CodeHS

Utah Computer Science 5th Grade Course Syllabus

One Year for Elementary School, 36 Hours

Course Overview and Goals

The **Utah Computer Science 5th 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/26173/overview

Course Breakdown

Optional Review

This optional review unit helps students refresh key computer science skills and prepare for the year ahead. Through Scratch-based activities, students will review vocabulary, practice programming on the coordinate plane, and customize their projects using drawing tools.

Objectives / Topics Covered	● Log in and navigate the Playground. ● Review core computer science vocabulary and Scratch basics. ● Practice using the coordinate plane for animation. ● Customize sprites and scenes with Scratch's drawing tools.
Lessons	**Welcome to CodeHS!** ● 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. **Scratch Drawing Tools** ● Use built-in Scratch drawing tools to create and personalize custom sprites and backgrounds.

Unit 1: Getting Started (4 lessons)

This introductory unit builds a strong foundation in how computers and networks work together to perform tasks. Students will explore the parts of computing systems, how data travels through networks using protocols and packets, and how to manage files efficiently. They'll also apply computational thinking to design creative solutions.

Objectives / Topics Covered	● Identify the parts of a computing system and how they function together. ● Apply basic troubleshooting strategies to solve common technology issues. ● Understand how data moves through networks using protocols and packets. ● Manage and organize digital files based on type and size. ● Use computational thinking to solve real-world design challenges.
Lessons	**Practicing Computing Systems** ● Identify key components of computing systems and apply troubleshooting strategies to resolve common technical issues. **Networks and Protocols** ● Learn how data is transferred across networks using protocols and packets, and model how this process works. **File Management and Data Exploration** ● Explore different file types, understand how they impact storage space, and practice organizing files effectively. **Computational Thinking: Design a Neighborhood** ● Use decomposition, pattern recognition, sequencing, and abstraction to design a neighborhood layout.

Unit 2: Sequences & Events (5 lessons)

In this unit, students strengthen their understanding of sequences, events, and algorithm design by creating animations and interactive programs. They'll practice debugging, refine algorithms for efficiency, and explore how computer science is used in real-world contexts like marine biology and professional sports.

Objectives / Topics Covered	● Use sequences and events to control sprite behavior in Scratch programs. ● Compare and improve algorithms based on clarity and efficiency. ● Apply debugging strategies by decomposing complex programs. ● Explore real-world uses of computer science, including storytelling through code. ● Use Scratch's drawing tools to create custom, animated characters.
Lessons	**Drawing Tools: Sea Creatures** ● Use all of Scratch's image editing tools to design and program animated sea creatures in a custom underwater scene. **Events: Traveling with Scout** ● Use different types of events to animate Scout's journey through various locations. **Compare and Refine Algorithms** ● Create multiple solutions to a problem, then compare and improve them to find the most efficient algorithm. **Debugging: Make a Pizza** ● Decompose a Scratch program step by step to find and fix errors in a pizza-making animation. **Careers in CS: Major League Baseball** ● Learn how coding is used in sports and retell important moments from an article using a timeline animation in Scratch.

Unit 3: Loops (2 lessons)

In this unit, students apply loops to create repeated designs using Scratch's pen tool, while also learning the importance of giving credit when using or remixing digital content. They'll combine creativity with responsible digital citizenship and efficient coding practices.

Objectives / Topics Covered	● Use loops to create repeating patterns and graphics with Scratch's pen tool. ● Understand and apply the concept of attribution when using or sharing digital work. ● Create original or remixed projects while practicing ethical digital behavior.
Lessons	**Creating Turtle Graphics** ● Use repeat loops and the pen tool in Scratch to draw looping patterns similar to turtle graphics. **Giving Credit Through Attributions** ● Learn how to appropriately credit original creators when remixing programs or using shared images online.

Unit 4: Conditionals & Operators (4 lessons)

In this unit, students explore how to use conditionals and operators to control program logic and add randomness. They'll build interactive projects like mazes, coin flips, and quests—practicing how to plan, decompose, and code using conditionals and logical expressions.

Objectives / Topics Covered	● Use if/then conditionals to control sprite behavior based on specific conditions. ● Apply operators and variables to introduce randomness and comparisons. ● Plan and break down complex projects into manageable steps. ● Create interactive programs that respond dynamically to input or data.
Lessons	**Conditionals: Mazes** ● Create a maze game in Scratch using conditionals to guide a sprite through the correct path. **Plan a Quest (2 day lesson)** ● Decompose the steps needed to build a quest-style game, planning out conditionals, events, and logic.

	Operators: Coin Flip ● Use variables and operators to create a coin-flipping program that simulates randomness and tracks outcomes.

Unit 5: Variables & Lists (3 lessons)

In this unit, students deepen their understanding of how variables and lists can be used to manage and manipulate information in Scratch programs. Through creative projects like dance animations and shopping simulations, they apply variables, lists, and operators to create more dynamic and interactive experiences.

Objectives / Topics Covered	● Use variables to store and control values like speed and pitch. ● Create and manage lists to store multiple related items. ● Combine variables, lists, and operators to power interactive simulations. ● Build projects that reflect real-world problem-solving using data structures.
Lessons	**Variables in Dance** ● Use variables to change dance speed and pitch, creating a dynamic and musical animation. **Lists: Shopping with Scout (2 day lesson)** ● Build a shopping simulator that uses lists to track items and variables and operators to calculate prices.

Unit 6: Clones & Functions (5 lessons)

In this unit, students explore how to use clones to efficiently duplicate sprites and actions in Scratch, particularly for game design. They'll also be introduced to the concepts of classes and objects in programming, using randomization to add variety and complexity to their projects.

Objectives / Topics Covered	● Use clones to duplicate sprites and streamline repetitive tasks in game development. ● Understand the value of clones for performance and efficiency in interactive programs. ● Explore foundational programming concepts like classes and objects. ● Use randomization to change object characteristics and enhance gameplay variety.
Lessons	**Clones: Throwing Acorns Game** ● Create a game where cloned acorns are thrown at targets, using events and movement to control gameplay. **Clones in Games (2 day lesson)** ● Build an endless runner game using clones and explain how clones help manage repeated elements efficiently. **Classes and Objects in Games (2 day lesson)** ● Learn the basics of classes and objects by creating a game where objects have random attributes like color, size, or speed.

Unit 7: Culmination Projects (13 lessons)

In this final unit, students apply their coding, design, and problem-solving skills to complete original, real-world projects. From building games and designing homes with functions to creating apps and visualizing data, these projects offer students opportunities to showcase their mastery through creativity, logic, and user-focused thinking.

Objectives /	● Apply programming concepts like loops, conditionals, variables, and functions in

Topics Covered	original designs. ● Integrate math concepts like area and perimeter into a creative computer science project. ● Use the design thinking process to create a user-centered app. ● Follow the inquiry process to collect, analyze, and display data using graphs.
Lessons	**Game Design Project (3 day lesson)** ● Design and build an original Scratch game using skills such as loops, conditionals, and variables. **House Design with Area and Perimeter (2 day lesson)** ● Use programming functions to design a digital house layout, applying area and perimeter calculations in a CS context. **Design an App (3 day lesson)** ● Follow the design thinking process to develop a prototype for an app that solves a real-world problem. **Pinball Game Project (3 day lesson)** ● Build a playable pinball game by applying game design principles such as physics, scoring, clones, and interaction. **Inquiry Project: Line Graph (2 day lesson)** ● Conduct an investigation and modify a Scratch program to display the results as a line graph.

Utah Computer Science 5th 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 Scratch Templates	Empower your students to explore and express their knowledge creatively with 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**.