CodeHop

Georgia Computer Science 4th Grade Course Syllabus

One Year for Elementary School, 36 Hours

Course Overview and Goals

The **Georgia 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. Digital literacy lessons are also available to complement the programming curriculum with non-programming skills. Additionally, this course includes optional interdisciplinary lessons to support cross-curricular integration.

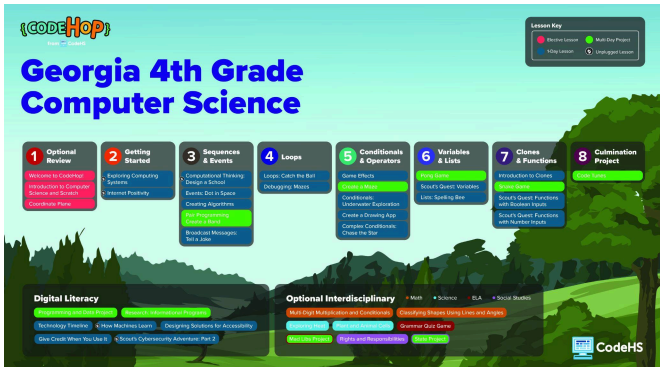
Learning Environment: This course is designed to be teacher-led, with ready-to-use lesson plans that follow a structured format: **Introduction, Guided Practice, Independent Practice, Extension, and Reflection**. Lessons are built with spiral review to reinforce key concepts and culminate in engaging 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. The concepts taught in this course spiral across grade levels, ensuring that students can build upon their understanding year after year. The course includes approximately **36 instructional hours**, providing a full school year of material if teaching one lesson per week. Depending on your pacing, you may find opportunities to adjust 1–2 lessons to best fit your calendar. Lesson prep notes on each lesson page provide ideas for streamlining activities while maintaining key learning goals.

Programming Environment: Students will write and run programs in **Scratch** embedded and saved in the CodeHop 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/26341/overview?lang=en.



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

Course Breakdown

Unit 1: Optional Review (2 weeks)

In this optional unit, students refresh foundational skills in computer science and Scratch. They revisit key vocabulary, explore the CodeHop platform, and use the coordinate plane to animate with precision.

Objectives / Topics Covered	● Log in and navigate the CodeHop Playground. ● Define basic computer science terms and concepts. ● Create simple Scratch programs using motion and coordinates.
Lessons	**Welcome to CodeHop!** ● Learn how to log in and use the CodeHop Playground. This short introductory lesson can be used on its own, or right before a full lesson. **Introduction to Computer Science and Scratch** ● Define important computer science vocabulary and create a simple program in Scratch. **The Coordinate Plane** ● Create an open-ended animation using the coordinate plane in Scratch.

Unit 2: Getting Started (2 weeks)

In this unit, students explore computing systems and practice digital citizenship. They learn about hardware, software, and the impact of online behavior.

Objectives / Topics Covered	● Identify parts of a computing system and solve simple tech problems. ● Recognize and promote positive digital behavior. ● Create a digital code of conduct for online responsibility.
Lessons	**Exploring Computing Systems** ● Identify parts of the computing system and identify simple hardware and software problems. **Internet Positivity** ● Explain how their actions can spread positivity on the internet and create a code of conduct for responsible online behavior.

Unit 3: Sequences & Events (6 weeks)

Students apply computational thinking and build interactive programs using sequences, multiple event types, and broadcast messages. They also practice collaboration through pair programming.

Objectives / Topics Covered	● Use sequences and events to control sprite behavior. ● Program and compare multiple algorithms to solve tasks. ● Collaborate using pair programming techniques. ● Use broadcasts to trigger interaction between sprites.
Lessons	**Computational Thinking: Design a School** ● Use computational thinking to design a school. **Events: Dot in Space** ● Create a program using multiple types of event blocks. **Creating Algorithms** ● Program multiple algorithms and assess which one best meets their needs. **Pair Programming: Create a Band (2-part lesson)** ● Collaborate through pair programming to design and code a band in Scratch using keyboard inputs. **Broadcast Messages: Tell a Joke** ● Use broadcast messages to program two sprites to tell a knock knock joke.

Unit 4: Loops (2 weeks)

In this unit, students explore different types of loops and apply them to build efficient, interactive programs. They also practice debugging skills by analyzing and fixing broken programs.

Objectives / Topics Covered	● Use repeat and forever loops to simplify repetitive actions. ● Apply loops in game design and animation. ● Debug programs by decomposing their components.
Lessons	**Loops: Catch the Ball** ● Use two types of loops to create a simple game in Scratch. **Debugging: Mazes** ● Decompose a program to debug and make the program run as intended.

Unit 5: Conditionals & Operators (6 weeks)

Students build dynamic programs using conditional statements, operators, and input events. They explore how to make decisions in code and apply logic to animations, drawing apps, and interactive games.

Objectives / Topics Covered	● Use conditionals and operators to control program flow. ● Combine inputs, loops, and conditionals for interactive designs. ● Apply logic to drawing apps, animations, and user-driven stories. ● Debug and refine programs based on peer feedback.
Lessons	**Game Effects** ● Modify a game to add engaging effects and make updates to their game based on peer feedback. **Create a Maze (2-part lesson)** ● Draw a maze backdrop in Scratch and program Scout to navigate through the maze. **Conditionals: Underwater Exploration** ● Create a program that uses conditionals. **Create a Drawing App** ● Create a drawing app by programming keyboard and mouse inputs, loops, and conditional statements. **Complex Conditionals: If/Then/Else Chase the Star** ● Explain what an “if/then/else” conditional is and use it in a program.

Unit 6: Variables & Lists (4 weeks)

This unit introduces students to data storage and tracking in programs. They create games and interactive projects that use variables and lists to store and update information dynamically.

Objectives / Topics Covered	● Create and use variables to store, update, and display information. ● Use lists to store multiple values in a program. ● Apply variables and lists in interactive games and point-tracking systems.
Lessons	**Pong Game (2-part lesson)** ● Create and use variables to keep score in an interactive pong game. **Scout's Quest: Variables** ● Create and use variables to track points in a program. **Lists: Spelling Bee** ● Use lists to create a spelling bee game.

Unit 7: Clones & Functions (5 weeks)

In this unit, students learn to reuse code and manage repeated actions using clones and functions. They build more complex projects like games and animations by organizing code with reusable blocks and inputs.

Objectives	● Create and use clones to manage repeated sprite actions.

/ Topics Covered	• Build games using clones and variables. • Use functions with boolean and number inputs to customize behaviors. • Structure code for reuse, clarity, and flexibility.
Lessons	**Introduction to Clones** • Create an animation using clones and investigate the limitations of their program. **Snake Game (2-part lesson)** • Use variables and clones to create a snake game. **Scout's Quest: Functions with Boolean Inputs** • Create a function including a boolean input to perform different actions based on whether a password is correct. **Scout's Quest: Functions with Number Inputs** • Create a drawing using functions with number inputs.

Unit 8: Culmination Projects (2 weeks)

Students apply key programming concepts in a creative final project that integrates conditionals, variables, operators, and user interaction. They build a custom music player to showcase their skills.

Objectives / Topics Covered	• Combine conditionals, variables, and operators in an original project. • Design interactive programs with user input and logic. • Apply multiple computer science skills in a creative, real-world context.
Lessons	**Code Tunes (2-part lesson)** • Use variables, operators, and conditionals to create their own custom music player in Scratch.

Unit 9: Digital Literacy (10 weeks)

In this unit, students explore data analysis, machine learning, cybersecurity, and digital responsibility. They conduct research, present findings, and learn how technology impacts society and personal safety.

Objectives / Topics Covered	• Apply math and science concepts through interactive programming. • Use lists and conditionals to build academic games and simulations. • Model real-world and civic concepts using events, variables, and logic. • Discuss how computing technologies have changed the world.
Lessons	**Programming and Data Project** • Develop an investigative question, collect data, draw conclusions based on the data, and create an interactive program to present data visually. **Research: Informational Programs** • Examine information from different resources and communicate main ideas by creating a Public Service Announcement (PSA) on healthy sleep habits. **Impacts of Computing: Exploration** • *This lesson is coming soon!* **How Machines Learn** • Explain the different machine learning approaches and create a classification system using a tree structure. **Designing Solutions for Accessibility (2-part lesson)** • Use the design thinking process to identify and solve real-world problems by redesigning a game to improve accessibility and usability for diverse users. **Give Credit When You Use It** • Search for information to answer questions online and provide proper attribution to sources. **Scout's Cybersecurity Adventure: Part 2** • Demonstrate how to stay safe online by practicing secure habits and understanding the tools and technologies that protect their information.

Unit 10: Optional Interdisciplinary (10 weeks)

In this cross-curricular unit, students integrate computer science with math, science, ELA, and social studies. They use loops, lists, conditionals, and variables to create interactive projects that reinforce academic content and promote deeper conceptual understanding.

Objectives / Topics Covered	• Combine conditionals, variables, and operators in an original project. • Design interactive programs with user input and logic. • Apply multiple computer science skills in a creative, real-world context.
Lessons	**Multi-digit Multiplication and Conditionals** • Use if/then conditionals to review multiplication with multi-digit factors. **Classifying Shapes Using Lines and Angles** • Create a program to categorize shapes based on the properties of their lines and angles. Use comments to document their code. **Exploring Heat** • Use events in their program to communicate information about how heat energy from the sun affects objects on earth. **Plant and Animal Cells** • Use broadcast events to create an interactive program about plant and animal cells. **Grammar Quiz Game** • Use conditionals to create a quiz that tests the user's understanding of standard English grammar usage. **Mad Libs Project (2-part lesson)** • Use lists in a program to create a Mad Libs game. **Rights and Responsibilities** • Use variables and events to create a voting program to demonstrate the rights and responsibilities of citizens. **State Project (2-part lesson)** • Use events to create an interactive project detailing state-specific facts.

4th Grade Course Supplemental Materials

Resources	**Description**
Parent Welcome Letter (Spanish)	Send this letter home to introduce families to computer science with CodeHop.
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.
These resources and more are found on the **Elementary Resources Page**.	