CORNELL
TECH

K-12 CS Coaching Toolkit

Written by: Meg Ray, Kelly Powers, Joe Melendez and Liz Gallo
Designed by: Niti Parikh
Edited by: Diane Levitt

In 2014, Cornell Tech launched the Teacher in Residence program as an effort to make computer science teachable in New York City public schools. NYC schools are primarily made up of Black (24%) and Latinx/Hispanic (42%) students, most of whom (72%) come from families living in poverty. After several fruitless attempts at using professional development alone to bring CS to life in partner schools, we leveraged the work of West and Cameron (2013) to design a computer science content coaching initiative that would bring comprehensive support to teachers new to computer science education.

In West's words, "Content-focused coaching provides a collaborative means for specialists and teachers to plan, teach, and reflect upon classroom lessons. Specialists provide individualized, adaptive, and situation-specific professional learning focused on content, pedagogy, and student learning." While initially designed for math education, we found this model translated very well to computer science. In our iteration, CS master teachers are embedded in schools in long-term residencies, providing professional development, helping administrators think through a CS-infused instructional plan, and coaching teachers in 5-6 week cycles to build their knowledge and practice.

RESOURCES

Cards with this icon 🔗 have related resources and citations on our website.

k12.tech.cornell.edu/what-we-do/resources

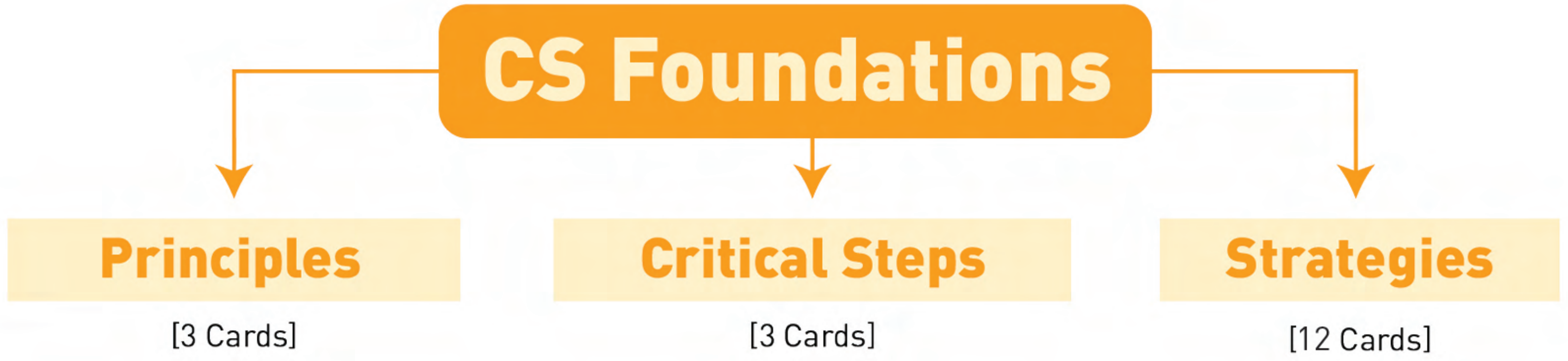
(this QR code will take you there).

CONTENTS

CARDS	
	CS Foundations
3	**Principles**
3	**Critical Steps**
12	**Strategies**
	CS Coaching
8	**Coaching Lenses**
27	**Coaching Practices**
	Coaching Cycle [Co-Planning]
7	Critical Steps
12	Instructional Design
	Coaching Cycle [Co-Teaching]
5	Interventions
11	Look Fors
	Coaching Cycle [Reflection & Feedback]
6	Critical Steps
8	Strategies
	CS Classroom
7	**Pedagogical Lenses**
44	**Classroom Practices**

CS Foundations

Foundation cards support CS coaches and leadership teams in planning and goal-setting.


CS Foundations
Principles
[3 Cards]
Critical Steps
[3 Cards]
Strategies
[12 Cards]

Principles

tech.cornell.edu/impact/k-12

Principles reflect overarching values for CS education.

Coaching for Equity

“Computer science for all students requires that equity [fairness, justice] be at the forefront of any reform effort. When equity exists, there are appropriate supports based on individual students’ needs so that all have the opportunity to achieve similar levels of success. Equity is not just about whether classes are available, but also about how those classes are taught, how students are recruited, and how the classroom culture supports diverse learners and promotes retention.” Equity is not the same as equality, which denotes sameness. Our commitment to equity is a commitment to access for students of all identities and abilities.

tech.cornell.edu/impact/k-12

Coaching for equity requires that we help teachers think through factors like race, gender, disability, socioeconomic status, and English language proficiency, to create learning activities with high expectations and access for all. In a classroom framed by equity, all students feel welcome and empowered to learn. During initial meetings, coaches and teachers should review the identities and abilities of all learners. These should be top of mind as lessons are planned: Is there representation in the curriculum materials? Are any disabilities accomodated? Are any stereotypes removed, mitigated, or raised for discussion? Embracing an explicit equity focus may create discomfort, but will ultimately build teacher practice.

Coaching for Agency

Personal agency refers to “the sense that I am the one who is causing or generating an action” (Gallagher 2000, p. 15). A student with a sense of personal agency perceives himself/herself as the subject influencing his/her own actions and life circumstances (Bandura 2006; Gallagher 2000). (Alper, S. 2020). As coaches, we work to build computational agency in teachers, so that they may help students develop agency themselves. This helps prepare teachers and students for full citizenship in the digital age.

During professional development and co-planning, coaches should help teachers build their understanding of the computing they teach beyond that expected by the lesson. This deeper mastery of content builds agency not just in computing, but in teaching, giving teachers confidence to differentiate, formulate multiple explanations to tricky problems, and demonstrate the variety of solutions often possible in CS.

Coaching for Rigor and Joy

When we work to bring students to the intersection of rigor and joy, we match our highest appropriate expectations for our students with our most engaging, interactive, motivating pedagogy. This is as much about balancing cognitive load as creating a vibrant lesson. The fact that we have students from groups historically excluded from tech in our CS classrooms does not mean we should reduce our expectations for their enjoyment or achievement.

Coaches should approach co-planning with explicit questions about rigor and joy. Have we met our own standard for the level of difficulty? Have we aligned the lesson to CSTA or our state standards? Will our students be engaged and excited by this lesson? During reflection and feedback, revisit this balance. Were students appropriately challenged? Was the lesson too hard, or too easy? Did we have enough opportunity for differentiation? Did students seem excited by the work?

Critical Steps

Critical Steps includes those activities in which every program must engage

Purpose Statement

Understanding and describing your purpose and motivation as a coach will focus your work and help teachers understand your role.

Write a statement that captures your core values and beliefs about CS education. Include your coaching goals, what drives your passion, and what motivates you to inspire and develop teachers’ CS teaching potential. In your coaching practice, when you make decisions or face challenges, return to this statement for guidance. Share this statement with teachers and schools as you begin your work together.

CS Vision

Building a CS Vision with the school will guide development of the program and your work with teachers. It should reflect what school leaders and other stakeholders want the school's CS implementation program to achieve.

Make time with stakeholders to visualize a mature CS education initiative in their school. What are their goals? Criteria for success? Do they envision a school where CS is a standalone course, or do they see CS integrated into other disciplines? Reflect on your CS Vision periodically with stakeholders to ensure that progress is being made to achieve its goals.

CS Implementation Planning

Administrators should be encouraged to create an overall plan for CS. Involving school administrators with high level decision making and keeping them informed helps them create a school culture that supports and sustains CS.

At an initial meeting (or series of meetings), co-design a written implementation plan with administrators and school appointed leaders. Be sure to give and solicit feedback about pacing, sequencing, availabity of devices, and ongoing school initiatives or curricular roadmaps. Provide regular updates on CS implementation and progress towards any school-wide goals. Share successes with the school community.

Strategies

tech.cornell.edu/impact/k-12

Strategies provides a variety of ideas and options for building an accessible, inclusive and sustatinable CS initiative.

Self-Awareness

Coaches must keep the focus on teacher learning and growth. It is vital for coaches to reflect and iterate on their own coaching. Practicing self-awareness and reflection helps center the coaching on teachers.

Be reflective, iterative, and intentional. Make time to reflect after each coaching cycle and make adjustments based on direct and indirect feedback gathered throughout the cycle. Use these reflections to examine implicit biases, become more aware of any triggers, and practice mindful stress reduction and emotion regulation during coaching sessions.

Your Learning Journey

Understanding and preparing to teach CS is a huge undertaking for teachers. Acknowledging and sharing your stories of challenges and growth shows compassion and helps develop the coach/teacher relationship.

In early conversations, offer some background or personal anecdotes, especially about missteps or challenges (funny is good here--laughing at yourself will help teachers go easier on themselves). Highlight strategies you used to perservere and resources you found helpful in your early years teaching. Emphasize that learning CS is a process for teachers as well as students, and that building mastery takes time but is within reach.

Professional Learning Networks

CS education is a dynamic, evolving field. Many researchers and practitioners generously share what they're doing and learning. Bringing administrators and teachers into those learning communities helps integrate CS and builds a sense of possibility and belonging.

Encourage teachers to join learning communities like CSTA and their local chapter (this may mean showing them how to join). Introduce teachers to relevant experts whose work you follow and trust. Share links to their resources, including social media accounts, books, and websites. Suggest they attend workshops and conferences when they can, and help connect them to scholarships or school funding.

School Priorities

School or district goals or initiatives can compete for PD and classroom time. Aligning CS education to existing initiatives helps prioritize and sustain CS.

Speak with administrators and teachers about ongoing commitments or priorities. Together, look for syngeries that can help meaningfully integrate CS into the school’s identity.

Class Visit

Insights from an initial classroom visit strengthens the coaching relationship and may inspire potential coaching strategies or practices.

Inform your coaching plan through a 30-40 minute visit to learn about classroom management, transitions, and routines. Look for choices the teacher has made on bulletin boards or other posted materials. Notice how the students work and/or collaborate. Make note of any potential ties to CS you may be introducing. This is purely an information gathering time, not an opportunity for feedback.

Program Needs Assessment

Even schools with prior CS education experience have successes and challenges that a coach can identify and share with leadership. This supports the school in providing a consistently high quality CS education for all students.

Periodically examine the whole picture of CS implementation, including any plan and its implementation. Identify strengths and gaps in CS content or pedagogy across grades, and work with the school to adjust their plan to build on strengths and fill gaps. When possible gather data from CS courses offered at the school or district.

Professional Learning Communities

Coaching cycles are limited in time and may be focused on specific teachers. Developing other CS learning and discussion opportunities builds momentum for the program and keeps CS education visible.

Ask school leaders to form a Professional Learning Community (PLC) for CS that meets frequently throughout the year to provide PD, plan CS events, revisit the CS vision, or solicit feedback on the status of the implementation plan. These PLCs might be based on grade level or band, subject, or open to all faculty.

Administrators' CS 101

Familiarizing administrators with CS content, and teaching practices, will allow for better teacher support and communication within the school community.

Offer administrators the opportunity to participate in a hands-on CS lesson themselves, either in a meeting with you, or by participating in PD with their teachers. Invite them to see CS teaching in action in their schools. Use the time to point out examples of strong CS teaching and CS-specific strategies. Familiarize administrators with CS standards and learning goals and provide resources for further study.

Building Mentors

Developing teacher mentors will work toward building a sustainable CS ecosystem that does not rely on you, and will give teachers a colleague they can turn to for support.

With administrators' support and approval, seek out and develop mentor teachers who are eager to take a leadership role around CS education for the school or community. Invite and support these mentors to co-lead PLCs and PDs. Encourage mentors to serve as a point person to disseminate information, CS learning opportunities and related events to their colleagues.

Family Engagement and Inclusion

Exposing families to CS will build their understanding of why CS is a crucial component of their student's education and build advocates to sustain the program.

Encourage schools to host family nights and invite families into the classroom during CS instruction. Provide resources for schools to plan family events that introduce families to computational thinking games and activities. Engage parents in supporting CS instruction and expose them to academic and career pathways with CS.

CS for All Teachers

All teachers play a big role in student learning. Offering CS learning opportunities to them increases access for all students and demonstrates to them that CS matters.

When creating a CS vision and implementation plan, advocate for all staff that interact with students, including special educators, teacher assistants, curriculum specialists, EL specialists, and paraprofessionals to have some CS learning opportunities. This may be joininig PD, creating PLCs, or offering bite-sized or just-in-time coaching during classroom implementation.

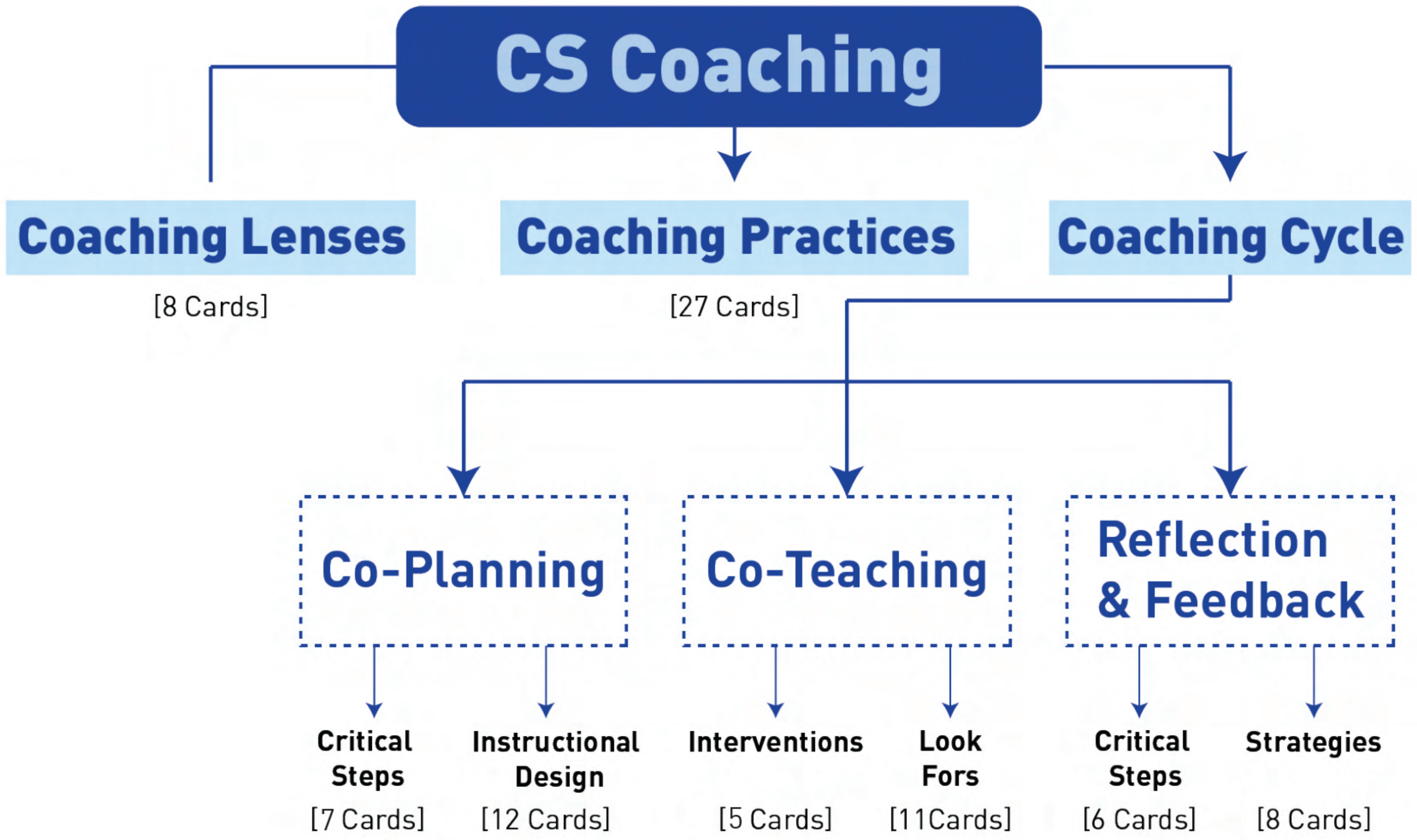
Build a CS School Community

Making computer science part of the everyday life and culture of the school ensures sustainability and creates momentum, even for teachers who are uneasy about teaching computing

Encourage administrators and teachers to make space to showcase CS. This can include bulletin boards of student work or CS heroes in entryways or hallways, family engagement nights, CS Ed Week celebrations, professional development for the entire school staff (not just classroom teachers), and, if possible, CS-related posters or even t-shirts that the whole staff can wear on CS activity days.

CS Coaching

This toolkit supports content coaching in the CS classroom. A coaching relationship is built on trust and shared expertise and is always consultative and never evaluative. The coaching cycle, as defined by West and Cameron, includes three components: co-planning, co-teaching, and reflection and feedback. In addition to supporting this coaching cycle, the cards in the CS Coaching section include Coaching Lenses, which give coaches an overall framework for guiding discussions, and Coaching Practices, which offers concrete, detailed strategies on working with teachers to achieve their goals.


CS Coaching
Coaching Lenses
[8 Cards]
Coaching Practices
[27 Cards]
Coaching Cycle
Co-Planning
Co-Teaching
Reflection & Feedback
Critical Steps
[7 Cards]
Instructional Design
[12 Cards]
Interventions
[5 Cards]
Look Fors
[11Cards]
Critical Steps
[6 Cards]
Strategies
[8 Cards]

CS Coaching

Coaching Lenses

tech.cornell.edu/impact/k-12

Coaching Lenses give coaches an overall framework for guiding discussions with teachers and/or administrators. They can also be used to reflect on your own coaching practice to inspire or inform areas for further development or improvement.

Gender Equity & Inclusion

Gender inclusive teaching practices can build the confidence of all students. Research shows that girls' participation in CS drops dramatically by middle school and that girls are underrepresented in high school CS classes. Transgender students, non-binary students, and students with other gender identities remain invisible in CS education because they have not yet been counted.

Promote gender equity: through visual representation, curriculum content, gender inclusive language, and critique of gender stereotyping. During co-teaching, check that teachers call on students and assign challenging assignments with equity. Coaches should reflect on their own biases and attend to how gender identity influences their coaching.

Trusting Relationships

Coaching centers on cultivating a trusting, collegial relationship between the coach and teacher. Coaches should be clear about their role (especially relative to evaluation), the kinds of information they collect and share with others, and the kinds of information that are kept confidential between the teacher and the coach.

Work with the teacher to address complex issues related to teaching and learning, and provide feedback with honesty and care. Be clear about the information that you will share with others and how you keep the rest confidential. Address difficult classroom issues and provide critical feedback with honesty and care.

Student Impact

While coaches may not work directly with students, their ultimate goal is to positively impact student learning and well-being through improved teacher practice. Student learning and growth is a priority for educators. The outcome of any support and development work must have students' best interests at the center.

Pay attention to student impact and incorporate student learning data into your coaching practice. Collect anecdotes of individual student impact and survey impact across groups.

Racial Literacy

Racial Literacy is:

- The knowledge, skills, and awareness needed to talk thoughtfully about race and racism; this naturally includes having a rich vocabulary including terms such as race, racism, prejudice, ally, upstander, and so on
- The ability to identify racism when it happens
- Having strategies to counter or cope with racism
- Understanding the role racism plays in society

Coaches must understand the impacts of institional racism and recognize their own triggers and implicit biases. Coaches should actively seek training or support from experts and participate in affinity groups, rather than relying on colleagues of color. How might I discuss and respond to topics of race in a mindful manner with both teachers and administrators? How might I remain aware of how my racial identity influences my coaching? How might I coach across racial differences with humility and grace?

Empathy

Practicing empathy strengthens your coaching relationships and helps teachers to feel safe and supported. These feelings may increase teachers' professional growth and performance in areas related to their coaching. Communicating empathy effectively also serves as a model for teachers to use in the classroom.

Understand and acknowledge the complexities of the teacher's role. Consider not just internal motivations, but external circumstances and the impact of systemic pressures on teachers. Practice empathy by accepting the teacher's feelings and reactions as valid and responding with compassion and respect.

Teacher Voice & Choice

The coach acts as a guide, honoring a teacher's expertise, experience, and practice. The teacher being coached will ultimately make their own decisions about planning and implementing CS in their classroom.

Honor the teacher's expertise regarding their own students, the curriculum, and the class culture. Combine your experiences and resources with the teacher's expertise and autonomy. Rather than imposing your own choices and acting as the knower of all, support the teacher in making their own instructional decisions.

Curiosity

Curiosity can strengthen your coaching relationships and boost your empathy for the teachers with whom you work. Curiosity has also been linked to increased learning, engagement, and performance. When you practice curiosity, you are also modeling the approach for teachers to use in the classroom.

Set your intention to approach a coaching session with curiosity. Try to notice when you might be assuming intentions behind actions or the outcome of a lesson. Step back and remind yourself that you may not know the answer.

Creativity

Coaching requires the ability to learn from and adjust to different perspectives, personalities, cultures, and logistics. Creativity encourages the development of new solutions and leverages a customizable approach to coaching.

CS education is still a relatively new and developing field. The “right” solution, lesson, curriculum, or practice may not exist, so look outside of the CS education field if necessary. Integrate ideas from different curricula. Be open to inspiration even from the least expected sources.

Coaching Practices

tech.cornell.edu/impact/k-12

Coaching Practices are a collection of strategies and best practices that offer coaches concrete detailed approaches that they may use when directly supporting the teachers, administrators, or other stakeholders with whom they work.

Use, Modify, Create Curriculum

Scaffolding ownership of curriculum will help teachers feel confident in teaching lessons to their students.

Use the CS strategy “Use, Modify, Create” as a way to develop teacher ownership of CS curriculum. When teachers start out, have them use pre-written curriculum. Then have them modify the curriculum slightly. Then, encourage them to create their own projects and lessons. (I. Lee)

Technology Practice

Teachers often feel most uncomfortable because they do not know how to use the devices of computing. Tinkering with the technology students will be using helps teachers gain confidence and expertise.

Reserve time during co-planning for teachers to use technology as their students will. When possible, build time into the coaching or PD schedule that is not focused on how or what to teach, but on mastering the devices themselves. Encourage teachers to work on projects that reflect their own interests and curiousities.

Build It First

Building a project or completing as task ahead of time can support teachers in developing their own content knowledge and improve their confidence and flexibility when supporting students.

Have the teacher complete the entire task or project that students are assigned. This can be done independently or they can pair program with the coach. Invite teachers to use this experience to clarify instructions and address potential barriers to student success. They can also share their artifact with students as an exemplar.

Continuous Learning

Computer science education is a dynamic and fast-changing field, and coaches should both continue to engage in learning experiences like workshops and conferences, and encourage the teachers they coach to do the same.

Identify and share key resources, and list the professors, leaders, organizations and research that you follow to inform your practice. Share stories of how these resources have helped you to improve your craft, acquire new lessons, and learn effective CS teaching strategies. Remind teachers of upcoming conferences and deadlines to register.

A Step Ahead

Broadening teacher knowledge beyond what they will need to teach their students builds confidence, fluency and the capacity for differentiation.

When co-planning CT or CS activities, offer the teacher one additional step or idea beyond what you expect teachers to learn.

Troubleshooting Tech

Helping teachers anticipate common technology issues during planning makes early lessons run more smoothly

tech.cornell.edu/impact/k-12

Identify common tech issues and share strategies for solving them in real time, including rebooting devices or confirming passwords. This should be part of all lesson planning that involves devices.

Technology Procedures

Creating effective student procedures for accessing and/or distributing devices saves time and avoids confusion and unnecessary disruptions to instruction.

Review, plan, and implement the procedures for how students will access the technology needed for the lesson. Think about how will devices stay charged and be distributed, and what materials students will need.

Embrace Challenges

Explicitly using the CT approaches like perseverance and debugging to demystify overcoming challenges as they arise helps build teacher confidence and model these behaviors for students.

When you come across an error or something goes wrong, don't try to hide it. Normalize errors, and model and label debugging skills and perseverance. Make explicit connections to how the teacher might do this with their class.

Demystifying Computing

Increasing teachers' understanding of computing systems and networks will build their independence, boost their confidence in teaching CS, and help them model knowledgeable troubleshooting for students.

Talk about computing systems and networks as you are using them, using appropriate CS vocabulary, explaining what you mean without jargon. Give specific and concrete examples of how computers work. When troubleshooting a network issue with a teacher, explain the different parts of a network and how they work together.

Paint a Picture

Teachers new to the subject area need a clear understanding of what CS education is and, most importantly, what it looks like.

Curate and share videos, websites, photos and lesson plans of strong CS teaching to share with educators new to computing. Look at them together and narrate the distinctive content or pedagagaogical choices the observed teacher is making (using this toolkit when appropriate). When possible, invite the teacher to observe a peer's classroom.

Professional Learning Roadmap

Support the teacher in creating a personalized plan for professional learning. The field of CS education continues to evolve. Teachers of all experience levels must continue to develop their practice. All teachers should have a professional learning plan against which they check their own journey.

tech.cornell.edu/impact/k-12

Use the CSTA teacher standards as a resource helps teachers see areas for growth to plan professional learning. Expose teachers to external opportunities including conferences, communities of practice, workshops, books, and research opportunities.

Teacher Goals

Clear goals help both coaches and teachers gauge progress and understanding. Goal setting helps validate professional growth and risk taking.

Have the teacher self-assess their growth toward any existing goals. After reflecting on the lesson, ask the teacher what areas for growth they want to focus next time and/or together review the teacher standards to pick goals to focus on over time.

Brainstorming

Working out problems or building ideas together leads to creative solutions and nourishes a collaborative relationship.

Identify an area of the lesson delivered, or classroom scenario to revise and improve. Collaboratively jot down ideas of possible solutions. Purposefully hold back on sharing ideas until the teacher has shared a few. Then add on to the teacher's ideas. Finally choose a solution and make a plan to test the solution during the the next lesson.

Active Listening

Understanding the teacher's perspective is important to a collaborative coaching relationship.

Listen to teachers reflect on how the CS lesson went. Encourage them to expand on these thoughts by asking clarifying questions, or prompting them to describe with a bit more detail, or provide an example. Validate their perspective by paraphrasing what they've said.

Start Strong

Identify and build on a teacher's strengths, fostering motivation and facilitating continued professional growth. This also models a strengths-based approach that can be used in the classroom.

Share strengths in their practice that you have observed. Encourage teachers to take small risks to grow in areas they're confident in. When they struggle, encourage them to transfer their strengths to this new application. Build on what is already working well when addressing challenges.

Guide on the Side

When a teacher encounters a problem or obstacle related to CS instruction, the coach may have a good idea of what is happening and how to effectively address it right away. However, supporting the teacher's own exploration and solution-building process is often more effective in building teacher capacity. The coach is also providing a model for use of this strategy in the classroom.

Resist the urge to solve implementation issues for the teacher. Introduce the process of resolving an issue through an investigation. As a coach, guide a close look at the problem with increasingly specific questions. With this series of questions, lead teachers to identify their own understanding of an issue and develop a strategy to address it.

Repair Relationships

None of us is perfect. We don't always say or do the right thing. Moreover, people have different lenses for interpreting meaning that can lead to misunderstanding. You may have made a mistake that led to hurt feelings. What matters here is that as the coach, you take the initiative to address the feelings, rebuild trust, and maintain professional learning relationships.

Notice if a teacher becomes “closed off”, withdrawn, or passive aggressive. Rather than letting these feelings fester, take steps to address them. First examine your own feelings and implicit biases and reflect on your actions as the coach. Then have an open conversation with the teacher. Depending on your professional relationship, this might include sharing your own experiences and feelings, doing a personal check-in, surfacing cultural differences, reviewing roles, or apologizing and making amends when a misstep has occurred.

Taking the Temperature

Coaching isn't just about what you say, but knowing when and how to say it. Coaching is most effective when teachers are mentally and emotionally flexible.

Gauge the teacher's mood and receptiveness in the moment. Use this information to guide the conversation including how and when you provide different types of feedback, how much to push thinking and skills, or if you need to pause for a personal check-in.

I Do, We Do, You Do

The gradual release model builds teacher confidence, strengths and identity as a new CS teacher.

First, model teaching a new concept (I do). Then you and the teacher might co-teach a lesson (we do). Finally, the teacher teaches the new concept independently and receives feedback from you (you do). Encourage teachers to use this strategy with their students as well, to build their confidence and sense of belonging.

Learning Journey

Understanding and preparing to teach CS is a huge undertaking for teachers. Acknowledging and sharing your stories of failures and growth shows compassion and helps develop the coach/teacher relationship.

In early conversations, offer some background or personal anecdotes, especially about missteps or challenges (funny is good here--laughing at yourself will help teachers go easier on themselves). Highlight strategies you used to perservere and resources you found helpful in your early years teaching. Emphasize that learning CS is process for teachers as well as students, and that building mastery takes time but is within reach.

Honor Expertise

The coach/teacher partnership implies a hierarchy of expert/novice which may impact the coaching relationship. How the coach honors the expertise and strengths the teacher brings to the partnership will influence the effectiveness of the coaching experience.

Always value the strengths and expertise that teachers bring to the partnership. Recognize and acknowledge the teacher's strengths and knowledge, and build upon them to improve CS instruction. Find moments to celebrate teachers' growing CS content and pedagogy.

Storytelling

Storytelling is a powerful way to learn, communicate, and make meaning. It can help you better understand a teacher, communicate ideas and concepts, process experiences, and to internalize learning. Stories that model resistance and persistence may help teachers manage difficult situations. Leverage storytelling to build rapport, promote learning, prompt critical thinking, and build teacher confidence.

Use storytelling for building the coaching relationship and sharing lessons learned from the one's CS teaching experience. Invite teachers to tell stories about their own experiences teaching CS: what the teacher thought would happen, what did happen, what the teacher thought and felt, and what the outcomes were.

Coaching Questions

Coaching is often guided through a set of carefully crafted questions.

Craft open ended questions to guide constructive conversations and advance teacher learning. Focus on engaging teachers in dialogue to provoke deep thought and reflection. Anticipate teacher questions, and lean on their expertise to respond. Keep creativity and positivity high by actively listening to the teacher and using language that invites dialogue.

Flexible Coaching Cycle

The coaching cycle is meant to be a flexible structure that allows you to respond to the needs of a teacher or a school. Coaching works best if it can happen within the constraints of a school and if it is centered around relationships. Be aware of the tension between flexibility and prioritizing quality coaching opportunities.

A typical coaching cycle is iterative, and consists of: co-planning → co-teaching → reflection and feedback → repeat. However, coaches must remain flexible and responsive to shifting schedules and other demands on teachers' time. Look for opportunities to make these important components of the coaching cycle happen.

One At a Time

Using too many CS Teaching Practices in a lesson may be overwhelming and impact teacher success.

Leverage one new CS Teaching Practice at a time, until the teacher feels and demonstrates mastery.

Walk Through Lesson

Unpacking a lesson will help teachers become fully prepared and helps you identify any gaps where the teacher needs support.

Go over all aspects of a lesson together. Make sure the teacher is comfortable with the content, structure, and flow of the lesson. Invite them to modify the lesson for inclusivity and cultural responsiveness as you discuss. Use strategies such as modeling instruction where the teacher takes on the role of a student while allowing them time to reflect on their experience. Consider scripting and rehearsing responses to common student questions.

Promote CS Careers

Demonstrating the importance of CS in the workplace helps make CS relevant, but teachers may be unfamiliar with the nature of these roles.

In co-planning, share resources about careers in computing with teachers and help teachers highlight CS careers for students. Whenever possible, tie these ideas to what students are learning. Help teachers make real-world connections between CS and other subjects they are teaching.

Coaching Cycle

Co-Planning, Co-Teaching, and Reflection & Feedback make up the three stages of the coaching cycle. The basic practices of coaching include moving through each of these stages repeatedly and over time with individual teachers or small groups of teachers.

Coaching Cycle

Co-Planning

tech.cornell.edu/impact/k-12

Co-planning is a primary way that coaches can build relationships with teachers and help build their content knowledge and pedagogy. It occurs before the lesson is implemented with the coach present, called co-planning. It is the time to choose curriculum, plan, modify and create curriculum resources, and review how lessons will be enacted.

Coaching: Co-Planning [Critical Steps]

Who Are We Designing For?

Developing an understanding of their individual student's needs will facilitate a teacher's efforts to create an inclusive and positive classroom culture.

Encourage teachers to creatively gather information on student cultures represented, languages spoken in their classroom, strengths and needs, and how students interact with each other. With specific students in mind, collaborate on a plan to build an inclusive classroom environment and CS learning experiences for all students.

Use What's Working

Leverage a teacher's existing routines and protocols. Transferring routines across subject areas helps maintain consistency and builds on the teacher's strengths in other subject areas.

During initial meetings and classroom visits, identify and observe class routines and practices. Encourage the teacher to incorporate and build on these practices as they bring CS to their classroom.

Breaking Barriers

Identifying barriers that are inhibiting students from learning is the first step in thinking about strategies to remove them.

Encourage teachers to think about struggling students during co-planning. Decompose the barrier. Ask the teacher: What are students struggling with, and why? Then, brainstorm creative ways to remove or reduce those barriers.

The Teacher's "Why"

Coaching relationships and plans should be informed by a teacher's motivation for teaching, and the inspiration to add CS to their classroom.

Develop a set of questions to learn about their journey to the teaching profession. Meet with the teacher to learn more about them. Take notes as you listen. Ask the teacher why they are teaching CS? Ask a follow up question to understand if they feel motivated or pressured to teach CS. Show compassion for the demands placed on teachers and offer to be a partner.

Describe Your Role

Establishing clear expectations for roles and what will happen during coaching creates a foundation for a good working relationship. As a coach you remain in a non-evaluative position with a common goal of bringing CS to the classroom.

Describe the process of the coaching cycle the teacher will experience. Clearly describe your role as a coach to support, guide and help the teacher grow.

Coaching Goals

Coaches must enter every co-planning session with a plan and well-defined goals, even if it's the first session with teachers. Both the teachers' and coaches' time is valuable, so coaches must ensure that all time spent with a teacher is used efficiently and focused on teacher learning and growth.

Before your co-planning session with a teacher, reflect on the coaching implementation plan developed with the school leadership team, especially if it is your first session with the teacher. Create goals for the teacher that build towards the school's overall goals as established in the coaching implementation plan. Make sure the goals for the session also reflect on and take into consideration the needs of the teachers, the resources they have access to, and the culture and identity of the students in their classroom. Ask yourself if the goals contribute to the teachers learning and growth. Are the goals the best use of yours and the teacher's time?

Decide Ahead

Teachers have different comfort levels with having other adults with them while they teach. Determining a plan and setting expectations ahead of time helps to ensure a positive coaching relationship.

During co-planning, review co-teaching options of presenting instruction together, modeling instruction, quiet observation, and other coaching postures. Is the teacher comfortable with you interacting with students during worktime? Would they prefer you provide some real time feedback (see Bug in Ear) or hold all input for reflection time? Reassure teachers that a coach's role is to support their work and not to officially evaluate them.

Lesson Objectives

Setting clear objectives helps keep the lesson focused and connects the learning experience to the assessment. This activity also builds teachers' content knowledge and expectations.

Use the ABCD model of setting objectives.
Audience - Who are your learners?
Behavior - What will they be doing? What is their task?
Condition - What tools or resources will they use? When should the task be complete?
Degree - To what degree of mastery should the task be completed?

Lesson Assessments

Assessing student progress should be built into learning experiences.

Check the plan to assess students' learning at the end of the lesson. Determine if it is aligned to the CS objectives and activities.

Coaching: Co-Planning [Instructional Design]

Content Area Connections

Connecting CS to other content areas demonstrates its relevance to students' academic experience.

Find opportunities to integrate CS content into other content areas. Ideas include connecting CS to grammar and syntax in Language Arts, algebraic/Boolean equations in math, experimentation in science, language acquisition in foreign language, storytelling in history.

Real Life Connections

Connecting CS to our daily lives brings relevance to the instruction and help students understand how CS impacts everything we do.

In building out lessons, identify ways that students will use CS in their everyday lives, including jobs and current events.

Curriculum Evaluation

Teachers need to become critical thinkers about CS curriculum, including how to evaluate curriculum, lessons and projects and determine the appropriateness for their students.

With the teacher, evaluate the CS curriculum for accessibility, equity, CS content, and pedagogical practices. Ask critical questions such as: How would you modify this curriculum for your class? Will this engage all of your students? Does this approach make sense for you and your students? How does this align to CS standards? Is the pacing right for your students? Does the content provide rigor for your students?

CS Vocabulary

Teachers need to understand and use CS vocabulary fluently.

Explicitly work with teachers to build and reinforce vocabulary around CS. Model the correct use of computing terms, check for meaning, and ask teachers to demonstrate their use in the context of the lesson.

CS Content Standards

All units, lessons, and activities must be connected to relevant CS standards. Aligning to the standards assures that curriculum is building toward student mastery of CS.

Consistently review standards with the teacher and reflect on how which standards are being met, where there are gaps, and student progress toward mastery. Ensure that you look at local, national and course-specific standards.

Coaching: Co-Planning [Instructional Design]

Career Connections

Many educators are relatively unaware of the breadth of STEM and CS careers and the intersection of computing across fields

Talk with teachers about CS careers. Share resources that highlight CS careers and integration across different fields to increase student awareness. Encourage teachers to invite professionals using CS in the workplace in different ways. Help build these connections into lessons and class discussions.

Social Emotional Learning (SEL)

CS and computational thinking open opportunities to explore emotional and social situations. Collaborating with peers in a scaffolded process allows students to practice expressing themselves and sharing their ideas. SEL helps students build self awareness, self management, social awareness, relationship skills and responsible decision making, key dispositions for creators and consumers of technology.

tech.cornell.edu/impact/k-12

Create lessons that include affective content, possibly through storytelling, problem solving, or data collection and analysis. Make time for students to share their own feelings during the presentation of the lesson and during reflection afterwards. Establish routines for collaboration (during think-pair-share or pair programming, for example) that guide students toward exchanging thoughts and feelings respectfully and productively. Sharing emotions can create a dynamic discussion. Prepare teachers with strategies to help students regulate their feelings and, if necessary, to provide comfort and support.

CS Integration

Integrated lessons and projects can make connections and provide more powerful and deeper learning experiences for our students while making time for CS in the school day. In addition, developing teacher agency to bring CS into existing topics helps sustain computing in their classroom.

Brainstorm ideas with the teacher to bring computing into another discipline. Look for existing topics that can be authentically deepened through the addition of CS. Bring examples of subject integrated resources.

Student Identity

Take students' background, identity and culture into account when planning CS lessons. Students become more engaged when they feel they have a connection to the content they are learning.

Find opportunities to integrate appropriate cultural and background knowledge into CS instruction. Be sure to avoid stereotyping. Encourage teachers to use images that reflect student identities.

Data Science

Data is used to power computing, and the ability to understand and work with data is critical for all citizens. Students of all ages should form questions, collect and analyze data, and interpret and communicate what they've learned. Students should also know how their personal data is used online, and how to protect their privacy and confidentiality.

Look for opportunities to build interaction with data into computing projects. During discussion, prompt students to notice and wonder about questions that can be better understood through data collection and analysis. Encourage students to look for patterns in data to make meaning from what they've found, and talk about different ways to present data with drawing or computing tools. During lessons on digital citizenship, talk about privacy and confidentiality, the commercial use of personal data, and when and how to share personal information online.

Co-Teaching

Co-teaching is a time when teachers teach a CS lesson with the coach present. A coach, while ensuring a joyous and positive collaborative time with the teacher, focuses on building teacher's CS practice and pedagogy. A coach actively engages during lesson implementation through modeling, co-teaching, or observing to later provide feedback.

Coaching: Co-Teaching

Interventions

tech.cornell.edu/impact/k-12

Intervention cards are to be used during a co-planning session to establish the agreed upon role of the CS Coach in the classroom as a teacher facilitates their lesson. They are meant to create transparency in how a coach will support or intervene while a teacher facilitates. removing any uncertaintiy and establishing a stronger coach/teacher relationship.

When to Interject

Teachers are still developing their CS teaching skills and capacities. Sometimes the coach may have additional information or relevant CS Classroom Practice that could elevate the current learning opportunity.

If something (a fact, anecdote, or CS Classroom Practice) comes to mind that can greatly enhance the lesson, find a proper time to share it with the class, without disrupting the teacher's flow. This can be done by raising your hand to give the teacher the choice of whether or not to call on you, or sharing privately during student worktime.

Stick to Your Role

Your role is to be a guide and actively help the teacher grow. Challenging teaching experiences, coupled with feedback and reflection, will build teacher competence, confidence, agency and ownership.

Do not feel that you have to teach the lesson even if it is not going well. Provide the teacher with CS Teaching Practices and guide the teacher with advice and suggestions. Look at the unsuccessful lesson as an opportunity for the teacher's growth and development.

Positive Problem Solving

Modeling positive struggle in front of students and with students, when addressing a problem, impacts the student culture of collaborative problem solving in the classroom. It is okay to make mistakes in front of students.

When an error occurs with tech or in a program, interject to share your own joy in debugging and persevering through a tough problem. Express to the teacher and students how much you love to solve problems. Students' approaches and mindsets begin to change when the teacher models this behavior regularly and encourages students to joyfully debug using perseverance and peer collaboration.

Bug-In-The-Ear

When teachers are prompted in real time with practical strategies, they are likely to implement them in the moment and in the future.

Prompt the teacher discreetly during the class period with a CS Teaching Practice that they can enact in the moment.

Coaching: Co-Teaching [Interventions]

Intervene

Building deep CS understanding takes time. Intervening when a teacher is explaining a concept incorrectly is important to ensure concepts are learned correctly.

When necessary, politely intervene in a way that you and the teacher agreed upon during co-planning. Use light sentence openers, such as “Ms/Mr teacher, I’ve seen that done.. or do you mean...?” Address the students and teacher by decomposing the concept in a manner that clarifies it well for both the students and teacher.

Coaching: Co-Teaching

Look Fors

tech.cornell.edu/impact/k-12

Look fors are teacher and student actions that delineate effective teaching and learning of K-12 computer science. Coaches should watch for, identify, and later reflect on these practices seen during co-teaching.

Look For: Classroom Culture

A CS classroom culture embraces mistakes and develops a positive problem solving mindset. It should emphasize collaboration and celebrate innovation.

Look for evidence of a classroom culture that supports CS attitudes and dispositions. How are they fostered? Is there room for collaboration, mistakes, perseverance, joy in problem solving? Look for procedures, rules and norms and how they affect learning.

Look For: Productive Struggle

Struggle is productive when it is intentionally scaffolded and does not push students into frustration and stress levels that interfere with learning. It supports self-efficacy and reinforces problem solving skills. It breaks down learned helplessness behaviors and facilitates deeper learning.

Does the teacher tend to solve students' problems? Are students given time and resources to problem solve before receiving help? Is the teacher supporting students by prompting them to describe their thinking, try out different solutions, and tolerate ambiguity? Is the lesson designed with appropriate rigor, with options for diffeerentiation if needed? Are students disengaged due to boredom or frustration?

Look For: Collaborative Learning

Collaborating is a core practice of CS, ensuring different perspectives and inclusiveness. Teachers may be new to or need support to implement collaborative learning structures.

Look for teacher modeling and building collaborative learning norms in the classroom. Are students taking turns talking about CS content? Are all students having input into the CS project they are working on? Does the teacher provide a clear structure for collaborative work? Is there fair grouping? Are there roles? Does the teacher model accountable talk? Does the teacher provide access and support to collaborative groups for emerging bilinguals and students with disabilities?

Look For: Student Engagement

Student engagement describes the degree to which all students are actively participating in their own learning process. Students learn more when they are actively engaged in learning. High active engagement builds excitement in the classroom, drives student interest, and motivates them to advance their learning.

How does a teacher engage every student in learning a new CS concept? Does the teacher provide multiple entry points? Is the lesson teacher focused or student focused? How does the teacher solicit student participation? Are all students on task? Do students share their thinking, ask questions, work collaboratively? Do students share out, write in journals, complete exit slips to self-report their learning and level of engagement?

Look For: Student Thinking

Students' understanding of concepts becomes clearer when given the opportunity to show their thinking in different ways. Making student thinking visible informs both the student experience and teaching practice.

Do students have opportunities to explain their thinking verbally, in writing, or through drawings? Is the teacher prompting students to identify the CS concept they are using and how? Is there time for students to reflect at the end of the lesson?

Look For: Student Talk

Collaboration and discussion enhance student learning. Modeling and reinforcing CS and CT vocabulary reinforces student understanding. Reflection helps students articulate their work and share what they've learned with one another.

Are students given the opportunity to interact with each other during the lesson? Are they using appropriate CS vocabulary, sharing their thinking about the work, giving helpful feedback to each other, collaborating towards a common goal?

Look For:
Content Accuracy

Quality CS education happens when teachers combine accurate content and appropriate pedagogy.

Listen to the content shared with students by the teacher. Is the content age appropriate? Is CS vocabulary incorporated into instruction and used correctly? Are concepts clear, or misleading in ways that students will need to unlearn? Is the context correct? Is there any important information missing? Are teachers addressing common misconceptions? Does the teacher include too much technical detail, detracting from the learning objectives?

Look For: Extended Learning

Students who complete the requirements of an activity should be encouraged to stay engaged. What opportunties are presented when students complete an assignment before their peers?

Look for protocols that offer differentiated options for once an activity is completed. These may include additional practice, deeper dives into content, or opportunities for creativity or personalization.

Look For:
Clear Instructions

Clear, explicit instructions set the stage for success in CS learning activities. In addition, how instructions are communicated to students can support or hinder CS learning.

Are instructions comprehensible to the students in the class (language, images, format, multiple modalities)? Are instructions broken down into steps (appropriate to the grade level)? Are instructions available to students to refer back to throughout the activity? Do students understand what is expected of them? Are students able to follow the instructions and complete the steps?

Look For: The Third CS Teacher

The "Third Teacher" is the physical or digital space in which the students learn. The walls of a classroom impact student relatedness/ connectedness to the field of computing and act as a resource to support CS learning.

Look for CS learning supports around the space such as word walls, anchor charts, visual cues, posters, and books. Look for how the physical space is set up to leverage CS experiences for the students. Look for equitable representation visible in the room.

Look For: Student Voice and Choice

Throughout the lesson, there should be opportunities for student voice, meaningful choices, and personal expression.

Make sure that student agency (choice and voice) is built into each lesson to foster engagement and independent learning. Do students have the opportunity to provide feedback and input and ask questions? Do students have the opportunity to make important and meaningful choices? Are there opportunities for personalization?

Reflection & Feedback

tech.cornell.edu/impact/k-12

Reflection and feedback is a time for coach and teacher to go back over the instruction, surface wins and challenges, strategize alternatives, and gather lessons learned. They use these shared observations and ideas to collaboratively design a set of actions or next steps to take into co-planning or the lesson.

Plan the Debrief

Go into a debfrief prepared and ready to get started with pre-established goals and topics for reflection and feedback.

Be prepared to reflect on the goals for the Co-Teaching session, established during Co-Planning. Was the teacher able to achieve their goals for the session? Allow the teacher to share their thoughts. Ask them to highlight their successes and where they see areas for improvement. Give yourself time for providing feedback and discussing next steps.

Opening a Debrief Session

Debriefing, or reflection, is critical to the coaching cycle and should begin with teacher reflection. Inviting teachers to unpack first prioritizes their learning and makes sure these thoughts are not swayed by the coach's feedback.

Initiate a debrief session with self-reflective questioning. Give teachers a lot of time to think about and talk out what happened during the lesson without jumping to solutions. Ask questions like: How do you feel the lesson went? What do you think went well? What happened during each part? How are you feeling about the students' understanding?

Write It Down

When used as a communication tool, shared notes can be referred back to and can provide continuity and transparency for coaching sessions.

Create a shared document where you and the teacher record goals, action items, teacher takeaways and any feedback. This is a separate document from your coaching notes or your raw observation notes.

Summarize the Session

Keeping track of a reflection and feedback session via notes, a journal, or other method provides an opportunity for coaches to reflect on their own practice.

Keep detailed notes of your reflection and feedback sessions. Highlight the topics of conversation, how they were approached by both the coach and teacher, and any outcomes and responses that stemmed out from the conversations.

Closing a Debrief Session

Closing questions help to solidify and confirm teacher understanding.

Close a debrief session with self-reflective questioning. Check for understanding using questions: What did you learn from our conversation? What are some big takeaways? What would you do differently next time? Be sure to celebrate teacher accomplishmets as you close the session.

Coaching Report

Since coaching may take a significant investment of time and resources for schools, some administrators may want to keep a record of the coaching program. A coaching report provides insight into the process and approaches a coach has taken for developing teachers' CS education skills.

Create a short write-up that states the coaching activities the teacher participated in and the topics that you and the teacher discussed without any element of evaluation. Create this document with the teacher or send it to the teacher to check over before sharing it with administrators. Alternatively, send a weekly email outlining who you worked with, general themes, and student "wins" from the week.

Student Work

Bringing student artifacts into the reflection process gives coaches and teachers a way to check for student meaning.

Look for patterns in the work from all students or examine a sample of students' work from all levels, looking for areas of strength, suprises, and any common misconceptions. The results of this should inform future planning.

Plan to Iterate

Sometimes a solid plan for a lesson does not go as we hoped. Emphasize that that's okay. Teachers should be prepared to iterate based on their reflections

When a lesson didn’t go as planned, discuss what went wrong and make a plan to revise, reteach, or modify. Discuss and think of the students who might have struggled with this lesson. What worked well for them? What didn’t?

CS Teacher Growth

CS teacher standards are a valuable resource for teachers and coaches in developing CS teachers. Self monitoring using standards promotes reflection and goal setting.

tech.cornell.edu/impact/k-12

Introduce, review and reflect on the CSTA CS Teacher Standards or state CS standards. Together, help the teacher to assess their growth, or encourage them to complete a self-assessment and share the results with you.

Action Steps

Defining a specific action is a key component of coaching and leads to ensuring continuous improvement.

Using clear language, identify actions that should follow the debrief. Both the teacher and coach should leave the session with specific steps to take prior to the next lesson or session.

The Teacher's Perspective

A coach should not assume that a teacher experienced or perceived a lesson or classroom instruction exactly as the coach did.

Before providing feedback as a coach, try to see the lesson from the teacher's perspective by asking them questions specific to things that you noticed. For example, instead of pointing out to a teacher that you saw them only working with a select group of students, ask them, "how did you support every student in your class today? Were you able to work with every group? Why or why not?" This line of questioning goes deep into the issue without trying to guess what the teacher's reasoning might have been.

Giving Feedback

Feedback is a powerful tool for learning when it is provided in as detailed and objective a way as possible.

From coaching notes, describe important moments from the lesson to highlight for learning. Use as much detail as possible, without use terms of judgement. Ask the teacher to reflect on these moments to gain their perspective and inform next steps.

Teacher Talk > Coach Talk

Learning and growth occurs from active participation and conversation. Teachers should be talking more than coaches during debrief.

Pay attention to talk time. Who is speaking more? Create room for the teacher’s voice by asking open-ended questions and allowing think time. Prompt deeper thought with short follow up questions like, “Can you say more about that?” The teacher should be engaging in thinking work with you as a guide.

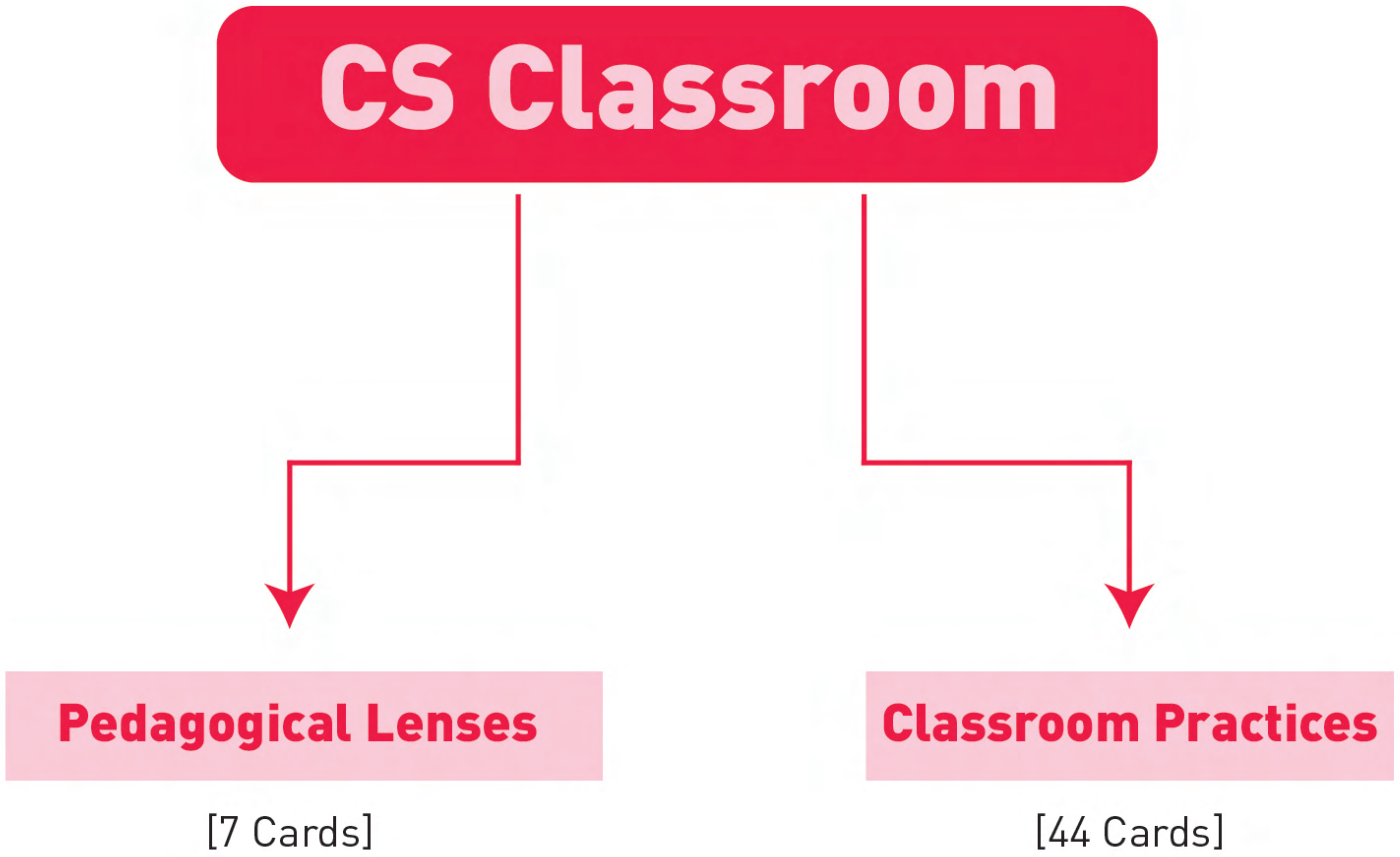
Recognition & Praise

Explicit, targeted praise can build confidence and reinforce new practices.

Praise the specific effort(s) that got teachers to the result. Recognize them for the work they put into this accomplishment. Don’t overpraise: be targeted and realistic. Notice and identify something obscure that they may not commonly hear. Let your praise stand alone, not connected to constructive criticism (this was good but…). Praise teachers to their colleagues and supervisors.

CS Classroom

CS Classroom cards include proven approaches to promoting CS engagement and learning. They include Pedagogical Lenses, which define overall multiple, well-researched approaches to teaching, and Classroom Practices, which offers concrete, detailed instructional strategies.


CS Classroom
Pedagogical Lenses
[7 Cards]
Classroom Practices
[44 Cards]

CS Classroom

Pedagogical Lenses

tech.cornell.edu/impact/k-12

Pedagogical Lenses provide coaches with overall frameworks through which to view teachers' Classroom Practices and CS lesson facilitations. They can also be shared with teachers so they may reflect on their own teaching practice to inspire or informa areas for further development or growth.

Culturally Responsive Education

Culturally responsive-sustaining computer science pedagogy ensures that students' interests, identities, and cultures are embraced and validated, students develop knowledge of computing content and its utility in the world, strong CS identities are developed, and students engage in larger socio-political critiques about technology's purpose, potential, and impact.

tech.cornell.edu/impact/k-12

Offer additional resources and training in CRE to help teachers examine their own cultural context and how it informs their teaching. Discuss teachers' knowledge of the school community and of individual student's cultures. If needed, provide ideas for how to deepen that understanding. Throughout the coaching cycle, have conversations with teachers about connections between CS content and students' community and prior knowledge. Ask questions that provoke reflection and iteration around the classroom environment and instruction and how they relate to culture, and encourage teachers to do the same with their students. (Brown University)

Universal Design for Learning

Universal Design for Learning (UDL) is a way of thinking about teaching and a framework for instructional design that improves and optimizes learning for all students. UDL is informed by neuroscience and supported by educational research. It is an inclusive approach that is beneficial to all students, including students with disabilities.

Coaches can model a UDL approach into their coaching practice. They can integrate UDL processes and approaches into support materials and coaching conversations, for example highlighting ideas that UDL focuses on like planning for variability and developing students as expert learners. This helps teachers to build their capacity for UDL over time, in practical ways. Coaches can also focus on supporting the application of the UDL guidelines to CS instruction. (CAST)

Translanguaging

Translanguaging in the classroom is a pedagogical approach whereby multilingual students are encouraged to use their languages as an integrated communication system to make meaning, learn, and express themselves. Use of translanguaging in the classroom recognizes the value of linguistic diversity and fosters deeper understanding and more meaningful expression of learning from bi/multilingual students. It also supports students in acquiring and comprehending English while maintaining their native language.

Coaches work with teachers to find opportunities to encourage students to work and represent in the language with which they are most comfortable. Coaches walk through this process with teachers and support them where they need help. Offer additional resources and training in translanguaging to the teacher. (O. Garcia & L. Wei) (PiLa-CS)

Inquiry-Based Learning

Inquiry-based learning is an approach that encourages students to construct their own knowledge through teacher-guided exploration and questioning. An inquiry-based learning approach is key to developing computational thinking and leads to higher order thinking and deeper levels of learning.

Model inquiry-based strategies for teachers throughout the coaching cycle, and support their implementation in the classroom. Ways to build a teacher’s skillset in this area include: model the teaching of an inquiry-based learning activity, point to frameworks for inquiry-based learning like POGIL, work with the teacher to find new opportunities to incorporate inquiry, discuss the ideas of teachers as experienced co-learners, and strengthen the teacher’s CS content knowledge.

Project-Based Learning

Project-based learning (PBL) is a student-centered approach to teaching that allows students to learn problem-solving techniques and subject content through the exploration of real-world challenges. Through PBL students solve big, authentic problems by collaborating with peers and using resources and knowledge to develop a solution. This approach lends itself to building students' proficiency in all seven of the core CS Practices described in the K-12 CS Framework and also aligns to computational thinking frameworks.

Support a teacher through the entire PBL process from planning and implementation to assessment and iteration. Help the teacher to identify driving questions in the field of CS, and push the teacher to make projects as meaningfully connected to real world problems as possible. Connect the teacher to community members and resources that can support the project. Provide examples of rubrics and materials for scaffolding student project management.

Pedagogical Lenses [CS Classroom]

Computational Thinking

"Computational thinking (CT) offers an encompassing approach to problem solving that exposes students to computing ideas and principles in the context of the subject areas they are already learning." Aman Yadav, Hai Hong, and Chris Stephenson.

tech.cornell.edu/impact/k-12

Implementing and developing CT content and pedagogical knowledge requires explicit modeling, repetition, and classroom practice. While definitions differ, there are four concepts of CT that all students should learn and use. They are: abstraction (removing unnecessary detail), algorithms (making steps and rules), decomposition (breaking problems down into parts), and pattern recognition (spotting and using similarities). CT is ideal for subject matter integration. Professional development for CT should focus on the four key concepts of CT, and how they might provide new problem solving strategies for teachers and students throughout the curriculum.

Abolitionist Pedagogy

Abolitionist pedagogy describes an approach that humanizes all children and seeks to help all children thrive in school rather than just survive.

Educators who subscribe to abolitionist teaching use a whole child approach. They are proactively anti-racist, and work to create learning environments and learning experiences where children's intersectional identities are validated and their social and emotional well-being is considered along with their learning.

Classroom Practices

Classroom Practices are a collection of strategies, best practices, and moves that offer teachers concrete detailed approaches that they may use when directly facilitating and instructing students in a CS lesson.

Getting Unstuck

Students need strategies to attack a problem when they get stuck. Explicit strategies for problem solving help students regulate their emotions, continue their learning and persist.

Encourage students to step away from the problem and take a short brain break. Suggest they collaborate with a peer to problem solve, describing what they want their program to do and where they think they're stuck, and reading the code aloud. Remind students to collaborate, tinker, debug and perservere.

Asking For Help

Teachers should model and practice using built-in tools and outside resources to find information and help. Students need a variety of strategies for solving problems. CS classrooms need routines for asking for help.

Teach explicit strategies for monitoring and checking work. Create a routine visual signal for students to use when they need to ask for help: a specific hand signal, writing their name on a whiteboard, or the use of cue cards, colored sticky notes, cups, traffic light cards, or emojis will help students alert you to their need for help and regulate their frustration.

Think Time

Giving students purposeful time to reflect before answering your questions provides all learners with time to process your question and formulate an answer.

Develop a thinking culture in your classroom by adopting Think Time routines. Wait 15-30 seconds before calling on a student to answer your questions, and encourage students to keep their hands down and ponder during that time. Encourage thoughtful conversation and reflection with routines like “Think, Pair, Share”, “Everybody Writes”, “Predict, Observe, Explain (POE)”.

The Power of Paraphrasing

Students' understanding of CT and CS concepts may be quickly checked by asking a student to say, in their own words, what they are expected to do.

Check for understanding. Have students repeat back their understanding of a CT or CS concept or instructions for a task. If one student doesn't have a clear understanding of what to do, there is a chance that many do not. One way to do this is to ask for a volunteer to explain the concept or instruction to the class. Be sure to make any corrections respectfully.

The Hook

A short, fun activity in the beginning of a lesson helps students transition and improves motivation by engaging in fun experiences related to the learning objectives.

Start a lesson with an engaging activity that last a few minutes. Hooks may activate prior knowledge or be used to preview new concepts.

Classroom Practices [CS Classroom]

Student Deliverables

Routines and tools for checking lesson goals and monitoring progress builds learners’ independence and understanding.

Make student expectations clear by visibly posting the deliverable of a lesson on the whiteboard and assignment sheet, along with a rubric or checklist. Refer to the written expectations when a student asks what they are supposed to do. Build time into the project for students to self-assess and iterate on their work.

Pseudocode

Pseudocode is a language that represents concepts across programming languages, but cannot actually be run by a computer. It helps students formulate and organize their thoughts before starting to write code. It removes the barrier of needing to know exact coding syntax.

Have students use a combination of natural language and programming language to sequence out an algorithm when planning a program. Students can express their understanding of a program by writing pseudocode for that program. Psuedocode is a good context for multilingual students to do “thinking work” in the language of their choice.

Worked Examples

Worked examples help students conceptualize the procedural nature of problem solving and program development within a meaningful context.

Worked examples typically include a problem to be solved, step-by-step instructions for developing a solution, and the solution itself (a block of code or complete program). Worked examples can be examined as a class to introduce new concepts or used as a reference for solving similar problems. You can also provide students with incomplete worked examples and have them fill in the missing information.

Commenting

Commenting is writing non-executable notes directly into a program. A form of collaboration, it is widely used in computing. It helps students organize their program and is a way students can communicate their thinking to their teacher and partners on a team project.

Set expectations for students to document their thinking by commenting within their code. Ensure that you are teaching and modeling commenting during instruction.

Lingering Questions

Students often have unasked questions at the end of your lesson or instructions. A routine of checking for any questions before sending students off to work will avoid misunderstandings and wasted time later.

To elicit students' lingering questions, try asking, "What questions do you have?", or directing all students to write down one question anonymously and drawing questions to answer yourself, or have other students answer.

Least to Most Prompting

In CS, there is a tendency for teachers to over-prompt and over-scaffold. Starting with minimal help and increasing your support helps you understand what your students really need.

When a student first asks for help, refer them to classroom resources, respond with a question, or give them a generalized prompt. If the student continues to ask for help, gradually increase the amount of help you give, saving things like direct instructions as a last resort. (M. Israel, et al.)

Facilitating Student Talk

Students must learn to listen and learn from each other through gentle reminders from you. If the teacher paraphrases, the class will lose accountability for listening and thinking on their own.

When you ask a student to answer a question in front of other students, do not repeat or rephrase their response. If it needs to be clarified or repeated have another student rephrase a prior students answer. Teach students a routine of “add on,” “build on,” or “yes, and” to clarify and deepen the prior student’s response.

Capturing Students' Thoughts

Time constraints, pacing, or class momentum can unintentionally silence student contributions. Creating alternative ways to contribute helps validate their ideas.

Seek opportunities to explore a student’s thinking through journaling, prompts, and in large or small groups to accomplish two goals: 1) acknowledge that the student’s idea is valued and 2) allow students to contribute to the refining of each other’s ideas and reason collaboratively.

Heads Up

It can be difficult for anyone to pull their attention away from a project, especially on a screen. Create a friendly protocol as a management tool for moving students from screen work to listening to the teacher.

Create a signal for students to shift their attention to the classroom. Physically adjusting the device or screen can help reduce distraction. Students may turn a tablet or phone face down on the desk or put laptop screens at a 45 degree angle known as “clamshell”, “Pac-Man”, or “half mast”. Students can also do something else with their hands like pick up a pencil, respond in sign language, or do a “fist to five” check-in.

Hands Behind Back

When teachers touch student work by editing their code, taking control of their keyboard, writing on their work, or even physically leaning into their workspace students tend to back away from ownership of their work.

Teachers should form a habit of clasping their hands behind their back when they observe students during worktime so as not to touch student work. Teachers can also instruct students to use this strategy when they are providing peer assistance or participating in pair programming. Guidelines like “Only the programmer touches their own keyboard and mouse” can support this idea.

Scaffolded Constructionism

Based on Papert's research, creating physical and digital products helps students construct their knowledge about CS. We can give all students access to this type of rigorous experience by building scaffolds to independent learning.

Create guidance that helps students build their own knowledge through interaction with computing, both plugged and unplugged. Gradually reduce these supports as a student's ability grows. Create opportunities for students to construct their own knowledge through open-ended play with physical materials and creating both digital and physical artifacts. Make these experiences accessible to all students by offering opportunities for explicit instruction on specific skills or topics, by scaffolding exploration and artifact development, and by guiding synthesis and reflection. (M. Ray, M. Israel, et al.)

That Could Be Me

Identity stereotypes, implicit biases, and systemic barriers to access and inclusion influence the lack of representation in tech and some students' interest in pursuing advanced education in CS. Students should be encouraged to see themselves as part of a CS community and have opportunities to engage with personally meaningful CS projects.

Dedicate class time to show current news of diverse professionals and students applying CS in multiple fields. Confront stereotypes of “who is a computer scientist” is by representing professionals including people of color, LGBTQ+ people, women, and people with disabilities in learning materials, storytelling and bulletin boards. Highlight technological advances in fields that show diverse teams collaborating on difficult problems such as disease detection, farming, artistic productions, natural disasters, etc. Feature historical triumphs of historically excluded groups whose contributions are often invisible in most curricula. Audit your classroom for images that reflect representation on many levels.

Multiple Entry Points

Lessons designed with multiple entry points provide access and engagement with CS concepts based on each learner’s strengths and needs.

Offer options for engaging with new concepts based on student preferences and learner needs. For example, when starting a new project, students may choose from different options to start exploring the task. These might include: building a program with unordered code blocks or snippets, debugging a broken or incomplete version of the program, using resources like a worked example or video instructions to build the program on their own, or modifying a completed program and adding advanced features. (Creative Technology Research Lab, UF)

Create a Culture of CT

Using visual and nonverbal cues in the classroom promotes computational thinking as an everyday problem solving approach

Make CT visible in your classroom. Strategies include hanging Computational Thinking posters, CT-related student work and a CT word wall. Teachers can use and teach hand signals for CT concepts, such as two fists held together then moved apart to signal decomposition, as a Universal Design for Learning strategy.

Call and Response

Call and response routines in the classroom can prime students' for a learning activity. Creating CS-specific call and response chants can help reinforce concepts, information or actions. Call and response has its roots in the African diaspora created by slave trading.

Create and reinforce call and response routines to introduce facts from CS history (for instance, teacher: Ada, students: Lovelace), get their attention, or prepare them for a familiar task.

Celebrate CT Approaches

When teachers showcase student use of computational thinking approaches to problem solving (debugging, collaboration, perseverance, tinkering), they reinforce the mindset and work ethic students should have during CS lessons.

Display the approaches on a bulletin board or word wall so that students are continuously reminded of the behavioral mindsets they should be using. Celebrate the perseverance you see, the collaboration, creativity, and tinkering techniques. Consider acknowledging students who engage in CT practices through CT badges, exit tickets, or by asking students to share CT successes.

Classroom Practices [CS Classroom]

Solve Problems with CT

The tools of CT can help students prepare a problem to be solved by a computer, but can also foster deep thinking and collaboration to reach solutions outside the digital world. “Computational thinking allows us to take a complex problem, understand what the problem is and develop possible solutions. We can then present these solutions in a way that a computer, a human, or both, can understand.”

tech.cornell.edu/impact/k-12

Integrating CT in classroom lessons requires allocating time for students to construct their own knowledge of CT skills and practices. To develop student knowledge, teachers should identify, highlight or call out the CT skill (s) the students would be expected to practice in the lesson. During the lesson teacher inquiry prompts should include CT language, such as, which CT skills are you using to solve the problem? Do you notice any patterns in the authors writings, or patterns in a math problem you are solving? How might you decompose the problem into smaller steps to find a solution?

Modelling Use of CT/CS Vocabulary

Using the language of the discipline sets expectations and develops content knowledge in CS.

Model CT and CS vocabulary, even outside of formal CS lessons (I can't get the smartboard to work. Let's debug!). Use vocabulary-building strategies like word walls, frayer models, visuals, and Total Physical Response (using physical movement to react to verbal input). Reinforcing vocabulary is a great opportunity to bring in translanguaging strategies such as multilingual word walls, allowing students to come up with their own translations, and providing opportunities for semantic mapping and comparing root words and cognates.

Subgoal Labeling

Subgoal labeling is grouping related steps and naming them to describe the part (subgoal) that they achieve in a larger task. Subgoal labels have been used in computer science education to help students learn how problems are constructed and solved. They have also been used to reduce cognitive load when using worked examples.

Decompose worked examples into code chunks and add subgoal labels. Use code comments or visual text aids to define the purpose of each segment. After some practice deconstructing code and creating code in novel problems, give students a worked example and have them create subgoal labels.

Parson's Problems

Parson's Problems are lines or blocks of scrambled code that need to be correctly ordered. They promote logical thinking and reduce cognitive load by removing the need to remember, look up, or fix syntax. They can also be used for assessment.

Provide students with blocks or lines of code that are out of sequence. Challenge students to re-order the code so that the program runs as intended. This strategy may be applied in any programming language. (B. Ericson)

Instructional Tools

Introducing teachers to multiple digital and graphic resources helps them create accessible classrooms and learning experiences.

Explore options like digital or paper graphic organizers, vocabulary charts, dictionaries, visual task lists, flow charts, and many more. Check in on the specific needs of learners and brainstorm appropriate tools to give access and support.

Collaborative Culture

Collaborating around computing and communicating about computing are recognized core CS Practices and important learning objectives for all students.

Teach students procedures for discussions and collaborative work. Provide students with scaffolded supports that facilitate collaborative work. Examples of commonly used procedures include accountable talk and assigning group roles. Scaffolds for these examples could include sentence stems, role cards, and prompts for self-monitoring participation.
(K-12 CS Framework)

Kinesthetic Learning Strategies

Physical movement can provide an additional entry point to CS learning for students of all ages. It can also improve understanding, retention, social skills, behavior, and brain functioning.

Include options for students to engage in CS learning activities through movement and working with physical materials. Example strategies include: “acting out” code, planning with printed and cut out code blocks or snippets, incorporating robots and making activities, and using Total Physical Response to teach CS vocabulary. Support students who want to stand while writing code or collaborating (most tech workplaces feature standing desks).

Model Problem Solving

Problem solving is a skill that needs to be taught and developed. Coaches need to expose teachers to explicit problem-solving strategies in the context of CT and CS.

Explicitly teach problem-solving skills to students during CS instruction. Consider leveraging CT strategies to scaffold problem solving. Model the strategies using think alouds. Work through problem-solving examples collaboratively with a group of students. Help students identify where in a lesson they may need to use different approaches. Create anchor charts that list different strategies or outline the steps of specific approaches.

Reference Materials

Programming is not about memorizing specifics, it is about being able to solve a problem using a deep understanding of concepts that transfer between languages. Providing reference materials and teaching students how to use them reduces cognitive load, fosters independence, and helps students focus on concepts rather than the details.

Make reference materials readily available in the classroom. Materials could include cheat sheets, online forums, formal documentation, language reference libraries, and code editor autofill tools. It is more important that students know how to apply, access, and use reference materials for deeper understanding than memorizing code.

Exploded Code

Scrambled code encourages students to demonstrate their conceptual understanding by creating a program that they might not be ready to create from scratch.

Provide students with some or all of the code needed to complete a task, disassembled. In block-based languages, individual blocks are “exploded” around the code editor. In a text-based language, you might offer completed lines of code in a series of unordered comments. Students use the provided code blocks or snippets to complete a program. (Creative Technology Research Lab, UF)

Flowcharts

Understanding the flow of a program is an important step in learning. Creating flowcharts helps students focus and organize processes using words or images that are easily understood by them and/or their peers.

Use flowcharts to develop student understanding, thinking, and approach to problem solving. Have students use a combination of natural language and programming language to plan the flow of a program and sequence its different parts. Students can express their understanding of a working program by writing pseudocode for that program.

Reading Code

When students read code, they create a mental model of what will happen when a program runs. By reading code, students begin to notice patterns in style and syntax in and across programming languages. Reading code not only builds programming skills, but also strengthens reading comprehension. Teaching strategies for reading and investigating code provides scaffolding for programming fluency.

Have students practice reading code. Ask students to predict what will happen when the code is run. Have them translate it to pseudocode. Ask students to look for patterns, structure, and syntax. Have them identify the purpose of specified blocks of code within a program. This should be a student-led experience: avoid reading code to students except to model.

TIPP & SEE

Designed to be used with Scratch, TIPP & SEE is a metacognitive strategy used in conjunction with Use, Modify, Create while learning to program. Students using the TIPP & SEE strategy have been shown to use more programming elements and write longer programs.

tech.cornell.edu/impact/k-12

Students preview a project using TIPP: examine the Title, read the author's Instructions and Purpose, then Play the project. Next, students apply the 'SEE' strategy: See the project code, Examine events and then Explore the project by making modifications. TIPP & SEE can be adapted for other platforms that allow program sharing and remixing.
(J. Salac, et al.)

Curriculum Mash-Up

Some CS curricular resources are stronger in some components and weaker in others. Combining different CS resources and pieces of curricula can create solid CS lessons and units.

Co-plan a “mash-up” lesson or unit together that combines lessons from different CS resources and/or curricula. Bring in multiple sources addressing the same concepts and have the teacher talk through their choices for what they want to incorporate, at what point in the unit or lesson, and why.

Classroom Practices [CS Classroom]

Concrete-Representational-Abstract (CRA)

CRA is an evidence-based strategy in math education for teaching abstract concepts represented in mathematical expressions. It can be applied in computer science to teaching abstract concepts represented in programming languages.

tech.cornell.edu/impact/k-12

When introducing abstract computing concepts, begin with a concrete activity using manipulatives or movement. Next, move to a learning activity that is representational involving students drawing out the concept or using pseudocode. Finally, help the students apply their understanding to an abstract programming environment.

Peer Feedback

Feedback between peers gives students an opportunity to practice communicating about computing. Receiving and applying feedback helps students build collegial relationships. Feedback during user testing and iteration are core to CS.

Design a peer feedback protocol to teach and develop students' ability to communicate and constructively evaluate each other's progress on a project. Feedback should be planned in the context of a project so that students can take action on their project to incorporate suggestions. Consider having students use an assignment rubric to review as they assess and give each other feedback.

PRIMM

PRIMM, (Predict, Run, Investigate, Modify, Make) is a protocol for learning programming concepts. The guided process of PRIMM allows students to deepen their learning and interact with completed programs when they are still beginners.

Have students read a program, predict what it will do and converse about their predictions. After they run the program, students should discuss the outcome. Students then participate in a guided investigation of the code prior to tackling modification task challenges. Finally, have students create their own program. (S. Sentance)

Concept Before Code Protocol

Abstract coding concepts are disconnected and irrelevant to learners when there is no context for the code they are using. Conceptual understanding can be improved through exposing students to CS concepts in a separate activity, prior to coding, that connects directly with their experience or prior knowledge.

tech.cornell.edu/impact/k-12

Teach new coding concepts prior to introducing them in a program.
1) Define the concept.
2) Make an analogy that will relate to students' background knowledge.
3) Engage students with concepts in unplugged or digital activities prior to coding.
4) Show the students code that reflects the new concept and make explicit connections to the prior activity.
(S. Grover, et al.) (P. Bagge)

Use-Modify-Create

Use - Modify - Create is a three stage progression used to engage students in CS learning. Reviewing and using existing code then modifying it to change its outcome before writing code from scratch has been demonstrated to increase confidence and reduce anxiety.

First have students interact with a finished program (use). Next, challenge students to make changes to (remix) the provided program (modify). Finally, have students create their own program using the provided program as a model (create). This strategy can be implemented independently or collaboratively. Provide guiding questions and prompts to facilitate each activity. (I. Lee, et. al.)

Tech Helpers

A system in which students take on different responsibilities during CS lessons helps build a collaborative classroom and a sense of community while reducing teacher overload and learned helplessness.

Create a student team to assist with devices, materials and helping other students. Provide guidelines and scaffolding to student helpers to prevent them from doing the work for others. Rotate students to allow all students the opportunity to take on a tech helper role. Post the current team of assistants on the board and/or in a handout.

Tracing Code

By tracing code, students will explore states of a program while executing, to learn to comprehend changes in program variables over time and build their debugging skills when issues arise.

Make sure that students first read the problem they are solving. Next, ask students to find where the program begins. Start tracing each line, conceptualizing what is being run one line at a time. Have students use a pen and paper to record values of code variables as code executes. Using trace tables is a strategy that helps students conceptualize the changes to variable values during code execution.

Zone of Proximal Development (ZPD)

The ZPD is the space between what a student can do on their own and with support from a teacher or peers. To maximize learning and engagement, problems should not be overly complex, causing some individual students to become overwhelmed or too simple, causing boredom. Threading this needle helps students develop expertise and independence.

1. Activate prior knowledge: Connect new learning to prior learning to build relevance and bridge understanding.
2. Build prerequisite knowledge: Create opportunities for students to build the content- or context-specific knowledge needed for access and understanding.
3. Try mini lessons: Break learning into smaller, more manageable chunks that build progressively in complexity.
4. Use visual, sensory, or auditory aids: Support multiple modes of learning using models, videos, artifacts, experiences, recordings, and read-alouds to reach students in different ways.
5. Have some Think-Write-Talk time: Provide low-stakes opportunities for students to process through thinking, writing, speaking, and listening, both independently and collaboratively.
6. Include intentional practice: Gradually release students to successive levels of independence through the I Do, We Do, You Do model.
7. Introduce increasingly complex questions or tasks: Design and order tasks and questions to build from simple to complex in a way that helps students gain just-in-time insights or skills.