Colorado CTE Course – Scope and Sequence

Course Name	**IT/CS 3 (Information Technology/Computer Science Level 3)**	**Course Details** **Course = 0.50 Carnegie Unit Credit**	**1.0**
Course Description	In this level 3 scope and sequence of the IT/Computer Science curriculum, students delve into more abstract and exploratory concepts. They transition from block-based programming to text-based languages like Python, focusing increasingly on programming in various projects. Introduction to advanced concepts such as recursion and object-oriented programming enriches their understanding. Emphasis is placed on fostering autonomy, encouraging students to seek challenges and support for a fulfilling learning journey. The goal is to nurture a mindset where students view themselves as creators rather than mere consumers of technology. The items listed above provide an overview of the essential knowledge and skills to be taught in computer science modules or courses at each grade level and are not intended to be exhaustive. While also important, we have purposefully omitted skills related to utilizing technology effectively and responsibly (e.g., digital literacy, basic computing, keyboarding, creating documents/spreadsheets/presentations, digital citizenship, and using technology to collaborate or access online content), as these skills should be incorporated into all classes. Additionally, the number of items included in each domain is not indicative of priority or significance. All aspects of this scope and sequence is meant to be exploratory.		
Note:	This is a suggested scope and sequence for the course content. The content will work with any textbook or instructional resource. If locally adapted, make sure all essential knowledge and skills are covered.		
SCED Identification #		Schedule calculation based on 60 calendar days of a 90-day semester. Scope and sequence allow for additional time for guest speakers, student presentations, field trips, remediation, or other content topics.	

All courses taught in an approved CTE program must include Essential Skills embedded into the course content. The Essential Skills Framework for this course can be found at **https://www.cde.state.co.us/standardsandinstruction/essentialskills**

Instructional Unit Topic	Suggested Length of Instruction	CTE or Academic Standard Alignment	Competency / Performance Indicator	Outcome / Measurement	CTSO Integration

Computational Thinking 1. **Decomposition** 2. **Pattern Recognition.** 3. **Abstraction. decomposed problem.** 4. **Algorithmic Thinking.** 5. **Applied Coding & Robotics**	Week 1-2: Introduction to Computational Thinking - Overview of computational thinking concepts: Decomposition, Pattern Recognition, Abstraction, Algorithmic Thinking. - Activities and exercises to introduce each concept, such as problem-solving challenges and group discussions. - Hands-on practice with decomposing problems into smaller parts, recognizing patterns, and abstracting details.	- 2-AP-10 Use flowcharts and/or pseudocode to address complex problems as algorithms. (P4.4, P4.1) - 2-AP-11 Create clearly named variables that represent different data types and perform operations on their values. (P5.1, P5.2) - 2-AP-12 Design and iteratively develop programs that combine control structures, including nested loops and compound conditionals. (P5.1, P5.2) - 2-AP-13 Decompose problems and	Computational Thinking Performance Indicators: Decomposition: - Identify and break down complex problems into smaller, manageable tasks. - Organize the decomposition process into sequential steps. - Demonstrate the ability to identify the relationships between the smaller tasks and the larger problem. Pattern Recognition: - Recognize recurring patterns or similarities within data or problems. - Analyze patterns to derive insights or make predictions. - Apply pattern recognition skills across various contexts, such as in data analysis or algorithm design. Abstraction:	Focus on text-based coding, app and game development, and an introduction to advanced robotics concepts and design thinking challenges. Implement spaced and interleaved practices to tackle coding and robotics problems. Continue with explicit teaching strategies, ensuring students understand the relevance and application of their projects in real-world contexts.	TSA

	Week 3-4: Applied Coding & Robotics ● Introduction to coding concepts using block-based programming languages (e.g., Scratch). ● Activities to create simple programs that apply computational thinking concepts. ● Introduction to robotics and hands-on activities with programmable robots to reinforce coding skills and problem-solving. Week 5-6: Advanced Programming Skills ● Introduction to flowcharts and pseudocode for	subproblems into parts to facilitate the design, implementation, and review of programs. (P3.2) ● 2-AP-14 Create procedures with parameters to organize code and make it easier to reuse. (P4.1, P4.3) ● 2-AP-15 Seek and incorporate feedback from team members and users to refine a solution that meets user needs. (P2.3, P1.1) ● 2-AP-16 Incorporate existing code, media, and libraries into original	● Identify and focus on the essential details while ignoring irrelevant information. ● Represent complex systems or concepts using simplified models or diagrams. ● Utilize abstraction to develop generalized solutions applicable to a range of problems. Algorithmic Thinking: ● Develop step-by-step procedures or algorithms to solve problems. ● Evaluate and refine algorithms for efficiency and effectiveness. ● Apply algorithmic thinking to develop solutions in various domains, including coding, robotics, and data analysis. Applied Coding & Robotics: ● Utilize coding concepts and programming languages to implement solutions		

	algorithm design. - Activities to create algorithms using flowcharts and pseudocode to solve complex problems. - Introduction to variables, data types, and operations in programming languages. - Practice designing and developing programs that combine control structures, including nested loops and compound conditions. Week 7-8: Computing Systems and Hardware - Exploration of computing devices and	programs, and give attribution. (P4.2, P5.2, P7.3) - 2-AP-17 Systematically test and refine programs using a range of test cases. (P6.1) - 2-AP-18 Distribute tasks and maintain a project timeline when collaboratively developing computational artifacts. (P2.2) - 2-AP-19 Document programs in order to make them easier to follow, test, and debug. (P7.2) - Computing Systems - 2-CS-01 Recommend improvements to the design of	to real-world problems. - Design and program robots to perform specific tasks or achieve objectives. - Test, debug, and iterate on code and robotics solutions to optimize performance and functionality.		

	their components. • Activities to analyze how users interact with computing devices and recommend improvements to their design. • Design projects that combine hardware and software components to collect and exchange data. • Introduction to troubleshooting and fixing problems with computing devices and components.	computing devices, based on an analysis of how users interact with the devices. • 2-CS-02 Design projects that combine hardware and software components to collect and exchange data • 2-CS-03 Systematically identify and fix problems with computing devices and their components.			
Computing and Programming. (Demonstrate dispositions compliant with open-ended problem-solving and programming	4-6 weeks for unit Week 1: Introduction to Computing and Programming:	• 2-CS-01 Recommend improvements to the design of computing devices, based	• Write programs using text-based programming languages. • Locate and debug errors in a program. • Read a program and translate it into English.	Students will exhibit dispositions-compliant with open-ended problem-solving and programming, showcasing their comfort with complexity, persistence,	TSA

(e.g., comfort with complexity, persistence, brainstorming, adaptability, patience, propensity to tinker, creativity, accepting challenge)	● Overview of computing concepts and programming fundamentals. ● Introduction to the importance of open-ended problem-solving and programming dispositions. ● Activities and discussions to introduce dispositions such as comfort with complexity, persistence, creativity, and adaptability. Week 2-3: Designing and Implementing Projects ● Introduction to project-based learning and its application to computing and programming. ● Design projects that combine	on an analysis of how users interact with the devices. ● 2-CS-02 Design projects that combine hardware and software components to collect and exchange data ● 2-CS-03 Systematically identify and fix problems with computing devices and their components.	Explain how a particular program functions. ● Design, code, test, and execute a program corresponding to a set of specifications. ● Design, develop, publish, and present products (e.g., web pages, mobile apps, animations) ● to demonstrate and communicate curriculum concepts. Performance Indicator Comfort with Complexity: ● Engage with complex problems without feeling overwhelmed. ● Demonstrate confidence in tackling challenging tasks or projects. ● Seek out opportunities to explore and learn from complex scenarios. Persistence: ● Demonstrate perseverance when faced with obstacles or setbacks.	brainstorming, adaptability, patience, propensity to tinker, creativity, and acceptance of challenges. Measurable Outcome: After the level 3 Computing Practice and Programming curriculum, students will design, develop, and execute a text-based program that meets specified criteria, debug any errors encountered during the coding process, and effectively communicate the functionality of their program in English. **Performance Indicators:** Comfort with Complexity: Students will successfully engage with a complex programming problem, demonstrating confidence and curiosity in tackling challenging tasks. Students will actively seek out opportunities to explore and learn from complex scenarios	

	hardware (such as microcontrollers or sensors) and software (programming languages or block-based coding platforms) components to collect and exchange data. ● Hands-on activities to develop and implement projects, focusing on problem-solving, creativity, and adaptability. Week 4: Analyzing User Interaction and Device Design ● Introduction to the design of computing devices and user interaction principles. ● Activities to analyze how		● Continuously work towards solutions even when progress is slow. ● Show resilience in the face of failure and use it as an opportunity for learning and growth. Brainstorming: ● Generate a variety of ideas and approaches when problem-solving. ● Encourage collaboration and open discussion to explore different perspectives. ● Utilize brainstorming techniques to generate innovative solutions to problems. Adaptability: ● Adjust strategies and approaches based on new information or changing circumstances. ● Embrace flexibility in problem-solving methods and	encountered during the programming process. Persistence: Students will demonstrate perseverance when faced with obstacles or setbacks during the programming process, continuously working towards solutions even when progress is slow. Students will exhibit resilience in the face of programming errors or failures, using them as opportunities for learning and growth. Brainstorming: Students will generate a variety of ideas and approaches to solve programming problems, encouraging collaboration and open discussion to explore different perspectives. Students will utilize brainstorming techniques to generate innovative solutions to programming challenges encountered.	

	users interact with computing devices and recommend improvements to their design. ● Discussions and case studies exploring real-world examples of device design and user experience considerations. Week 5: Troubleshooting and Problem-Solving ● Introduction to systematic problem-solving strategies and troubleshooting techniques. ● Activities to identify and fix problems with computing devices and their components. ● Hands-on troubleshooting exercises and		programming techniques. ● Demonstrate the ability to adapt to different tools, languages, or platforms as needed. Patience: ● Exhibit patience when debugging code or troubleshooting technical issues. ● Take the time to thoroughly understand problems before attempting solutions. ● Recognize that mastery takes time and effort, and demonstrate patience in the learning process. Propensity to Tinker: ● Show curiosity and a willingness to experiment with technology and programming. ● Explore different features, settings, and functionalities	Adaptability: Students will adjust programming strategies and approaches based on new information or changing circumstances encountered during the coding process. Students will demonstrate flexibility in problem-solving methods and programming techniques, adapting to different tools, languages, or platforms as needed. Patience: Students will exhibit patience when debugging code or troubleshooting technical issues, taking the time to thoroughly understand problems before attempting solutions. Students will recognize that mastering programming concepts takes time and effort, demonstrating patience in the learning process. Propensity to Tinker:	

	simulations to practice systematic problem-solving skills. Week 6: Culminating Project and Reflection ● Culminating project where students apply their knowledge and skills in computing and programming to design and implement a final project. ● Presentations or demonstrations of projects to peers, teachers, or external audiences. ● Reflection activities to review learning outcomes, assess disposition development, and set goals for future learning.		to understand their effects. ● Embrace tinkering as a means of discovering new possibilities and gaining deeper insights. Creativity: ● Think outside the box and generate novel solutions to problems. ● Combine existing ideas or techniques in innovative ways to create unique outcomes. ● Embrace creativity as an essential aspect of programming and problem-solving. Accepting Challenges: ● Welcome challenges as opportunities for growth and learning. ● Approach difficult tasks with a positive attitude and a willingness to persevere. ● View challenges as a chance to push boundaries and	Students will show curiosity and a willingness to experiment with different programming concepts, exploring various features, settings, and functionalities to gain deeper insights. Students will embrace tinkering as a means of discovering new possibilities and refining their programming skills. Creativity: Students will think creatively to generate novel solutions to programming problems, combining existing ideas or techniques in innovative ways to create unique outcomes. Students will recognize and embrace creativity as an essential aspect of programming and problem-solving. Accepting Challenge: Students will welcome programming challenges as opportunities for growth and learning,	

			expand one's capabilities.	approaching difficult tasks with a positive attitude and a willingness to persevere.	
Programming Skills (Continued) ● **Iteration: Nested Loops** ● **Conditional Statements** ● **Randomization** ● **Functions**	2-4 weeks	● 2-AP-10 Use flowcharts and/or pseudocode to address complex problems as algorithms. (P4.4, P4.1) ● 2-AP-11 Create clearly named variables that represent different data types and perform operations on their values. (P5.1, P5.2) ● 2-AP-12 Design and iteratively develop programs that combine control structures, including nested loops	Nested Loops Demonstrate the ability to implement nested loops to iterate through multidimensional data structures effectively. Utilize nested loops to solve complex problems requiring multiple levels of iteration, such as matrix operations or nested patterns. Apply nested loops in algorithmic solutions to real-world problems, demonstrating efficiency and elegance in code design. Implement conditional statements to create robust and adaptive programs capable of responding dynamically to varying inputs and scenarios. Skill in Randomization Utilize randomization techniques to introduce variability and unpredictability in program	Outcomes: Measurements Mastery of Iteration: Nested Loops ● Students will demonstrate the ability to implement nested loops through a series of coding exercises and projects. ● Performance assessments will include analyzing and debugging code that utilizes nested loops to iterate through multidimensional data structures. ● Students will complete a project where they apply	TSA

		and compound conditionals. (P5.1, P5.2) ● 2-AP-13 Decompose problems and subproblems into parts to facilitate the design, implementation, and review of programs. (P3.2) ● 2-AP-14 Create procedures with parameters to organize code and make it easier to reuse. (P4.1, P4.3) ● 2-AP-15 Seek and incorporate feedback from team members and users to refine a solution that meets user needs. (P2.3, P1.1)	behavior, enhancing user experience and realism. Apply random number generation functions to simulate probabilistic events or create randomized elements in games and simulations. Incorporate randomness judiciously, ensuring that randomization serves a purpose and enhances the functionality or entertainment value of the program. Proficiency in Functions: Define and implement functions to encapsulate reusable code blocks, promoting modularity and code organization. Design functions with clear input parameters and return values, adhering to principles of abstraction and encapsulation. Utilize functions to decompose complex tasks into smaller, manageable units, enhancing code readability and maintainability.	nested loops to solve complex problems, such as matrix operations or generating nested patterns. ● Assessment rubrics will evaluate the efficiency, correctness, and elegance of students' nested loop implementations. Proficiency in Conditional Statements ● Students will create programs that utilize conditional statements to respond dynamically to varying inputs and scenarios. Performance assessments will include coding challenges where	

		• 2-AP-16 Incorporate existing code, media, and libraries into original programs, and give attribution. (P4.2, P5.2, P7.3) • 2-AP-17 Systematically test and refine programs using a range of test cases. (P6.1) • 2-AP-18 Distribute tasks and maintain a project timeline when collaboratively developing computational artifacts. (P2.2) • 2-AP-19 Document programs in order to make them easier to follow, test,		students design algorithms that incorporate conditional statements to handle different cases. • Students will complete a project where they implement conditional statements to create robust and adaptive programs. • Assessment rubrics will evaluate the effectiveness and correctness of students' conditional statements, assessing their ability to handle various conditions and scenarios. Skill in Randomization	

		and debug. (P7.2) ● Computing Systems ● 2-CS-01 Recommend improvements to the design of computing devices, based on an analysis of how users interact with the devices. ● 2-CS-02 Design projects that combine hardware and software components to collect and exchange data ● 2-CS-03 Systematically identify and fix problems with computing devices and their components.		● Students will demonstrate the use of randomization techniques through coding exercises and projects. ● Performance assessments will include analyzing and modifying code that utilizes randomization functions to introduce variability and unpredictability. ● Students will complete a project where they incorporate randomization to simulate probabilistic events or create randomized elements in games or simulations. ● Assessment rubrics will evaluate the	

				purposefulness and effectiveness of randomization in enhancing program functionality and user experience. Proficiency in Functions ● Students will define and implement functions through coding exercises and projects. ● Performance assessments will include analyzing and debugging code that utilizes functions to encapsulate reusable code blocks. ● Students will complete a project where they design and implement functions to decompose complex tasks	

				into smaller, manageable units. ● Assessment rubrics will evaluate the clarity of students' function definitions, the effectiveness of function usage, and the adherence to principles of modularity and code organization.	
Computers and Communication Devices Describe the components and functions of computer systems and networks. Apply strategies for identifying and solving routine problems that occur during everyday computer use.	2-4 weeks	2-NI-04 Model the role of protocols in transmitting data across networks and the Internet. 2-CS-01 Recommend improvements to the design of computing devices, based on an analysis of how users interact with the devices.	Competency: Computers and Communication Devices Performance Indicators: Describe the Components and Functions of Computer Systems and Networks: Identify and describe the major components of a computer system, including the CPU, memory, storage devices, input/output devices, and peripherals.	Outcome: Describe the Components and Functions of Computer Systems and Networks Measurement: Students will accurately identify and describe the major components of a computer system, including the CPU, memory, storage devices, input/output	TSA

		2-CS-02 Design projects that combine hardware and software components to collect and exchange data.Design projects that combine hardware and software components to collect and exchange data.	Explain the function and role of each component within the computer system, highlighting their interactions and contributions to overall system functionality. Apply Strategies for Identifying and Solving Routine Problems in Computer Use: Develop systematic approaches for identifying common issues that arise during everyday computer use, such as software glitches, connectivity problems, or hardware malfunctions. Utilize troubleshooting techniques to diagnose and isolate the root causes of computer problems, employing methods like trial and error, process of elimination, and systematic testing. Apply critical thinking skills to analyze symptoms, gather relevant information, and implement appropriate solutions to resolve computer-related issues efficiently. These performance indicators assess students' competency in understanding computer systems and networks, as well as their ability to identify and solve routine problems encountered during	devices, and peripherals. Performance assessments will include quizzes, tests, or presentations where students demonstrate their understanding of computer system components and their roles. Students will complete a project or assignment where they explain the function and interaction of each component within a computer system, highlighting their contributions to overall functionality. Assessment rubrics will evaluate the completeness, accuracy, and clarity of students' descriptions of computer system components and functions. Outcome: Apply Strategies for Identifying and Solving Routine	

			everyday computer use. By mastering these essential skills, students develop a foundational understanding of computing devices and gain practical problem-solving abilities that are valuable in both academic and real-world contexts.	Problems in Computer Use Measurement: Students will develop systematic approaches for identifying and solving common problems encountered during everyday computer use. Performance assessments will include scenarios or case studies where students apply troubleshooting techniques to diagnose and resolve computer-related issues. Students will demonstrate critical thinking skills by analyzing symptoms, gathering relevant information, and implementing appropriate solutions to solve computer problems efficiently.	

				Assessment rubrics will evaluate students' ability to apply troubleshooting techniques effectively, analyze symptoms, and implement solutions to resolve computer problems.	
Community, global, and ethical impacts Use information and technology responsibly and ethically. Analyze the effects of computing on society within economic, social, and cultural contexts. Describe the widespread impact of the internet in connecting people and ideas from across the world. Use computing to positively impact the community.	3-4 weeks	2-IC-21 Discuss issues of bias and accessibility in the design of existing technologies. 2-IC-22 Collaborate with many contributors through strategies such as crowdsourcing or surveys when creating a computational artifact. 2-IC-23 Describe tradeoffs between allowing information to be public and keeping information private and secure.	Competency: Community, Global, and Ethical Impacts of Computer Science and IT (Level 3) Performance Indicators: Use Information and Technology Responsibly and Ethically: Demonstrate an understanding of ethical considerations related to information and technology use, including issues such as privacy, security, intellectual property rights, and digital citizenship. Apply ethical principles to decision-making in various technology-related contexts, making informed choices that prioritize integrity, honesty, and respect for others' rights and interests. Act responsibly in digital environments, adhering to established guidelines and protocols for safe and ethical online behavior,		TSA

			and demonstrating awareness of the potential consequences of unethical actions. Analyze the Effects of Computing on Society within Economic, Social, and Cultural Contexts: Identify and analyze the economic, social, and cultural impacts of computing technologies on individuals, communities, and society at large. Evaluate the role of technology in shaping economic structures, employment opportunities, social interactions, and cultural practices, considering both positive and negative consequences. Critically examine issues such as digital divide, algorithmic bias, and technological displacement, and propose strategies for addressing or mitigating their effects. Describe the Widespread Impact of the Internet in Connecting People and Ideas from Across the World: Explain the transformative role of the Internet in facilitating global communication, collaboration, and information exchange. Identify and describe various internet technologies and platforms		

			that enable connections between individuals, communities, and organizations worldwide. Explore examples of how the internet has influenced cultural exchange, political activism, economic development, and other aspects of global society. Use Computing to Positively Impact the Community:		
Collaboration ● Work cooperatively and collaboratively with peers, teachers, experts, and others. ● Engage in pair programming, as both driver and navigator. ● Exhibit dispositions necessary for collaboration: providing useful feedback, integrating feedback, ● understanding and accepting	Entirety of course	CSTA 2-IC-22	Competency: Collaboration in Computer Science and IT (Level 3) Performance Indicators: Work Cooperatively and Collaboratively with Peers, Teachers, Experts, and Others: Actively participate in group projects, discussions, and activities, demonstrating a willingness to contribute ideas, share resources, and collaborate with others. Communicate effectively with team members, teachers, and external stakeholders, demonstrating active listening skills, clear expression of ideas, and respectful interaction. Engage in Pair Programming, as Both Driver and Navigator:	Outcome 1: Work Cooperatively and Collaboratively with Peers, Teachers, Experts, and Others Measurement: Students will actively participate in group projects, discussions, and activities, as evidenced by their contributions, ideas shared, and	TSA

multiple perspectives, and socialization.			Collaborate effectively in pair programming activities, alternating roles between the "driver" (writing code) and the "navigator" (providing guidance and feedback). Demonstrate effective communication and teamwork skills while working as a pair, actively discussing ideas, sharing insights, and jointly solving programming challenges. Utilize pair programming as a strategy to enhance learning, promote code quality, and build problem-solving skills through shared exploration and dialogue. Integrate Feedback: Incorporate feedback received from peers, teachers, and experts into one's work, demonstrating flexibility and adaptability in response to suggestions and critiques. Iterate projects and solutions based on feedback, striving for continuous improvement and refinement. Understand and Accept Multiple Perspectives: Respect and value diverse perspectives and ideas shared by peers and collaborators, recognizing	collaboration with others. Performance assessments will include peer evaluations and teacher observations of students' communication and collaboration skills during group activities. Students will complete a group project where they demonstrate effective communication, resource-sharing, and collaboration with team members. Assessment rubrics will evaluate students' willingness to contribute ideas, share resources, and engage in respectful	

			the importance of diversity in problem-solving and innovation.	interaction with peers and collaborators. Outcome: Engage in Pair Programming, as Both Driver and Navigator Measurement: Students will demonstrate effective pair programming skills through collaborative coding activities. Performance assessments will include observations of students' roles as both driver (writing code) and navigator (providing guidance and feedback) during pair programming sessions. Students will complete pair programming	

				exercises where they actively discuss ideas, share insights, and jointly solve programming challenges with their partner. Assessment rubrics will evaluate students' communication, teamwork, and problem-solving skills demonstrated during pair programming activities. Outcome:: Integrate Feedback Measurement: Students will incorporate feedback received from peers, teachers, and experts into their work. Performance assessments will	

				include analyses of students' ability to iterate projects and solutions based on feedback received. Students will revise and refine their projects or solutions based on feedback provided by peers, teachers, or external stakeholders. Assessment rubrics will evaluate students' flexibility and adaptability in responding to feedback, as well as the quality of their revised work. Outcome 4: Understand and Accept Multiple Perspectives Measurement:	

				Students will demonstrate respect for diverse perspectives and ideas shared by peers and collaborators. Performance assessments will include reflections on students' recognition of the importance of diversity in problem-solving and innovation. Students will engage in discussions and activities that promote understanding and acceptance of multiple perspectives within the context of computer science and IT. Assessment rubrics will evaluate students' ability to respect and value diverse	

				viewpoints, as well as their contributions to creating an inclusive and collaborative learning environment.	
Careers in IT	2-4 weeks		Identify various fields within IT fields and their respective career opportunities. a. Recognize the work typically performed, tools and technology used, and nature of work environment b. Identify potential certification opportunities c. Find membership organizations associated with the careers d. Understand the necessary education associated within the careers e. Research security clearance requirements associated within the careers		