3rd Grade Computer Science Course Syllabus

One Year for Elementary School, 36 Hours

Course Overview and Goals

The **3rd Grade Computer Science 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.

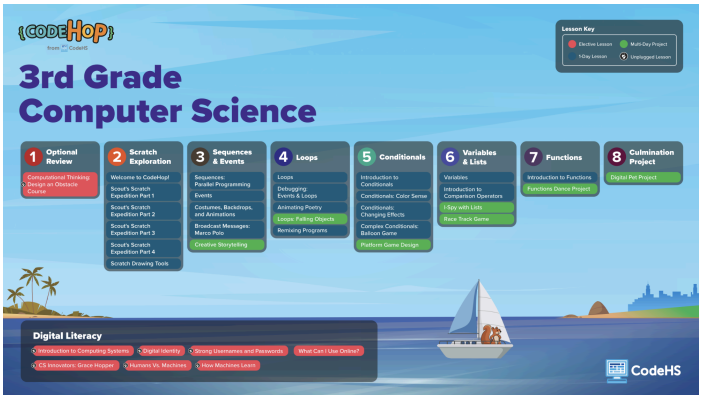
Learning Environment: This course is designed to be teacher-led, with ready-to-use lesson plans that follow a structured format: **Introduction, Guided Practice, Independent Practice, Extension, and Reflection**. Lessons are built with spiral review to reinforce key concepts and culminate in engaging 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**, each 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 students' accounts. 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/20685/overview?lang=en



A clickable PDF can be found at https://codehs.com/K5CSRoadmaps.

Course Breakdown

Unit 1: Optional Review (1 week)

In this optional unit, students apply computational thinking strategies to plan and design an obstacle course, using logic and sequencing to break down and solve design challenges.

Objectives / Topics Covered	● Apply decomposition, pattern recognition, and algorithmic thinking. ● Plan a step-by-step solution for a design-based problem. ● Translate real-world ideas into structured, logical sequences.
Lessons	**Computational Thinking: Design an Obstacle Course** ● Use computational thinking skills to design an obstacle course by breaking the challenge into smaller steps, recognizing patterns, and organizing a logical plan.

Unit 2: Scratch Exploration (5 weeks)

Students begin programming in Scratch through a guided, story-driven adventure with Scout, gaining hands-on experience with basic commands, animation, loops, events, and creative design using Scratch's drawing tools.

Objectives / Topics Covered	● Log in and explore the CodeHop Playground to get started with Scratch. ● Use basic Scratch blocks to program motion, speech, and sequences. ● Build multi-step stories using loops, events, looks, and motion. ● Customize characters and scenes using Scratch's drawing tools.
Lessons	**Welcome to CodeHop! (15 minute lesson)** ● Log in and explore the CodeHop Playground as a short introduction before beginning a full project. **Scout's Scratch Expedition Part 1** ● Use basic Scratch commands to program a sprite to move and talk as part of the Scout story. **Scout's Scratch Expedition Part 2** ● Follow the next part of the story by adding sprites and creating a sequence to animate a scene. **Scout's Scratch Expedition Part 3** ● Create an animated story using loops, events, looks, and motion blocks to bring the narrative to life. **Scout's Scratch Expedition Part 4** ● Continue building the Scout story using events, looks, and motion blocks in a fully animated scene. **Scratch Drawing Tools** ● Use Scratch's built-in drawing tools to design custom sprites and backdrops for creative projects.

Unit 3: Sequences and Events (6 weeks)

In this unit, students practice with sequences and events in Scratch by programming animations, coordinating sprite interactions, and using parallel and broadcast actions to enhance storytelling and creativity.

Objectives / Topics Covered	● Create sequences that run at the same time using parallel programming. ● Use event and broadcast blocks to trigger specific actions between sprites. ● Animate scenes with costumes, backdrops, and timed interactions. ● Plan and code creative stories using sequences and event-based logic.

Lessons	**Sequences: Parallel Programming** ● Build a program where multiple sequences run at the same time, allowing sprites to act simultaneously. **Events** ● Create a program that uses event blocks to trigger actions when specific keys are pressed or sprites are clicked. **Costumes, Backdrops, and Animations** ● Animate sprites and scenes using costume changes and backdrop transitions. **Broadcast Messages: Marco Polo** ● Use broadcast messages to coordinate actions between sprites in a call-and-response style program. **Creative Storytelling (2-part lesson)** ● Plan and animate an original story using a combination of sequences, events, and sprite interactions.

Unit 4: Loops (6 weeks)

Students explore how loops can simplify repetition in code, apply debugging strategies, and use looping logic in creative projects like poetry animations and falling-object simulations, while practicing ethical remixing.

Objectives / Topics Covered	● Understand and apply different types of loops to repeat instructions in Scratch. ● Debug programs that combine events and loops to ensure proper function. ● Use loops creatively in storytelling, animation, and simulations. ● Remix existing programs while giving proper credit to original creators.
Lessons	**Loops** ● Explain that loops repeat instructions and create a Scratch program that uses loops to simplify repeated actions. **Debugging: Events and Loops** ● Decompose a program that uses events and loops to identify and correct errors. **Animating Poetry** ● Create an animated version of a poem using loops and events to add rhythm and visual effects, while focusing on core CS concepts. **Loops: Falling Objects (2-part lesson)** ● Program falling objects using different types of loops and compare the pros and cons of each loop approach. **Remixing Programs** ● Practice remixing a digital Scratch program and provide proper attribution to the original creator.

Unit 5: Conditionals (6 weeks)

In this unit, students explore how conditionals are used to make decisions in programs, progressing from simple if/then statements to complex game logic with if/then/else and live condition checks.

Objectives / Topics Covered	● Understand and apply if/then and if/then/else conditionals in Scratch. ● Use conditionals to control visual effects, sprite behavior, and interactions. ● Build games that react to player actions and program conditions. ● Combine conditionals with loops and events to design responsive experiences.
Lessons	**Introduction to Conditionals** ● Explain what conditionals are and create a program that uses if/then blocks to respond to specific situations. **Conditionals: Color Sense** ● Use if/then conditionals to program responses based on sprite color or color detection. **Conditionals: Changing Effects** ● Create a program that uses conditionals to change sprite appearance, actions, or scene behavior based on specific triggers.

	Complex Conditionals: Balloon Game ● Use if/then/else blocks to build a more advanced game that changes behavior based on different player interactions. **Platform Game Design (2-part lesson)** ● Design a platform-style game using keyboard events, forever loops, and conditionals to detect and respond to game conditions; debug and improve the program for better gameplay.

Unit 6: Variables and Lists (6 weeks)

Students build on their understanding of data in programming by using variables, lists, and comparison operators to create interactive games and activities that track, compare, and respond to changing values.

Objectives / Topics Covered	● Define and use variables to store and update values in a program. ● Use comparison operators with variables and numbers to guide conditional logic. ● Create and manipulate lists to track multiple items in interactive activities. ● Combine variables, lists, and conditionals in personalized game design.
Lessons	**Variables** ● Learn what a variable is, and create a program that stores, changes, and displays variable values. **Introduction to Comparison Operators** ● Use comparison operators in if/else statements to compare values and control program flow using variables and numbers. **I-Spy with Lists (2-part lesson)** ● Build an I-Spy style game that uses variables and lists to track which items have been found. **Race Track Game (2-part lesson)** ● Design a custom race track and car, then program a racing game that uses conditionals, variables, and lists to keep score and control game logic.

Unit 7: Functions (4 weeks)

In this unit, students are introduced to functions in Scratch and learn how to create reusable code blocks, applying them to structured sequences like dance routines that align with music.

Objectives / Topics Covered	● Understand the concept of functions and how they simplify code. ● Create and call functions (custom blocks) to organize repeated actions. ● Apply functions in creative projects to control timing and movement in sequences.
Lessons	**Introduction to Functions** ● Create a function in Scratch and use it within a program to reduce repetition and organize actions. **Functions Dance Project (3-part lesson)** ● Use custom functions to program dance moves that align with music, demonstrating the efficiency and creativity enabled by reusable code blocks.

Unit 8: Culmination Projects (3 weeks)

In this culminating project, students apply their programming knowledge to create an interactive digital pet that responds to inputs using events, conditionals, variables, comparison operators, and broadcasts.

Objectives / Topics Covered	● Use events, conditionals, variables, comparison operators, and broadcasts. ● Create interactive behaviors that simulate caring for a digital pet. ● Combine multiple coding elements into a cohesive, creative project.
Lessons	**Digital Pet Project (3-part lesson)**

	• Design a Scratch program for a digital pet using events, conditionals, and variables, with enhanced logic through comparison operators and broadcasts.

Unit 9: Digital Literacy (7 weeks)

In this optional unit, students build digital literacy by learning how computing systems work, how to protect their digital identity, and how to interact safely and responsibly with technology and online information.

Objectives / Topics Covered	• Identify parts of a computing system and troubleshoot basic issues. • Understand digital identity, privacy, and safe password creation. • Practice responsible internet use and proper attribution of online sources. • Explore computer science pioneers, machine learning, and how computers compare to humans.
Lessons	**Introduction to Computing Systems** • Identify hardware and software components and learn how to solve basic computing problems. **Digital Identity** • Connect real-world and online identities and learn how to build a positive digital footprint. **Strong Usernames and Passwords** • Create secure usernames and passwords and explain why strong passwords help keep personal information safe. **What Can I Use Online?** • Search for information online to answer questions and give proper credit to original sources. **CS Innovators: Grace Hopper** • Learn about Grace Hopper's impact on computer science and explore binary code through decoding activities. **Humans Vs. Machines** • Compare human and computer abilities, explain the strengths and limitations of technology, and learn how computers perceive the world. **How Machines Learn** • Understand different types of machine learning and create a decision tree to classify and sort information like AI would.

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 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 **CodeHop Resources Page**.	