Utah Computer Science 4th Grade Course Syllabus

One Year for Elementary School, 36 Hours

Course Overview and Goals

The **Utah Computer Science 4th Grade Course** introduces students to foundational programming concepts through **Scratch**, a 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.

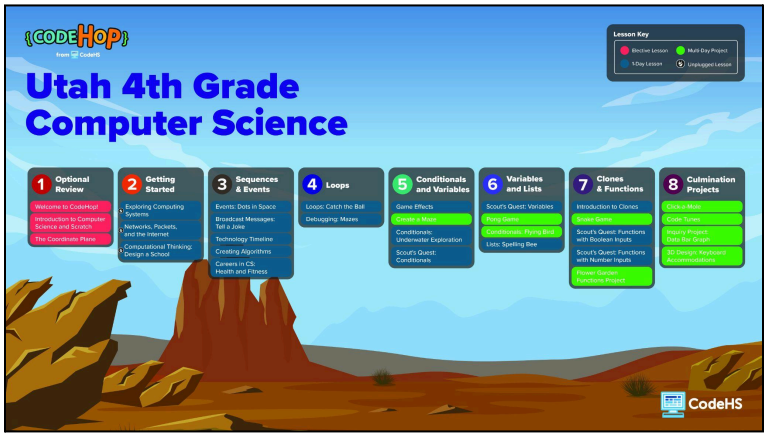
Learning Environment: This course is designed to be teacher-led, with ready-to-use lesson plans that follow a structured format: **Introduction, Guided Practice, Independent Practice, Extension, and Reflection**. Lessons are built with spiral review to reinforce key concepts and culminate in engaging projects to showcase student understanding.

The lessons are delivered in an **"I do, we do, you do"** format, ensuring a gradual release of responsibility and fostering confidence in students as they learn. Teachers can adapt the content to fit their schedule and instructional needs. The concepts taught in this course spiral across grade levels, ensuring that students can revisit and build upon their understanding year after year, even if all lessons are not completed within a single year. The course includes a total of 36 **lessons**, with each lesson approximately 45 minutes long. This provides a full school year of material if teaching one lesson per week. Optional digital literacy lessons are also available to complement the programming curriculum with non-programming computer and technology skills.

Programming Environment: Students will write and run programs in **Scratch** embedded and saved in the CodeHS platform. The environment supports interactive, hands-on programming, enabling students to create and debug projects in a user-friendly interface.

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/26172/overview



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

Course Breakdown

Optional Review

This optional review unit helps students refresh foundational computer science skills and get reacquainted with Scratch. It includes lessons on logging in, reviewing vocabulary, and practicing programming with coordinates and simple animations.

Objectives / Topics Covered	● Log in and navigate the Playground. ● Review key computer science terms and Scratch basics. ● Use the coordinate plane in Scratch to control sprite movement.
Lessons	**Welcome to CodeHop! (15 minute lesson)** ● Practice logging in and exploring the Playground before starting a full lesson. **Introduction to Computer Science and Scratch** ● Review basic computer science vocabulary and create a simple Scratch program. **The Coordinate Plane** ● Use the coordinate plane in Scratch to design an open-ended animation.

Unit 1: Getting Started (3 lessons)

In this introductory unit, students build a foundational understanding of how computing systems work, how the internet communicates using data packets, and how to solve problems using computational thinking. Through real-world modeling and design activities, they prepare for deeper exploration of computer science concepts throughout the course.

Objectives / Topics Covered	● Identify components of computing systems and explain how they work together. ● Apply basic troubleshooting strategies to solve hardware and software issues. ● Understand how the internet sends information using packets and secure communication methods. ● Use computational thinking skills to plan and design a creative solution.
Lessons	**Exploring Computing System** ● Identify parts of a computing system, describe how they work together, and practice simple troubleshooting steps. **Networks, Packets, and the Internet** ● Model how information travels through the internet as packets, and design a secure way to communicate in the classroom. **Computational Thinking: Design a School** ● Use computational thinking strategies like decomposition and sequencing to design a school with creative problem-solving.

Unit 2: Sequences & Events (5 lessons)

In this unit, students expand their programming skills by using sequences, multiple event types, and broadcast messages to control interactions between sprites. They'll also explore real-world applications of computer science by designing timelines, comparing algorithms, and creating animations that connect technology to culture and health.

Objectives / Topics Covered	● Use various event blocks and sequences to control program flow. ● Apply broadcast messages to create sprite interactions. ● Build interactive projects that integrate timelines and storytelling. ● Compare multiple algorithms to determine efficiency and effectiveness.

	● Explore careers in computer science and how coding connects to health and culture.
Lessons	**Events: Dot in Space** ● Create a program using multiple event blocks to control sprite behavior in a space-themed animation. **Broadcast Messages: Tell a Joke** ● Use broadcast and receive blocks to animate two sprites telling a knock-knock joke together. **Technology Timeline** ● Design an interactive timeline showing the evolution of music player technology and its impact on culture. **Creating Algorithms** ● Build and compare different algorithms to decide which one works best for solving a task. **Careers in CS: Health and Fitness** ● Animate how technology supports health and fitness, and learn how coding can help people reach their goals.

Unit 3: Loops (2 lessons)

In this unit, students deepen their understanding of loops by using them to control repetition in games and animations. They'll apply both finite and infinite loops in Scratch and learn to debug programs by breaking them down into manageable parts.

Objectives / Topics Covered	● Understand and use different types of loops to simplify code. ● Apply loops in interactive Scratch programs. ● Decompose and debug programs to identify and fix errors involving loops.
Lessons	**Loops: Catch the Ball** ● Use repeat and forever loops to build a simple ball-catching game in Scratch. **Debugging: Mazes** ● Break down a maze program into smaller parts to find and fix errors involving loops and movement.

Unit 4: Conditionals & Operators (4 lessons)

In this unit, students explore how to use conditionals and operators to make their programs more interactive and responsive. They'll apply these concepts by building games, customizing effects, and using logic to guide characters through mazes and underwater scenes—all while incorporating feedback and practicing skill review in Scout's Quest.

Objectives / Topics Covered	● Use if/then conditionals and operators to control actions based on specific conditions. ● Apply conditionals in game logic to enhance interactivity. ● Create custom visuals like mazes and game effects. ● Improve programs through peer feedback and reflection.
Lessons	**Game Effects** ● Modify an existing game by adding creative effects and making improvements based on peer feedback. **Create a Maze (2 day lesson)** ● Design a custom maze backdrop in Scratch and program Scout to navigate through it using logic and movement. **Conditionals: Underwater Exploration**

	● Build an underwater scene that uses conditionals to control character behavior and responses. **Scout's Quest: Conditionals** ● Use if/then conditionals to build a program that responds to different inputs or scenarios.

Unit 5: Variables & Lists (6 lessons)

In this unit, students learn how to use variables and lists to store, update, and organize information in Scratch. They'll apply these concepts to build interactive games like Pong, Flying Bird, and a Spelling Bee, while reinforcing skills through Scout's Quest.

Objectives / Topics Covered	● Create and use variables to track information like scores. ● Use conditionals with variables to control game behavior. ● Create and manage lists to organize and display information in a program. ● Apply these concepts in interactive and skill-building Scratch projects.
Lessons	**Scout's Quest: Variables** ● Practice creating and updating variables to track points in a Scratch program. **Pong Game (2 day lesson)** ● Design a pong-style game using variables to keep score as the ball bounces between paddles. **Conditionals: Flying Bird (2 day lesson)** ● Use conditionals and variables to control gameplay and scoring in a Flying Bird game. **Lists: Spelling Bee** ● Build a spelling bee game that uses lists to store and display words players must spell.

Unit 6: Clones & Functions (7 lessons)

In this unit, students explore advanced Scratch concepts like cloning and functions with inputs. They'll use these tools to create dynamic projects—from animations and games to creative drawings—while developing efficient, reusable code structures and continuing skill review through Scout's Quest.

Objectives / Topics Covered	● Create and use clones to animate or duplicate sprites in a program. ● Build functions with Boolean and number inputs to control program behavior. ● Use variables and clones to design interactive games. ● Apply functions to automate and simplify complex drawings or repeated tasks.
Lessons	**Introduction to Clones** ● Create an animation using clones and explore how duplicate sprites can be used efficiently—and where they may have limitations. **Snake Game (2 day lesson)** ● Build a version of the classic Snake game using clones and variables to track movement and score. **Scout's Quest: Functions with Boolean Inputs** ● Create a function with a true/false input to determine actions, like checking a password. **Scout's Quest: Functions with Number Inputs** ● Use number inputs in functions to create a program that draws with specific values. **Flower Garden Functions Project (2 day lesson)** ● Use a function to draw multiple flowers in a scene, making the code cleaner and more efficient through reuse.

Unit 7: Culmination Projects (8 lessons)
In this final unit, students put their programming and problem-solving skills into action through creative, real-world projects. They'll apply conditionals, variables, operators, and data visualization to design interactive games, music tools, and visual displays, and explore 3D modeling through an accessibility-focused design challenge.

Objectives / Topics Covered	• Apply programming concepts like conditionals, events, variables, booleans, and operators in original projects. • Follow the inquiry process to collect, analyze, and display data through coding. • Design solutions with creativity and purpose using code and 3D modeling tools. • Demonstrate mastery of core computer science skills through interactive projects.choreography.
Lessons	**Click-a-Mole (2 day lesson)** • Design a Whack-a-Mole–style game using conditionals, variables, booleans, and events to create interactive gameplay. **Code Tunes (2 day lesson)** • Build a custom music player in Scratch using conditionals, operators, and variables to control features. **Inquiry Project: Data Bar Graph (2 day lesson)** • Conduct an investigation and modify a Scratch program to display survey results using a bar graph. **3D Design: Keyboard Accommodations (2 day lesson)** • Use a 3D design tool to create keyboard accommodations by aligning and positioning shapes with the align tool.

Utah Computer Science 4th Grade Course Supplemental Materials

Resources	**Description**
Parent Welcome Letter (Spanish)	Send this letter home to introduce families to computer science.
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	Empower your students to explore and express their knowledge creatively with

Scratch Templates	our versatile Scratch 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.
All of these resources and more are found on the **Elementary Resources Page**.	