solving) https://www.geeksforgeeks.org/recursion/
- Programiz Recursion Guide (recursive functions, logic, problem solving) https://www.programiz.com/python-programming/recursion

- freeCodeCamp Recursion Explained (recursion, problem solving, coding strategies)
  https://www.freecodecamp.org/news/recursion-in-programming/
- Khan Academy Recursion Overview (recursive thinking, algorithm design)
  https://www.khanacademy.org/computing/computer-science/algorithms/recursion

9. Testing Debugging and Software Quality

- W3Schools Debugging Guide (debugging, testing, error handling)
  https://www.w3schools.com/python/python_debugging.asp
- GeeksforGeeks Debugging Techniques (testing, debugging strategies, program errors)
  https://www.geeksforgeeks.org/debugging-techniques-in-software-development/
- freeCodeCamp Testing Guide (testing, debugging, software quality)
  https://www.freecodecamp.org/news/software-testing-basics/
- Khan Academy Debugging Programs (debugging, troubleshooting, logic errors)
  https://www.khanacademy.org/computing/computer-programming/programming/debugging

10. Software Development Workflow and Collaborative Coding

- Atlassian Software Development Lifecycle (workflow, development process, collaboration)
  https://www.atlassian.com/software-development
- GitHub Guides (version control, collaboration, project workflow)
  https://guides.github.com
- Trello Project Management Guide (workflow, task management, collaboration)
  https://trello.com/guide
- Scrum Alliance Basics (agile development, teamwork, workflow)
  https://www.scrumalliance.org/why-scrum

11. Work Based Learning

- Advance CTE Resource Center (work based learning, career readiness, industry connections)
  https://careertech.org/resource-center
- OCTAE Work Based Learning Toolkit (workplace learning, employer connections, program design)
  https://octae.ed.gov/wbltoolkit
- CareerOneStop Career Exploration (career pathways, job skills, workforce readiness)
  https://www.careeronestop.org
- O NET OnLine Career Data (job roles, skills, workplace expectations)
  https://www.onetonline.org

12. Leadership in Software Development

- Lead4Change Leadership Program (leadership skills, teamwork, project based learning)
  https://lead4change.org

- Barclays LifeSkills Leadership Lessons (communication, teamwork, leadership development)
  https://barclayslifeskills.com/help-others/lessons/leadership/
- MindTools Leadership Skills (leadership strategies, communication, decision making)
  https://www.mindtools.com
- SkillsYouNeed Leadership Skills (professional skills, teamwork, leadership development)
  https://www.skillsyouneed.com

13. Final Assessment Project

- GitHub Student Developer Pack (project hosting, version control, collaboration)
  https://education.github.com/pack
- Code org Project Gallery (student projects, coding applications, examples)
  https://studio.code.org/projects
- Replit Coding Projects (coding projects, collaboration, development environments)
  https://replit.com
- MIT OpenCourseWare Programming Projects (project examples, programming applications, software development)
  https://ocw.mit.edu

Technology Student Association – National Competitive Events

Coding

Objective: *Students will demonstrate advanced coding knowledge and develop a software solution to an on-site programming challenge aligned with TSA Coding competition requirements.*

- Study advanced coding concepts such as syntax, data structures, control flow, algorithms, functions, recursion, and object-oriented programming.
- Prepare for an individual coding-concepts test used to determine semifinalist placement.
- Analyze an on-site coding problem, evaluation criteria, required tools, and programming constraints.
- Design and develop a software solution within the allotted competition time.
- Use an approved programming language, operating system, or API specified for the challenge.
- Test and debug the program to improve accuracy, functionality, and efficiency.
- Present the completed solution for objective testing and judge evaluation.

Resources

- Technology Student Association – Coding competition guidelines
  https://tsaweb.org/competitions-programs/tsa/themes-problems
- Python Documentation
  https://docs.python.org/3/
- freeCodeCamp – Programming tutorials and practice
  https://www.freecodecamp.org/
- W3Schools – Programming references and examples
  https://www.w3schools.com/

Technology Student Association – National Competitive Events

Software Development

Objective: *Students will design, develop, document, test, and present a software application that responds to the annual TSA Software Development challenge.*

- Analyze the annual TSA Software Development challenge and identify the intended users, purpose, and functional requirements.
- Plan the software application using appropriate design tools such as pseudocode, flowcharts, diagrams, wireframes, or class structures.
- Develop a functioning software product using programming practices appropriate to the selected platform.
- Create documentation that explains the software purpose, development process, features, code structure, testing, and revisions.
- Apply quality assurance practices through testing, debugging, and refinement.
- Prepare a presentation or demonstration that explains the software solution and development decisions.
- Evaluate the final software using TSA criteria for functionality, design quality, documentation, usability, and presentation.

Resources

- Technology Student Association – Software Development competition guidelines https://tsaweb.org/competitions-programs/tsa/themes-problems
- GitHub Docs – Version control and project collaboration https://docs.github.com/
- Replit – Browser-based coding environment https://replit.com/
- Visual Studio Code Documentation https://code.visualstudio.com/docs

Technology Student Association – National Competitive Events

Video Game Design

Objective: *Students will design, develop, test, document, and present an original video game that meets TSA Video Game Design requirements for gameplay, technical quality, creativity, documentation, and presentation.*

- Research the annual TSA Video Game Design theme and develop an original game concept.
- Plan gameplay, rules, goals, mechanics, characters, levels, scoring, user interaction, and technical requirements.
- Develop a playable game using appropriate programming, game engine, or interactive media tools.
- Apply programming concepts such as variables, conditionals, loops, functions, objects, collision logic, and user input.
- Test and refine the game to improve functionality, usability, balance, clarity, and player experience.
- Prepare documentation that explains the game concept, development process, technical tools, testing, and revisions.
- Present the game to judges and explain design decisions, technical implementation, and response to the annual theme.

Resources

- Technology Student Association – Video Game Design competition guidelines https://tsaweb.org/competitions-programs/tsa/themes-problems
- Unity Learn – Game development tutorials https://learn.unity.com/
- Godot Engine Documentation https://docs.godotengine.org/
- Scratch – Beginner-friendly interactive programming and game design https://scratch.mit.edu/