Virginia Computer Science
4th Grade Course Syllabus
One Year for Elementary School, 36 Hours

Course Overview and Goals

The **Virginia 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 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/27183/overview

A clickable PDF can be found at https://codehs.com/VA-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)

This unit introduces students to the basics of computing systems, digital data, and computational thinking. They learn how computers work, how data is stored, and how to approach problem-solving with computational thinking strategies.

Objectives / Topics Covered	• Identify parts of a computing system and troubleshoot simple hardware/software problems. • Understand how different types of digital data take up space and can be stored in multiple ways. • Apply computational thinking to design and problem-solve.
Lessons	**Exploring Computing Systems** • Identify parts of a computing system and resolve simple hardware/software problems. **File Management and Data in Action** • Explain how different digital files vary in size and storage methods. **Computational Thinking: Design a School** • Use computational thinking to design and plan the layout of a school.

Unit 2: Sequences, Events & Loops (2 lessons)

In this unit, students expand their understanding of core programming structures by learning to use sequences, events, and loops. They practice designing algorithms, controlling program flow with events, and debugging errors. Students also explore real-world applications of coding, such as sports technology, to see how computer science connects to everyday life.

Objectives / Topics Covered	• Use different event blocks to control sprite behavior. • Design and compare algorithms to determine efficiency. • Apply loops to create repeated actions in interactive programs. • Debug sequences, events, and loops to fix errors. • Connect coding to real-world applications, including sports and data visualization.
Lessons	**Events: Dot in Space**

	● Create a program using different types of event blocks to control actions. **Creating Algorithms** ● Create a program using different types of event blocks to control actions. **Careers in CS: Major League Baseball** ● Explore how coding is used in sports by retelling events from an article in a timeline program. **Loops: Catch the Ball** ● Use repeat and forever loops to build a simple ball-catching game in Scratch. **Debugging: Mazes** ● Decompose and fix a maze program to make it run as expected.

Unit 3: Conditionals & Operators (5 lessons)

Students explore conditional logic and operators to create smarter, more responsive programs. They apply these skills to animations and games with custom effects.

Objectives / Topics Covered	● Use if/then conditionals and logical operators in Scratch. ● Build maze games and interactive experiences. ● Revise programs based on peer feedback.
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 4: Variables & Lists (5 lessons)

In this unit, students are introduced to variables and lists, two essential tools for storing and managing data in programs. They apply these concepts to create interactive games and simulations, while also connecting coding to science concepts such as forces and motion. By working with variables and lists, students learn how to track scores, manage information, and design more complex, data-driven projects.

Objectives / Topics Covered	● Create and modify variables to track game data. ● Use lists to manage multiple data points like spelling words. ● Apply variable and list logic in Scratch games.
Lessons	**Scout's Quest: Variables** ● Practice creating and updating variables to track points in a Scratch program. **Balanced and Unbalanced Forces** ● Model how forces affect motion using conditionals and variables, and explain how the simulation connects to science concepts. **Pong Game (2 day lesson)** ● Create a pong game that uses variables to keep score. **Lists: Spelling Bee** ● Build a spelling bee game that uses a list of words for interactive play.

Unit 5: Clones & Functions (4 lessons)

Students explore advanced programming tools—clones and functions with inputs—to create more dynamic and efficient Scratch projects.

Objectives / Topics Covered	● Use clones to animate repeated or duplicated elements. ● Create functions with boolean or number inputs. ● Apply functions in creative and logic-based animations.
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.

Unit 6: Culmination Projects (8 lessons)

In this unit, students apply their programming knowledge and design skills to create meaningful, original projects. They'll solve real-world problems, personalize interactive tools, and present data-driven findings.

Objectives / Topics Covered	● Apply the design thinking process to solve real-world problems. ● Create interactive cultural, musical, and data-driven experiences. ● Use conditionals, variables, lists, and operators in original projects.
Lessons	**Designing Solutions for Accessibility (2 day lesson)** ● Use design thinking to build a program that improves access or inclusion. **Choose Your Own Path: Elements of Culture (2 day lesson)** ● Design a cultural choose-your-own-path interactive game. **Code Tunes (2 day lesson)** ● Create a custom music player using variables, conditionals, and operators. **Inquiry Project: Data Bar Graph (2 day lesson)** ● Follow the inquiry process to collect data and display it in a bar graph using Scratch.

Unit 7: Digital Literacy (7 lessons)

In this unit, students build digital literacy skills that prepare them to use technology safely, responsibly, and thoughtfully. Lessons highlight how to research and give credit, explore cybersecurity, understand networks, and analyze the cultural impacts of computing. Students also compare human and machine abilities to better understand the role of technology in society.

Objectives / Topics Covered	● Search for information online and provide proper attribution. ● Practice safe online habits and cybersecurity strategies. ● Explain how the Internet works, including packets and secure communication. ● Explore the cultural impacts of technology, identifying both benefits and challenges. ● Compare human and machine performance to understand strengths and limitations of computing.
Lessons	**Give Credit When You Use It** ● Conduct research and cite online sources appropriately. **Data Investigators**

	• Evaluate and analyze data for reliability to draw conclusions and make predictions. **Scout's Cybersecurity Adventure: Part 2** • Practice safe online habits and understand digital protection tools. **Networks, Packets, and the Internet** • Model how data travels as packets across a network and is reassembled securely. **Impacts of Computing: Exploration** • Explore how computing has changed communities and evolved over time. **Give Credit When You Use It** • Conduct research and cite online sources appropriately. **Humans Vs. Machines** • Compare human and computer performance on tasks, explain advantages and limitations, and describe how computers perceive information.

Virginia 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 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 **CodeHop Resources Page**.	