Indiana 3rd Grade Computer Science Course Syllabus

One Year for Elementary School, 36 Hours

Course Overview and Goals

The **Indiana 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.

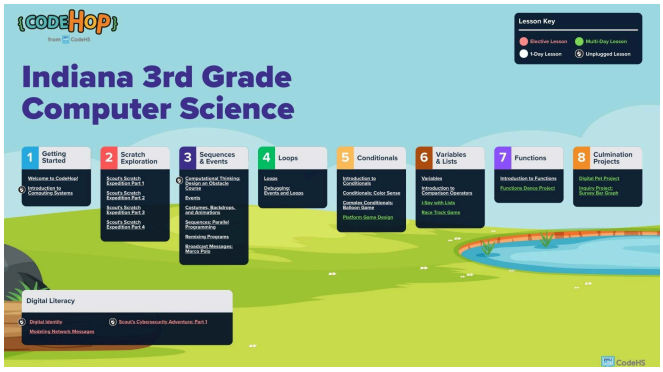
Learning Environment: This course is designed to be teacher-led, with ready-to-use lesson plans that follow a structured format: **Introduction, Guided Practice, Independent Practice, Extension, and Reflection**. Lessons are built with spiral review to reinforce key concepts and culminate in engaging 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. 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/22926/overview?lang=en



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

Course Breakdown

Unit 1: Getting Started (1 week)

In this unit, students get introduced to the Scratch interface and begin exploring the components of computing systems, including how to solve basic hardware and software problems.

Objectives / Topics Covered	● Log in and navigate the CodeHop Playground. ● Identify computing system components and distinguish between hardware and software.
Lessons	**Welcome to CodeHop! (Optional)** ● Log in and explore how to use the CodeHop Playground as an introduction or warm-up activity. **Introduction to Computing Systems** ● Identify the parts of a computing system and solve basic hardware and software problems.

Unit 2: Scratch Exploration (4 weeks)

In this unit, students follow a sequential, story-driven Scratch adventure to learn core programming skills including motion, looks, events, loops, and sprite animation.

Objectives / Topics Covered	● Use basic Scratch blocks to animate sprites and create interactive stories. ● Add sprites, apply motion and looks blocks, and build sequences. ● Incorporate loops and events to structure animations and story logic.
Lessons	**Scout's Scratch Expedition Part 1** ● Use basic Scratch commands to program a sprite to move and talk as part of a story. **Scout's Scratch Expedition Part 2** ● Add sprites and create a sequence of code to animate a story scene. **Scout's Scratch Expedition Part 3** ● Build an animated story using loops, events, looks, and motion blocks in Scratch. **Scout's Scratch Expedition Part 4** ● Continue developing the animated story using events, looks, and motion blocks.

Unit 3: Sequences and Events (6 weeks)

In this unit, students deepen their understanding of sequences and events by designing obstacle courses, animating scenes, using parallel programming, remixing projects, and coordinating sprites through broadcast messages.

Objectives / Topics Covered	● Apply computational thinking to design and problem-solve. ● Create interactive programs using events and parallel sequences. ● Animate sprites and backdrops. ● Remix programs and coordinate sprite actions with broadcasts.
Lessons	**Computational Thinking: Design an Obstacle Course** ● Use computational thinking to plan and design an obstacle course challenge. **Events** ● Create a Scratch program that uses events to trigger actions. **Costumes, Backdrops, and Animations** ● Animate sprites and scenes using costumes and interactive backdrops. **Sequences: Parallel Programming** ● Build a program where multiple sequences run at the same time.

	Remixing Programs ● Remix an existing program and provide attribution to the original creator. **Broadcast Messages: Marco Polo** ● Use broadcast messages to trigger communication between sprites.

Unit 4: Loops (2 weeks)

In this unit, students explore how loops repeat instructions in Scratch and practice debugging programs that use events and loops.

Objectives / Topics Covered	● Use loops to repeat one or more instructions in a Scratch program. ● Decompose and debug programs that combine loops and events.
Lessons	**Loops** ● Explain how loops repeat instructions and use them to simplify code in a Scratch project. **Debugging: Events and Loops** ● Decompose and fix a program that uses both events and loops to ensure it runs as intended.

Unit 5: Conditionals (5 weeks)

In this unit, students learn to use conditionals to make decisions within programs, progressing from basic if/then logic to more complex game mechanics using if/then/else statements and continuous condition checks.

Objectives / Topics Covered	● Explain and use basic conditionals to control program behavior. ● Implement if/then/else logic to create dynamic interactions. ● Design and debug a platform-style game using conditionals, loops, and keyboard events.
Lessons	**Introduction to Conditionals** ● Explain what a conditional is in programming and create a Scratch project using if/then blocks. **Conditionals: Color Sense** ● Use conditionals in a Scratch program to respond to color-based triggers. **Complex Conditionals: Balloon Game** ● Introduce if/then/else conditionals to create more advanced logic in a game scenario. **Platform Game Design (2-part lesson)** ● Build a platform game using keyboard controls, loops, and conditionals; debug and improve the program based on how it responds to ongoing conditions.

Unit 6: Variables and Lists (6 weeks)

In this unit, students explore variables and lists to store and compare data, enabling them to build interactive games and activities with scoring, tracking, and dynamic logic.

Objectives / Topics Covered	● Define and use variables to store and update data. ● Use comparison operators with conditionals to control program flow. ● Create and manipulate lists in interactive Scratch projects. ● Apply variables, lists, and conditionals in custom game design.
Lessons	**Variables** ● Explain what a variable is and create a program that changes and displays variable values. **Introduction to Comparison Operators** ● Use comparison operators in if/else blocks with numbers and variables to guide

	program behavior. **I-Spy with Lists (2-part lesson)** ● Create an I-Spy style game using lists and variables to track selected items. **Race Track Game (2-part lesson)** ● Design a race car and track, then build a scoring game using conditionals, variables, and lists.

Unit 7: Functions (4 weeks)

In this unit, students are introduced to functions in Scratch, learning how to define and reuse custom blocks to organize and simplify their code.

Objectives / Topics Covered	● Create and use functions (custom blocks) to structure programs. ● Apply functions in creative projects to repeat actions and improve code clarity.
Lessons	**Introduction to Functions** ● Create and use functions in Scratch to simplify and organize repeated code. **Functions Dance Project (3-part lesson)** ● Build a dance animation that uses functions to call dance moves in sync with music.

Unit 8: Culmination Projects (5 weeks)

In this unit, students apply their accumulated knowledge of programming concepts to design original projects that involve interactivity, data collection, and creative expression.

Objectives / Topics Covered	● Combine events, conditionals, variables, and broadcasts in a creative program. ● Follow the inquiry process to collect data and display results using code. ● Demonstrate mastery of core Scratch programming skills through original projects.
Lessons	**Digital Pet Project (3-part lesson)** ● Create an interactive digital pet using events, conditionals, variables, comparison operators, and broadcasts. **Inquiry Project: Survey Bar Graph (2-part lesson)** ● Conduct an investigation, then modify a program to display survey results using a bar graph.

Unit 9: Digital Literacy (3 weeks)

In this unit, students build digital literacy by exploring digital identity, understanding how data moves through networks, and learning foundational cybersecurity concepts to stay safe online.

Objectives / Topics Covered	● Understand the relationship between real-world and online identity. ● Describe how networks connect devices and transmit data. ● Identify common cybersecurity threats and safety strategies.
Lessons	**Digital Identity** ● Connect real and online identities and identify actions that contribute to a positive digital footprint. **Modeling Network Connections** ● Describe how data moves between devices over a network and create a program to model that movement. **Scout's Cybersecurity Adventure: Part 1** ● Learn basic cybersecurity concepts, recognize common cyber threats, and explore tips for online safety.

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 **Elementary Resources Page**.	