

Morning Meeting SEL/ Computer Science Lessons

By: Wendy Weaver

Total Number of Lessons: 18

Total Estimated Time for Module: 10-15 minutes

Section 1: Module Overview and Curricular Connections – Required

- **Module goal:** SWBAT work with peers to create and solve problems through conversation and planning in a respectful way.
- **Connection to standards:**
 - **3-5.DI.3:** Demonstrate how variables can represent data and are used to store and modify information.
 - **3-5.DI.4:** Describe how data can be represented in different forms understandable by people, including words, symbols, and digital displays of color
 - **3-5.PA.1:** Collaborate with peers to implement problem-solving steps to create a variety of programming solutions. (E)
 - **3-5.PA.2:** Design programs that incorporate sequences, events, loops, and conditionals.
 - **3-5.PA.3:** Test and debug (i.e., identify and fix errors) a program or algorithm to ensure it runs as intended.
 - **3-5.IC.2:** Seek diverse perspectives for the purpose of improving computational artifacts.
- **RET materials/ideas to be leveraged:** I learned many things this summer in the RET program. Many of our topics included AI usage. Understanding what AI is, how it works, and how to use it in the classroom as a teacher has been helpful. Specifically, I learned how to prompt AI to create and answer things like lesson plans (one linked below), answer questions like grade-level text, and determine whether what I am reading is reliable information. As a fourth-grade teacher, I use AI less than a high schooler would. However, the use of technology and AI is growing, and students must know how to leverage AI effectively. I want to help students understand that AI is now a book of absolute truth, but I can also assist with many things, as I listed above. My lesson plan, hopefully, helps students problem-solve and see the value of collaboration and peer input. Each lesson is specifically designed to create an environment where students can use their knowledge of technology while practicing skills that make them a good citizen of their community through character development.

Section 2: Overview of Module Framework:

Real-world context:

Students will understand the importance of problem-solving in a variety of activities. Indiana's Computer Science standards for grades 3–5 ask students to understand and apply the basic steps of algorithmic problem solving — defining a problem, examining sample instances, designing a solution, implementing it, and testing it (3-5.DI.1) while also connecting computer science to other fields and to daily life (3-5.DI.5).

For fourth graders, this becomes a real-world experience when the "problem" is something they actually encounter, not an abstract puzzle: reorganizing the classroom library, speeding up the lunch line, or writing clear step by step directions for a familiar task like making a sandwich. Students move through the same cycle a computer scientist would — stating the problem, looking at how similar problems have been solved, designing a step-by-step solution (an algorithm), trying it out, and revising it when it doesn't work as expected. Framing the lesson around a genuine classroom or community problem, rather than a canned coding exercise, allows students to see computational and algorithmic thinking as a transferable problem-solving process rather than a subject confined to a screen.

Possible Lesson Ideas:

- **"Find the Bug in Our School"**

SEL Trait: Courage · CS Standard: 3-5.DI.1 (problem statement and exploration)

- Frame it as: "Every good program starts with someone noticing a problem worth solving. Today we're going to find real bugs in our school — not code bugs, life bugs."
- In groups, students brainstorm things in the classroom/school that don't work well (recess line takes too long, library books are hard to find, the classroom job chart is confusing). Each group writes their "bug report" on the whiteboard: what's the problem, who does it affect, how do you know it's a problem (evidence, not just opinion).
- Gallery walk, then a whole-class vote narrows it to one or two problems the class will actually try to solve over the next few weeks. This directly hits the "problem statement and exploration" step of the standard and gives real ownership — the debrief question could be "why does it take courage to point out a problem instead of just living with it?"

- **"Pitch the Plan"**

- *SEL Trait: Perseverance or Ownership · CS Standard: 3-5.PA.1 (collaborate to implement problem-solving steps)*
- This is the natural follow-up once a problem is chosen. Groups sketch a first-draft "algorithm" for solving it — numbered steps, just like Lesson 03's sandwich algorithm, but applied to the real problem (e.g., steps for a new recess line-up system).

- They pitch their draft to another group, who plays the "robot test" role from Lesson 03: follow the steps literally and point out where it breaks down. Groups revise based on that feedback. Debrief centers on perseverance — first drafts of real solutions almost never work, and that's expected, not a failure.

●

Background STEM content:

- **Possible Lesson Ideas:** Many of these lessons have students notice and interact with patterns and “if/then” thinking. There are also lessons where students must discover the problem in patterns and flowcharts. What makes these lessons appropriate for fourth grade is that they focus on SEL traits through a computer science-based activity.
- **"Engineers Find Real Problems"**
 - *Focus: Ask/Define phase of the engineering design process · CS tie-in: 3-5.DI.1 (problem statement and exploration)*
 - Open with: "Engineers don't start with a solution — they start by noticing a problem worth solving." In small groups, students survey the classroom, school, or their own routines for a genuine design problem: the recycling bins overflow before Friday, the plant corner doesn't get enough light, the pencil sharpener line eats up class time. Each group documents their problem on a simple engineering notebook page: what's wrong, who's affected, and what evidence they have (this could even involve a quick tally or measurement — a nice STEM/math tie-in).
 - Groups present their problem statement to the class, and the class votes or narrows to one shared problem to tackle as an ongoing project. This mirrors real STEM practice: defining a problem with evidence before jumping to solutions.
- **"Build It, Break It, Fix It"**
 - *Focus: Plan/Create/Test/Improve phases · CS tie-in: 3-5.PA.1 and 3-5.PA.3 (collaborate on problem-solving steps; test and debug)*
 - Once a problem is chosen, groups design a first prototype solution — this could be a physical model (a redesigned recycling station, a sunlight reflector for the plant corner) built with basic materials (cardboard, tape, string, clothespins), or a step-by-step process solution if the problem isn't physical. Critically, build in a real test: does the prototype actually work when tried under real conditions?
 - Groups present their prototype, another group "stress-tests" it, and everyone documents what broke and why — framing failure as data, the same way Lesson 16 (Debug the Morning Message) frames bugs. Groups get one revision round based on what they learned. Debrief: "What did failure teach you that success couldn't have?"

Description of Main Activity/Project:

- **Possible Activity/Project Ideas:**
 - **WHAT:** Each lesson is designed for a 10–15-minute morning meeting using vertical whiteboards. Activities are grounded in Peter Liljedahl’s Building Thinking Classrooms framework: visibly random groupings, non-permanent vertical surfaces, and tasks that require genuine thinking rather than answer-finding. Every lesson includes an SEL character trait connection and a closing debrief question. Computer science connections are embedded naturally — not as a separate subject, but as a way of thinking. The goal of each lesson is to have students work together with peers. While solving problems, they are also focusing on an SEL trait. Each task is specifically designed to help students practice that trait.
 - **WHEN:** Each week, a new trait is introduced. Each day of the week has a specific task. This makes it easy to remember, and students know what to expect. Towards the end of the school year (April/May), there is flexibility to review skills that are lacking or were not covered during the year. Due to state testing and breaks, this flexibility allows teachers not to feel rushed to cover skills.
 - *Monday: Intro skill and discuss meaning*
 - *Tuesday: Question of the week/ application*
 - *Wednesday: SEL group activity*
 - *Thursday: Flex day (finish activity, class goal setting, catch up)*
 - *Friday: Reflect and shout out to students*
 - **HOW:** At the beginning of the week, students will be discussing skills and questions as a whole class or in turn and talk. Then, students will work mostly in groups using vertical whiteboards to aid collaboration. These vertical whiteboards are essential to the lessons and beyond.

Section 3: Module Sequencing and Assessment:

- **Brief description of sequenced learning objectives:** The overarching goal is to teach SEL traits through an activity that allows students to meet computer science standards. Each trait is deliberately planned into the calendar to create a strong classroom community at the beginning of the year and to review expected conflicts during long periods of teaching and breaks.
 - **August:** Cooperation, Teamwork, Inclusion
 - **September:** Self-Control, Respect, Fairness, Acceptance
 - **October:** Encouragement, Kindness, Empathy, Compassion
 - **November:** Responsibility, Ownership, Gratitude
 - **December:** Generosity, Loyalty, Integrity
 - **January:** Honesty, Perseverance, Resilience, Tenacity
 - **February:** Commitment, Trustworthiness, Focus, Patience
 - **March:** Forgiveness, Bravery, Humility, Respect, Confidence
 - **April:** Review needed skills
 - **May:** Review needed skills

- **Brief description of formative and summative assessment approaches:** For each lesson, students will complete the activity on a vertical whiteboard using the “Building Thinking Classroom” framework. This framework is meant to help teachers scan the room and make formative assessments of students' understanding. Based on students' responses to the activity, the teacher will know whether to pull students in for reteach or to correct misconceptions, to have a gallery walk and discuss the uniqueness of the work, and to question students' understanding.
- **Recommendations for implementation:**
 - Quick explanation of “Building Thinking Classroom”:
 - One marker per board. I like to binder clip a sandwich bag to the top of the whiteboard. In the bag, I keep an eraser, a marker, and students place their playing cards inside
 - **Visible random grouping:** I use old playing cards. Students receive one with a number 1-7 on it, which corresponds with their whiteboard. I have 7 boards hung in my room. I keep groups to 3/4 students. Within a group, there are dots of different colors. When students are struggling to share the marker, I call out a color to assign a “writer”.
 - When students are stuck, they may walk around the room and look at their peers' boards. It's important that they do not talk to each other. I explicitly teach students to put their hands behind their backs and slowly walk around silently.
 - Students are standing, never sitting! There is supporting research that standing increases thinking

Section 4: Module Files – Encouraged/Optional

- **List of attached files:**
 - [Lesson Plans](#)
 - [Yearly Calendar](#)
 - [Vertical whiteboards](#)