Utah Computer Science
3rd Grade Course Syllabus
One Year for Elementary School, 36 Hours

Course Overview and Goals

The Utah Computer Science 3rd Grade Course introduces students to foundational programming concepts through block-based programming language. Students will develop computational thinking and problem-solving skills while learning to create interactive projects, animations, and games. This course emphasizes creativity and collaboration, providing students with a solid base in computer science concepts and digital literacy.

Learning Environment: This course is teacher-led and includes ready-to-use lessons following a consistent structure: Introduction, Guided Practice, Independent Practice, Extension, and Reflection. Instruction follows an "I do, we do, you do" model and incorporates spiral review to reinforce concepts and build confidence over time.

The course includes 36 lessons, each approximately 45 minutes long, providing a full year of instruction when taught once per week. While the course allows for instructional flexibility, some lessons are required to fully meet state computer science standards and are clearly identified within the syllabus. All Digital Literacy lessons are required to ensure full standards alignment, as they address essential non-programming computer science concepts. All lessons are labeled with the specific standards they address to support planning and compliance.

Standards Alignment Note: Standards addressed in each lesson are listed in parentheses after the lesson title. Standards may be reinforced across multiple lessons, and full coverage of state computer science standards is achieved through the course as a whole.

Programming Environment: Students will write and run programs that are saved in students' accounts. The environment supports interactive, hands-on programming, enabling students to create and debug projects in a user-friendly interface.

Assessments: Some units conclude with a brief five-question quiz designed to assess student understanding of the concepts and skills introduced in the unit. Quizzes can be assigned digitally, completed as a QuizBuzz activity, or printed for paper-based completion.

Prerequisites: There are no prerequisites for this course. It is designed to support all learners, regardless of prior computer science experience.

More Information: Browse the content of this course at https://codehs.com/course/26171/overview



A clickable PDF can be found at https://codehs.com/UT-CSRoadmaps

Course Breakdown

Unit 1: Getting Started (3 lessons)

In this introductory unit, students explore how computers work and what to do when something goes wrong. They'll identify key parts of computing systems, apply troubleshooting steps, and think like computer scientists by using computational thinking to creatively design an obstacle course.

Objectives / Topics Covered	● Identify and describe the components of a computing system and how they work. ● Apply basic troubleshooting steps to solve common computer problems. ● Use computational thinking to break down problems, recognize patterns, and design a solution.
Lessons	**Welcome to CodeHop! (15 minute lesson)** ● Practice logging in and exploring the CodeHS Playground before starting a full lesson. **Introduction to Computing Systems** (3.CS.1) ● Learn about the parts of a computer system and how they work together, then apply simple strategies to fix common hardware and software problems. **Computational Thinking: Design an Obstacle Course** (3.CT.1, 3.CT.2) ● Use computational thinking to plan and design a new course by breaking the problem into parts, finding patterns, and sequencing ideas logically. **Getting Started Quiz (5 questions)**

Unit 2: Programming Introduction (5 lessons)

In this story-driven unit, students will follow Scout on an animated adventure as they learn to use CodeHop. They'll develop core programming skills like sequencing, using events and loops, and customizing their animations with drawing tools—all while creating interactive stories.

Objectives / Topics Covered	● Program characters with motion, dialogue, and animations. ● Sequence blocks to create interactive stories. ● Apply events, loops, and looks blocks to enhance animations. ● Customize characters and backgrounds using drawing tools.
Lessons	**Scout's Programming Expedition Part 1** (3.AP.3, 3.CT.1) ● Use basic blocks to program Scout the sprite to move and talk in the beginning of the story. **Scout's Programming Expedition Part 2** (3.AP.3, 3.CT.1) ● Continue the story by adding sprites and creating a sequence that animates a simple scene. **Scout's Programming Expedition Part 3** (3.AP.3, 3.CT.1) ● Create a looping animation using events, looks, and motion blocks as the story becomes more interactive. **Scout's Programming Expedition Part 4** (3.AP.3, 3.CT.1) ● Add complexity to the story by combining events, motion, and looks to animate multiple characters. **Drawing Tools** (3.AP.1, 3.AP.3) ● Explore built-in drawing tools to design custom sprites and backdrops for your projects. **Programming Introduction Quiz (5 questions)**

Unit 3: Sequences & Events (6 lessons)

In this unit, students will deepen their understanding of how sequences and events work. They'll explore parallel programming, animated sprites, interactive backdrops, and broadcast messages to create engaging and dynamic programs—including an original story that brings their ideas to life.

Objectives / Topics Covered	● Create programs using sequences, events, and parallel actions. ● Animate sprites and switch backdrops to enhance storytelling. ● Use broadcast messages to coordinate actions between characters. ● Plan and program original stories with multiple interactive elements.
Lessons	**Sequences: Parallel Programming** (3.AP.3) ● Create a program where multiple sprites perform actions at the same time using parallel sequences. **Events** (3.AP.1) ● Use events to trigger actions in a program, such as when the green flag is clicked or a sprite is tapped. **Costumes, Backdrops, and Animations** (3.AP.1, 3.AP.3) ● Animate sprites by changing costumes and switching between backdrops to create interactive scenes. **Broadcast Messages: Marco Polo** (3.AP.1, 3.AP.3) ● Use broadcast and receive blocks to trigger communication between sprites in a Marco Polo–style animation. **Creative Storytelling** (2 classes | 3.AP.1) ● Plan and animate a custom story using sequences and events to bring characters and scenes to life. **Sequences & Events Quiz (5 questions)**

Unit 4: Loops (5 lessons)

In this unit, students explore how loops help simplify code by repeating actions. They'll practice using loops to build efficient programs, work through debugging challenges, and collaborate with a partner to create an interactive digital band using keyboard inputs.

Objectives / Topics Covered	● Understand how loops repeat instructions in a program. ● Use loops to simplify and organize code. ● Debug programs that involve loops and events. ● Collaborate with peers through pair programming to build interactive projects.
Lessons	**Loops** (3.AP.1, 3.AP.3) ● Learn how a loop repeats instructions and use loops to animate or control characters in a program. **Debugging: Events and Loops** (3.AP.3) ● Break down a program into smaller parts to find and fix errors involving events and loops. **Remixing Programs** (3.AP.6, 3.IC.2) ● Create or remix a project while practicing digital citizenship by giving credit to original creators and understanding copyright basics. **Pair Programming: Create a Band** (2 classes | 3.AP.1, 3.AP.3, 3.AP.4, 3.AP.5, 3.CT.1) ● Work with a partner to design and code a band using keyboard inputs and loops to make instruments play. **Loops Quiz (5 questions)**

Unit 5: Conditionals (4 lessons)

In this unit, students dive into conditionals to learn how computers make decisions. They'll use if/then and if/then/else blocks to control what happens under certain conditions in a program. Alongside coding practice, students will explore how to remix projects responsibly and give proper credit to original creators.

Objectives / Topics Covered	● Understand and use conditionals to control program behavior. ● Create programs that respond to specific conditions using if/then and if/then/else blocks. ● Develop awareness of copyright, plagiarism, and giving credit when remixing digital content.
Lessons	**Introduction to Conditionals** (3.AP.1) ● Learn what a conditional is and build a program that uses if/then blocks to make decisions. **Conditionals: Color Sense** (3.AP.1) ● Use conditionals to trigger actions when specific colors are touched in a program. **Conditionals: Changing Effects** (3.AP.1) ● Create a program that uses if/then logic to change how sprites behave or react. **Complex Conditionals: Balloon Game** (3.AP.1) ● Explore if/then/else conditionals by building a balloon-popping game that responds to different outcomes. **Conditionals Quiz (5 questions)**

Unit 6: Variables & Lists (4 lessons)

In this unit, students explore how variables and lists are used to store and manage information. They'll build programs that track values, make comparisons using conditions, and create an I-Spy–style game that uses both variables and lists to make projects more interactive.

Objectives / Topics Covered	● Understand what a variable is and how it can be used to store changing information. ● Use comparison operators in conditionals with variables and numbers. ● Create and use lists to organize and manage multiple pieces of information. ● Apply variables and lists to build interactive games and animations.
Lessons	**Variables** (3.AP.1) ● Create a program that uses a variable to store and change a value, such as keeping score. **Introduction to Comparison Operators** (3.AP.1, 3.AP.2) ● Use comparison operators (>, <, =) with numbers and variables to build conditional logic using if/else blocks. **I-Spy with Lists** (2 classes | 3.AP.1, 3.AP.2) ● Design an I-Spy–style game using lists to track hidden objects and variables to manage interactions. **Variables & Lists Quiz (5 questions)**

Unit 7: Functions (3 lessons)

In this unit, students are introduced to functions as a way to organize and reuse code. They'll learn how to create custom blocks to simplify their programs and build a fun dance project that uses functions to control movement in sync with music.

Objectives / Topics Covered	● Understand what a function is and how it helps organize code. ● Create and use custom blocks (functions). ● Apply functions to simplify repeated actions in a sequence.

	● Build a creative project that uses functions to control timing and choreography.
Lessons	**Introduction to Functions** (3.AP.1, 3.AP.3) ● Create and use custom blocks (functions) to organize and reuse code in a project. **Functions Dance Project** (2 classes | 3.CT.2) ● Build a dance animation by creating functions for different moves and sequencing them to match music. **Functions Quiz (5 questions)**

Unit 8: Culmination Project (4 lessons)

In this final unit, students apply the skills they've learned throughout the course to complete creative, inquiry-based projects. Students demonstrate mastery of key concepts like conditionals, variables, and lists.

Objectives / Topics Covered	● Apply core programming skills—like conditionals, variables, and lists—to design interactive projects. ● Use the inquiry process to collect, analyze, and display data through coding.
Lessons	**Race Track Game** (2 classes | 3.AP.1, 3.AP.2) ● Design a race car and track, then program a racing game that includes scorekeeping with conditionals, variables, and lists. **Inquiry Project: Survey Bar Graph** (2 classes | 3.AP.2, 3.DA.1, 3.DA.2) ● Follow the inquiry process to investigate a topic and modify a program to display survey results using a bar graph.

Digital Literacy (3 lessons)

In this unit, students explore how computing impacts people and communities while developing safe and responsible technology habits. They learn to evaluate information, recognize online safety practices, and analyze data to make informed decisions.

Objectives / Topics Covered	● Explore how computing impacts people and communities. ● Identify strategies for staying safe online. ● Evaluate data and determine whether information is reliable. ● Analyze data to draw conclusions and make predictions.
Lessons	**Impacts of Computing: Introduction** (3.IC.1) ● Explore how computing has changed over time and its impact on people and communities. **Cybersecurity Introduction** (3.NI.1, 3.NI.2) ● Identify common online threats and explore ways to stay safe online. **Data Detectives** (3.DA.2) ● Evaluate data for reliability and use it to draw conclusions and make predictions. **Digital Literacy Quiz (5 questions)**

Optional: Artificial Intelligence (3 lessons)

In this optional unit, students explore how artificial intelligence learns from data, recognizes patterns, and helps people solve problems. Through hands-on activities, students investigate how AI systems are trained, how new technologies are created, and how to communicate effectively with AI.

Objectives / Topics Covered	● Explain how AI learns from data. ● Explore how people design and improve AI technologies. ● Identify ways AI can help solve problems.

	• Write clear prompts and evaluate AI responses.
Lessons	**Introduction to Training AI** (3.AP.1, 3.DA.2) • Explore how AI learns by using training data. **Creating Smart Machines** (3.AP.1, 3.CT.1) • Explore how people design technologies to solve problems. **Ask AI Better Questions** (3.IC.1) • Write clear prompts and evaluate AI responses. **Artificial Intelligence Quiz (5 questions)**

Utah Computer Science 3rd Grade Course Supplemental Materials

Resources	**Description**
Parent Welcome Letter (Spanish)	Send this letter home to introduce families to their new computer science curriculum.
Warm-Up Activities	This warm-up activity slide deck provides 5-10 minute problems aligned with computer science skills to engage students at the start of class, allowing teachers to preview or review concepts with answer keys and discussion tips included in the Speaker Notes.
Program Self-Assessment (Spanish)	This is a student self-assessment tool designed to help K-6 learners reflect on their programming projects, evaluate their skills in algorithms, debugging, collaboration, and reflection, and set goals for improvement.
Peer Review Resources (Spanish)	This provides structured worksheets to facilitate student feedback during collaborative coding projects. It encourages reflection by guiding students to highlight successes, ask questions, and offer constructive feedback on their partner's work.
Lesson Reflection & Computational Thinking (Spanish)	This guides students in engaging with computational thinking concepts, preparing for discussions, reflecting on lessons, and applying their learning to real-world problem-solving.
Design-Your-Own-Lesson Templates	Empower your students to explore and express their knowledge creatively with our versatile graphic organizer templates. Designed with adaptability and ease of use in mind, these interactive tools transform any subject into an engaging, hands-on learning experience.
These resources and more are found on the **CodeHop Resources Page**.	