Computational Thinking and Computer Science in Early Childhood

In recent years, researchers, policymakers, and educators have increasingly recognized the importance of supporting young children's development of computational thinking (CT) skills, practices, and processes. CT is a universal way of thinking that is relevant for children's learning, with many real-world applications and meaningful connections to other content areas, such as math and science. CT skills also serve as a foundation for learning in computer science (CS). As such, national and state education systems—including California—have published standards and frameworks focused on integrating CT and CS into early learning (California State Board of Education, 2018; K–12 Computer Science Framework, 2016).

In response to the growing need to support educators and families with implementing CT and CS in early childhood, this brief was developed by WestEd, in collaboration with partners, as part of the Count Play Explore (CPE) initiative, a statewide, multi-agency effort led by the Office of the Fresno County Superintendent of Schools, Early Care and Education Department. CPE aims to raise awareness of the importance of early math, science, and CS, and to build the capacity of educators and families to support children's early learning and development.

This brief addresses the development of CT and CS in children from birth through third grade. It begins by defining CT and CS and provides an overview of key CT skills that are developmentally appropriate for young children. It then summarizes research-based strategies for how educators can incorporate CT and CS into their early learning settings and participate in high-quality CT and CS professional learning (PL) experiences. Finally, the brief concludes with key takeaways and recommendations for PL facilitators, trainers, and coaches as they implement CT and CS learning experiences with children, educators, and families in their local communities.

Defining Computational Thinking and Computer Science

Computational thinking (CT) is a set of critical thinking and problem-solving skills, practices, and processes that everyone, including computer scientists, can learn and use in various contexts (Bers, 2018; Grover & Pea, 2013; Wing, 2006). CT involves skills such as following step-by-step instructions to complete a task, recognizing patterns, and breaking down complex tasks into smaller steps.

Consider these everyday examples that involve CT:

- You might accomplish a complex task, such as planning a birthday party, by breaking it down into smaller tasks, such as sending

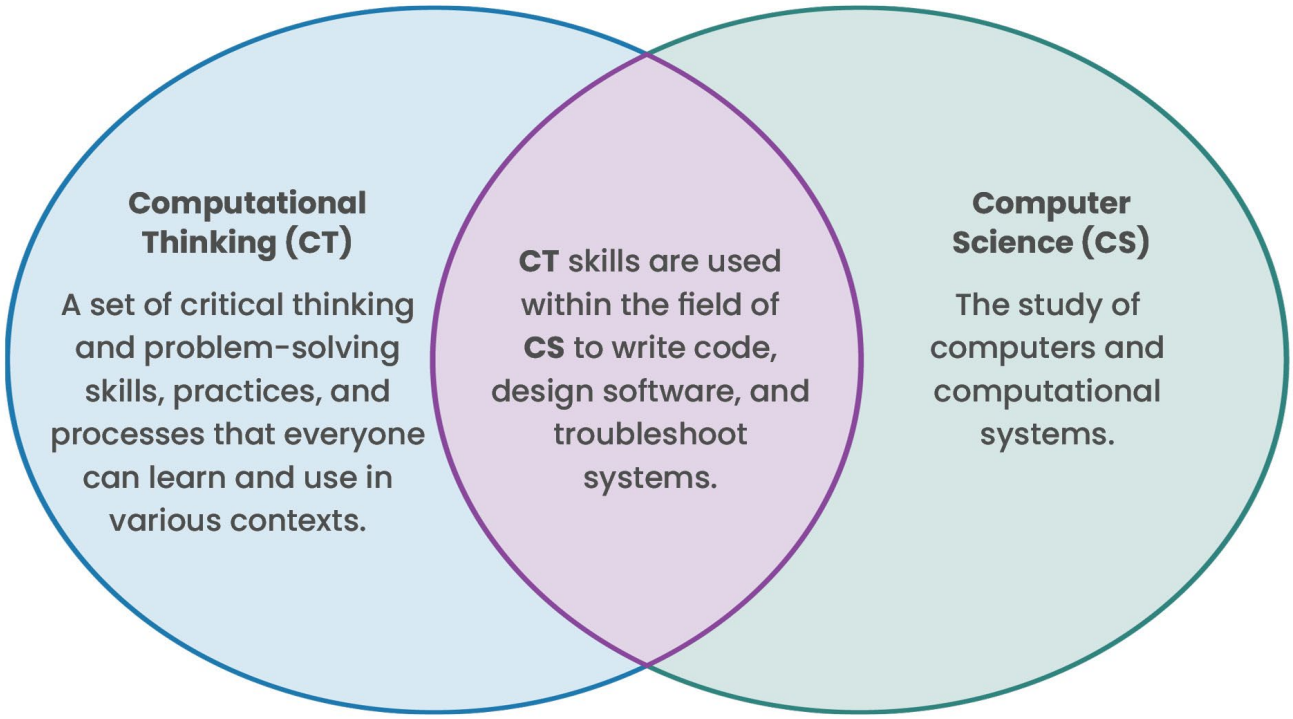
out invitations, ordering cake, and putting up balloons. **(decomposition)**

- If you misplace something, you might retrace your steps to figure out where and when you last had it. **(debugging)**
- You might systematically plan your route at the grocery store by getting your fruits and vegetables first, then dry goods, and finally dairy products. **(sequencing)**

Computer science (CS) is the study of computers and computational systems (Bers et al., 2022). Computer scientists are people who study computer algorithms and processes and use technology to solve real-world problems. They perform a variety of tasks, such as writing and running computer code, developing and testing software, and troubleshooting and improving computer systems (California State Board of Education, 2018; Google Inc. & Gallup Inc., 2016a, 2016b; Garfield County School District, 2023). All of these tasks require foundational CT skills and advanced coding skills. However, children as young as three years old can learn how to code and engage in CS learning. For instance, they may use color-coded cards or blocks that serve as coding instructions to make a robot move and make sounds (Monteiro et al., 2021; Bers, 2018).

Understanding the overlap and difference between CT and CS is essential for supporting children's early learning experiences (see figure 1). Not every child will eventually become a computer scientist. However, everyone—including children—can learn to think critically, design systems, and solve real-world problems that draw upon fundamental CT skills, practices, and processes.

Figure 1. Defining Computational Thinking and Computer Science



Key CT Skills in Early Childhood

Over the years, CS education researchers have developed frameworks to describe how young children engage in CT (for example, Barr & Stephenson, 2011; Bers, 2018; Grover & Pea, 2013). Although the frameworks differ in their language and scope, they all target similar aspects of CT. For example, the K–12 Computer Science Framework (2016) focuses on patterns, problem-solving, representation, and sequencing. Bers's (2018) "seven powerful ideas of CT" uses different vocabulary yet emphasizes similar skills and practices. For instance, sequencing is referred to as "algorithms."

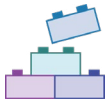
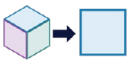
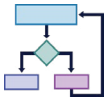
Despite the variation in the frameworks, there are six CT skills that are consistently mentioned in the literature (refer to table 1 for definitions and real-world examples):

- Algorithms and Sequencing
- Pattern Recognition
- Decomposition and Modularity
- Abstraction and Representation
- Testing and Debugging
- Conditional Logic and Loops

CT involves critical thinking and problem-solving skills that everyone, including computer scientists, can benefit from learning and using.

Table 1. Key Computational Thinking Skills in Early Childhood

CT Skill	Examples in Everyday Life
 **Algorithms and Sequencing** • **Algorithm:** set of step-by-step instructions required to solve a problem or complete a task • **Sequencing:** act of arranging objects or instructions in a specific order	• A step-by-step morning routine **(algorithm)** • Identifying the order of steps for brushing your teeth **(sequencing)**: (1) Squeeze toothpaste on your toothbrush. (2) Hold the toothbrush under the tap. (3) Brush your teeth for two minutes. (4) Spit the toothpaste into the sink. (5) Rinse your mouth with water.
 **Pattern Recognition** Identifying or creating a sequence of repeating items, such as objects, shapes, and numbers	• Making a bracelet with repeating color of beads (for example, following an AB pattern) • Singing and clapping "Pat-a-Cake" with the educator, repeating a clapping pattern

CT Skill	Examples in Everyday Life
 **Decomposition and Modularity** • **Decomposition:** breaking down a large problem or task into multiple smaller steps • **Modularity:** combining smaller steps, or modules, in different ways to solve a problem or complete a task	• Breaking down all the steps needed to clean up a messy room before deciding what to tackle first **(decomposition)**: trash needs to be thrown away, toys should be picked up, floor needs to be vacuumed, and so forth • Setting a table for six children for snack time by assembling one place setting first with a plate, cup, and spoon and then repeating this setup for the other five children **(modularity)**
 **Abstraction and Representation** • **Abstraction:** extracting key information about a complex object or concept • **Representation:** displaying or recognizing an object or idea using a symbol	• Finding a banana by searching for a yellow fruit shaped like a C **(abstraction)** • Identifying a red octagonal sign on the street **(abstraction)** and recognizing that the sign means to stop **(representation)**
 **Testing and Debugging** • **Testing:** checking if something works the way you want it to • **Debugging:** finding an error and using problem-solving strategies to fix the error	• Checking whether a flashlight works, identifying that it does not work because it is missing batteries **(debugging)**, replacing the batteries, and testing again to make sure the flashlight works **(testing)** • Realizing a puzzle piece is not in the right place and turning it in different orientations to figure out where the piece fits **(debugging)**
 **Conditional Logic and Loops** • **Conditional Logic:** using if-then statements to make decisions about what to do next or to control the flow of an algorithm • **Loop:** a recurring set of steps used to determine if and how an action is repeated	• Making decisions about what to wear based on the weather; for example, "IF it's sunny, THEN I will wear shorts" or "IF it's raining, THEN I will bring my umbrella" **(conditional logic)** • Developing a dance routine consisting of three claps and a spin, and then repeating the sequence of actions two more times **(loops)**

How Children Learn and Develop CT and CS Skills

The development of CT begins early in life as children explore their environment and make sense of the world around them. Infants and toddlers begin to identify patterns and repeated structures (for example, a daily bedtime routine) and think in a conditional way (for example, "If I push this button, then it will make a noise"). As children develop, they engage in critical thinking and reasoning as part of their everyday routines and play. They learn to solve more complex problems and follow step-by-step instructions to complete tasks. For example, four-year-olds may sing a song to remember the steps for tying their shoelaces.

Although children develop CT skills through everyday learning experiences, there are benefits to targeted exposure to CS learning experiences. Emerging research indicates that early exposure to CS can support children to develop positive mindsets about computing. For example, a study with elementary children ages 7 to 10 found that participation in a robotics and coding afterschool program increased children's understanding of CS and robotics, and it increased their motivation to pursue a CS-related career (Ragusa & Leung, 2023). Early exposure to CS can also disrupt negative stereotypes that children may hold about who computer scientists are and what they do (Hansen et al., 2017; De Wit et al., 2021; Master et al., 2021). For example, many studies have found that positive experiences with programmable robots can increase girls' interest in CS, which can help promote gender equality in STEM fields (Girshin et al., 2023; Sullivan, 2019).

The development of CT begins early in life as children explore their environment and make sense of the world around them. As children grow older, there are benefits to targeted exposure to CS learning experiences.

The California K–12 Computer Science Standards (California State Board of Education, 2018) and K–12 Computer Science Framework (K12CS, 2016)—in alignment with other state and international CS frameworks—highlight the importance of early exposure to CT and CS learning experiences. Although these frameworks differ in many ways, they share some similarities in their framing about how children learn and develop CT skills. For instance, they recognize that children can build foundational CT skills through active, playful learning experiences. Especially for the youngest learners, these experiences are often **unplugged** (meaning that they do not involve the use of technology) and are integrated with other content areas. For example, toddlers sorting socks by size and color not only develop math skills such as sorting and classifying, but they are also demonstrating the CT skill of abstraction, understanding that some are big socks and others are small socks.

As children develop, they continue to explore CT skills in active, playful ways. For example, when preschoolers and early elementary children play the game "Simon Says," they learn the concept of conditional logic ("IF I say 'Simon Says,' THEN perform this action"). At these later ages, children may also have opportunities to engage in **plugged** CT and CS activities. Plugged activities involve some type of technology, such as a robot or coding application, that engages children in coding or programming. **Coding** or **programming** is the "practice of developing a set of instructions (code) that a computer can understand and execute, as well as debugging, organizing, and applying that code

to appropriate problem-solving contexts" (Mills et al., 2021). Programming engages children in an active learning experience that is grounded in a cognitive theory of constructivism (learning-by-doing) with a focus on constructionism (learning-by-creating). In becoming producers of technological artifacts such as computer code, children can develop critical thinking and problem-solving skills while also learning how to express their ideas in new ways.

Implementing CT and CS in Early Learning Settings

Educators can implement CT and CS activities in different ways to support children's development and learning. These activities can be done with or without technology, facilitated through both structured and open-ended experiences, and be taught on their own as standalone CT and CS experiences or integrated with other lessons and content areas.

Unplugged and Plugged Activities

Unplugged activities are highly desirable in early learning settings. In these activities, children explore computational ideas in active, playful ways. For example, they might use story cards to practice sequencing, play games that involve following or creating sequences through movement (such as directing a friend to a specific location using simple instructions), or use collections of objects to model sorting and pattern recognition. Unplugged activities introduce children to CT without any screen time. Additionally, unplugged activities are more accessible, as they pose less of a financial barrier. Educators do not require access to tablets, computers, or robots to implement unplugged CT activities (Bers & Sullivan, 2019; Relkin & Strawhacker, 2021). For more unplugged CT activity ideas and curriculum resources, explore the **Coding Resources for Educators and Families** handout.

Plugged activities involve the use of technology. Some technologies are made of tangible, or **physical**, parts. The KIBO robotics kit, for example, does not involve screens (see figure 2a). Children assemble color-coded wooden blocks containing barcodes, use the robot to scan the barcodes in a particular order, and observe the robot performing the programmed actions. Other programming technologies are graphical, or **screen-based**, interfaces, such as the ScratchJr coding application (see figure 2b). With ScratchJr, children drag together color-coded blocks to create programs that control characters, backgrounds, and their interactions. For example, blue blocks in ScratchJr make characters move, purple blocks change the appearance of characters, and so on. Block-based programming languages like KIBO and ScratchJr use icons instead of text, making them accessible for pre-readers. Finally, other programming technologies have a combination of physical and screen-based parts and are called **hybrid** interfaces. For example, the Blue-Bot robot (see figure 2c) can be programmed using the buttons on top of the robot or using a screen-based application. For more examples of physical, screen-based, and hybrid coding tools for children, explore the **Coding Resources for Children** handout.

Figure 2a. KIBO robotics kit

Figure 2b. ScratchJr application

Figure 2c. Blue-Bot robot

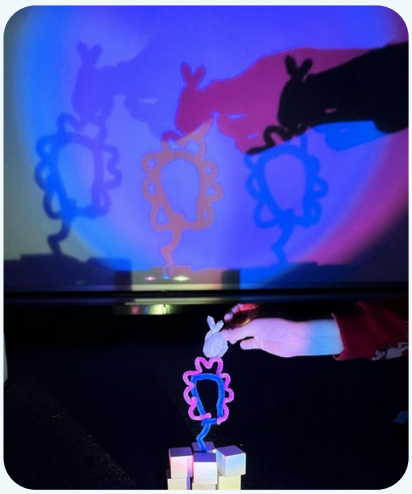
Educators can scaffold children's CT and CS learning through a combination of unplugged and plugged activities. Unplugged CT activities can provide experiences with concrete materials that can help build children's conceptual understanding and foundational skills, which they can draw on when they later engage in more abstract, plugged CS activities. For example, teaching children about algorithms may start with an unplugged activity done in pairs. One child is the programmer, and the other child acts like a robot. The programmer decides on a sequence of actions and communicates the actions step-by-step using directional cards to the other child, or robot. In this activity, the programmer is developing an algorithm for the "robot" to execute. This activity can then lead into a plugged activity, such as children programming a KIBO robot to move from one location to another.

Computational Tinkering

CT and CS activities are most successful in early learning settings when educators use effective teaching and learning strategies, such as:

- creating opportunities for purposeful play,
- providing appropriate materials and space for active exploration,
- prioritizing learners' interests and goals, and
- supporting collaborative learning.

These four strategies are core components of computational tinkering. **Tinkering** involves manipulating, building, making, or taking apart materials in order to test ideas (Tinkering Studio, n.d.). Computational tinkering is a more recent term in the field used to describe how educators can engage children in CT and CS learning experiences in ways that are social, relevant, physical, and cross-disciplinary. For example, educators might offer children a wide variety of materials to explore, observe their interactions with the materials and learning environment, and ask open-ended questions to build on their natural curiosity. As a result, children's explorations are meaningful, learner-driven, and flexible according to their interests and ability levels. The ultimate goals of computational tinkering experiences are joy, agency, and belonging.



Computational Tinkering Using Interactive Exhibits

The AIMS Center for Math and Science Education, in collaboration with Count Play Explore (CPE) partners, has developed facilitator guides on interactive exhibits for children and families on STEAM-related topics such as light and shadow play, wind tubes, and sensory sand. These exhibits serve as prime examples of how to support children's computational tinkering. For instance, in a light and shadow exhibit, children might tinker with a variety of materials (such as flashlights, colorful translucent paper, cooking utensils with holes, and so on) and explore how light passes through the materials differently to cast different kinds of shadows. To extend this playful learning experience, children might work together to introduce robots with light sensors in the space and observe how the robots interact with the exhibit materials. These interactive exhibits illustrate how educators can create learning environments where children engage with both unplugged and plugged activities in ways that promote their playful curiosity about natural scientific phenomena.

Standalone and Integrated Activities

Educators can implement playful CT and CS explorations as standalone activities, like many of the examples in this brief illustrate. Standalone CT activities enable children to work on targeted problem-solving skills, such as solving puzzles on Code.org or programming a robot to move from one location to another. While standalone activities can offer valuable opportunities for children's learning, integrating CT and CS into core subjects is an especially effective and practical approach for educators. By weaving CT and CS investigations into content areas like math, language and literacy, science, and social studies, educators can make CT more relevant and meaningful to children's everyday learning experiences (Rich et al., 2017). For example, a child might retell the story of how they broke their arm by breaking it down into what happened first, next, then, and last (an algorithm). Another example is by implementing a robotics project at the end of a multi-week study on a particular topic. For instance, second-grade children might learn about different cultural dances around the world and then program robots to perform a cultural dance sequence. They may even decorate their robot to look like a dancer from their chosen cultural community.

Taking an integrated approach to CT and CS learning not only promotes meaningful learning experiences, but it also maximizes instructional time without detracting from other content areas. Emerging research with young children demonstrates that when CS lessons are thoughtfully integrated with math and literacy, children make gains in computational skills without compromising their math and literacy achievement (Bers et al., 2022; Yang & Bers, 2024). Moreover, integration positions CT as a universal way of thinking with broad real-world applications, helping educators recognize that they are already fostering CT within their regular curriculum (Rich et al., 2017). As such, while standalone activities can be effective, integrated CT and CS explorations offer a powerful and flexible strategy for developing computational thinkers in early learning settings.

CT and CS Professional Learning for Early Childhood Educators

Many factors influence educators' CT and CS mindsets, knowledge, confidence, and ability to implement CT and CS in their early learning settings. Research shows that many educators are unfamiliar with CT and CS and may feel nervous about teaching it (Ketelhut et al., 2020). Educators may also feel underprepared to teach CS because it involves new concepts they were not introduced to during their pre-service or in-service preparation experiences (California State Board of Education, 2018; Love et al., 2022; Sands et al., 2018).

High-quality professional learning (PL) can strengthen educators' knowledge of CT and CS and improve their confidence in teaching it. High-quality PL consists of the following seven key elements: 1) active, playful learning, 2) clear, coherent content focus, 3) opportunities for collaboration, 4) modeling of effective practices, 5) coaching and individualized support, 6) feedback and reflection, and 7) opportunities for sustained learning (Darling-Hammond et al., 2017).

High-quality professional learning (PL) can strengthen educators' knowledge of CT and CS and improve their confidence in teaching it. High-quality PL consists of the following seven key elements: (1) active, playful learning, (2) clear, coherent content focus, (3) opportunities for collaboration, (4) modeling of effective practices, (5) coaching and individualized support, (6) feedback and reflection, and (7) opportunities for sustained learning (Darling-Hammond et al., 2017).

Table 2 provides a high-level summary of three key studies conducted in early learning settings (Marcella-Burdett et al., 2020; Rich et al., 2017; Yadav et al., 2018). In each study, early childhood educators participated in high-quality PL experiences to build their CT and CS knowledge and increase their confidence to implement CT and CS learning experiences with children. The table highlights ways that the seven key elements of high-quality PL were evident in each of the three studies' PL implementations.

Altogether, these studies serve as examples for how PL facilitators, trainers, and coaches might structure their PL to support educators in their respective settings.

High-quality professional learning can strengthen educators' knowledge of CT and CS and improve their confidence to teach it.

Table 2. Summary of Research Studies Illustrating the Elements of High-Quality CS and CT Professional Learning in Early Learning Settings

Key Element of High-Quality PL	Yadav et al. (2018)	Rich et al. (2017)	Marcella-Burdett et al. (2020)
Active, playful learning	Engaged educators in adult-level unplugged CT activities (for example, writing instructions for the best way to get to a popular destination from their current location)	Engaged educators in engineering design challenges (for example, "Simple Machines") and unplugged computing lessons using Code.org and Scratch	Engaged educators in CT and CS activities (for example, coding penguins, embodied movement with directional cards and floor mats, Sphero and Matatalab robots)
Clear, coherent content focus	PL sessions had a targeted focus on CT	PL sessions had a targeted focus on CS	PL sessions had a targeted focus on CS
Opportunities for collaboration	Educators worked together to design lessons and discuss developmentally appropriate strategies for implementing CT activities with children	Educators worked together to design lessons and discuss developmentally appropriate strategies for implementing CS activities with children	Educators worked together to design lessons and discuss developmentally appropriate strategies for implementing CS activities with children
Modeling of effective practices	Facilitators modeled adult-level CT activities to engage educators as learners and shared strategies for implementing inquiry-based science instruction	Facilitators modeled lessons using the "Engineering is Elementary" curriculum, "Hour of Code" activities on code.org, and Scratch programming activities	Facilitators modeled hands-on activities that educators could later implement with children

Key Element of High-Quality PL	Yadav et al. (2018)	Rich et al. (2017)	Marcella-Burdett et al. (2020)
Coaching and individualized support	Bi-weekly meetings facilitated by a science education researcher	CS and CT researchers were also coaches who supported school faculty and educators	PL facilitators offered multiple forms of coaching to educators who participated in the PL series on coding
Feedback and reflection	Workshop and follow-up meetings included opportunities for educators to reflect on CT concepts and skills	Educators and faculty reflected together on their school vision for engineering and computing core competencies	Sessions included opportunities for educators to discuss how they would implement activities in their learning settings
Opportunities for sustained learning	Initial nine-day workshop followed by bi-weekly sessions	Weekly grade-level sessions over one year	Four sessions with coaching offered throughout the year

Note. The seven key elements of high-quality PL are based on Darling-Hammond et al.'s (2017) framework for effective professional development for educators..

Supporting Families' CT and CS Learning Experiences

Although much of this brief discusses the role of educators, families also play an important role in supporting children's CT and CS learning experiences. Several studies examine CT and CS learning in informal learning spaces, such as libraries, museums, and other community spaces. Interactive exhibits and workshops for families present valuable opportunities for children to engage in CT and CS together with their family members (Govind, 2021). By collaborating on creative coding projects, families can develop new skills, spend quality time together, and learn more about each other's interests. Campana and Mills (2023) explored how library and museum educators incorporated CT into their work with children ages birth to eight and their families. The researchers found that the educators implemented a variety of CT activities, including unplugged, plugged, integrated, and tinkering experiences. For example, educators organized stations featuring Scratch and ScratchJr coding applications, drop-in programs where families built cardboard cities using various sizes and types of recycled cardboard, and craft-based activities where families made binary coding bracelets and designed colorful patterns.

Families can also engage in CT and CS learning experiences at home. For example, families might play logic games, plan the steps for a recipe, or participate in storytelling and building challenges that encourage children to use critical thinking and practice collaborative problem-solving. Families can also introduce age-appropriate coding tools and robotics, allowing children to experience CS learning in a fun and engaging way. Educators can support families in these experiences by encouraging opportunities for inquiry and exploration. For instance, educators might support families to ask children open-ended questions, instead of readily providing the answers, and follow children's natural curiosity and interests. For more resources to engage families in CT and CS learning experiences, explore the **Coding Resources for Educators and Families** handout.

Key Takeaways and Recommendations for Facilitators and Leaders

Establish a vision for implementing CT and CS education in early learning settings.

Given the range of approaches to implementing CS and CT learning, facilitators should work with leaders to establish a vision for what CS and CT learning looks, feels, and sounds like for children ages birth to eight years. This vision can support facilitators to consider ways to tailor resources for educators who work with specific age or grade levels, such as infants and toddlers, pre-kindergarteners, and early elementary children. Consider how facilitators and educators might integrate CT and CS with other content areas to enrich their learning experiences on a range of topics. The way that leaders approach CT and CS education will determine the type of PL experiences facilitators will implement with educators, children, and families in their local communities.

Identify ways that educators and families can integrate CT and CS into their existing practices.

CT is everywhere! All children and educators engage in CT as part of their everyday routines, experiences, and curriculum. Therefore, educators do not need to learn and teach a new content area. Instead, they can find areas that naturally make sense to bring CT and CS into. Supporting educators to integrate CT and CS in active, playful ways can help increase their confidence and comfort in these topics, while also drawing on their existing expertise in content areas they are already teaching. The same principle applies to families. Support families' awareness of how CT and CS can be integrated into everyday learning experiences. Identify resources to support families' development of healthy relationships with technology, including ways to use technology to promote active, playful CT and CS learning.

Explore resources and strategies to support educators' and families' CT and CS mindsets.

Educators and families may experience feelings of anxiety or fear around CT and CS. Identify resources to support adults' mindsets, such as playful CT and CS activities for adults, introductory activities to implement with children, or videos of educators and family members implementing CT and CS learning experiences alongside children. Encourage opportunities to reflect on any prior CT and CS experiences and make connections with other content areas. Highlighting areas where CT and CS can be integrated into daily routines and play can show educators and families that they are already addressing some of these concepts and practices.

The CPE initiative has several web-based suites to support facilitators with implementing CT and CS PL with educators:

- Computational Thinking
- Playful and Creative Coding

Each suite contains resources, including a slide deck, facilitation guide, activities for adults, and handouts.

Support all educators, families, and children to have equitable CT and CS learning opportunities.

As described earlier in this brief, there are many ways to implement CT and CS in early learning settings (for example, unplugged and plugged activities; physical, screen-based, and hybrid coding tools). Consider all these different options (and the costs associated with them) when disseminating resources for educators to implement in their learning settings. For example, consider prioritizing free and low-cost resources to maximize reach, while also providing

options at varying price points. In addition, take time to learn about the unique strengths and backgrounds of educators, children, and families. Consider ways to promote CT and CS learning opportunities for communities with culturally, ethnically, and socioeconomically diverse populations. Drawing upon universal design for learning principles and inclusive pedagogies may help support equitable CS and CT learning for all.

References

Barr, V., & Stephenson, C. (2011). Bringing computational thinking to K-12: What is involved and what is the role of the computer science education community? *ACM Inroads, 2*(1), 48–54.

Bers, M., Strawhacker, A., & Sullivan., A. (2022). *The state of the field of computational thinking in early childhood education.* OECD Education Working Papers, No. 274. OECD Publishing.

Bers, M. U. (2018). Coding and computational thinking in early childhood: The impact of ScratchJr in Europe. *European Journal of STEM Education, 3*(3).

Bers, M. U., & Sullivan, A. (2019). Computer science education in early childhood: The case of ScratchJr. *Journal of Information and Technology Education: Innovations in Practice, 18*, 113–138.

California State Board of Education. (2018). *Computer science standards for California public schools.* https://www.cde.ca.gov/be/st/ss/documents/csstandards.pdf

Campana, K., & Mills, J. E. (2023). Playing, tinkering, and problem solving: Understanding early computational thinking in libraries and museums. *Journal of Early Childhood Research, 21*(3), 369–383.

Darling-Hammond, L., Hyler, M. E., & Gardner, M. (2017). *Effective teacher professional development.* Learning Policy Institute.

De Wit, S., Hermans, F., & Aivaloglou, E. (2021). Children's implicit and explicit stereotypes on the gender, social skills, and interests of a computer scientist. In A. J. Ko & J. Vahrenhold (Eds.), *Proceedings of the 17th ACM Conference on International Computing Education Research* (pp. 239–251).

Garfield County School District. (2023). *What is computer science?* https://www.garfk12.org/what-is-computer-science/

Girshin, R. Z., Rosenberg, N., & Kukliansky, I. (2023). Early childhood robotics: Children's beliefs and objective capabilities to read and write programs. *Journal of Research in Childhood Education, 38*(2), 317–335.

Google Inc., & Gallup Inc. (2016a). *Diversity gaps in computer science: Exploring the underrepresentation of girls, Blacks and Hispanics.* http://goo.gl/PG34aH

Google Inc., & Gallup Inc. (2016b). *Trends in the state of computer science in U.S. K-12 schools.* https://services.google.com/fh/files/misc/trends-in-the-state-of-computer-science-report.pdf

Govind, M. (2021). Fostering computational thinking in homes and other informal learning spaces. In M. U. Bers (Ed.), *Teaching computational thinking and coding to young children.* IGI Global.

Grover, S., & Pea, R. (2013). Computational thinking in K–12: A review of the state of the field. *Educational Researcher, 42*(1), 38–43.

Hansen, A. K., Dwyer, H. A., Iveland, A., Talesfore, M., Wright, L., Harlow, D. B., & Franklin, D. (2017). Assessing children's understanding of the work of computer scientists: The draw-a-computer-scientist test. In M. E. Caspersen (Ed.), *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education,* pp. 279–284.

K12CS. (2016). *K–12 Computer Science Framework.* http://www.k12cs.org

Ketelhut, D. J., Mills, K., Hestness, E., Cabrera, L., Plane, J., & McGinnis, J. R. (2020). Teacher change following a professional development experience in integrating computational thinking into elementary science. *Journal of Science Education and Technology, 29*(1), 174–188.

Love, T. S., Bartholomew, S. R., & Yauney, J. (2022). Examining changes in teachers' beliefs toward integrating computational thinking to teach literacy and math concepts in grades K-2. *Journal for STEM Education Research, 5*, 380–401.

Marcella-Burdett, J., Roth, A., Savelkouls, S., & Zur, O. (2020). *California statewide early math initiative in local communities: Building educator math capacity.* WestEd. https://www.wested.org/resources/california-statewide-early-math-initiative-local-communities/

Master, A., Meltzoff, A. N., & Cheryan, S. (2021). Gender stereotypes about interests start early and cause gender disparities in computer science and engineering. *PNAS, 18*(48).

Mills, K., Coenraad, M., Ruiz, P., Burke, Q., & Weisgrau, J. (2021, December). *Computational thinking for an inclusive world: A resource for educators to learn and lead.* Digital Promise. https://doi.org/10.51388/20.500.12265/138

Monteiro, A. F., Miranda-Pinto, M., & Osório, A. J. (2021). Coding as literacy in preschool: A case study. *Education Sciences, 11*(5). https://doi.org/10.3390/educsci11050198

Ragusa, G., & Leung, L. (2023). The impact of early robotics education on students' understanding of coding, robotics design, and interest in computing careers. *Sensors, 23*(23). https://doi.org/10.3390/s23239335

Relkin, E., & Strawhacker, A. (2021). Unplugged learning: Recognizing computational thinking in everyday life. In M. Bers (Ed.), *Teaching computational thinking and coding to young children* (pp. 41–62). IGI Global.

Rich, P. J., Jones, B. L., Belikov, O., Yoshikawa, E., & Perkins, M. (2017). Computing and engineering in elementary school: The effect of year-long training on elementary teacher self-efficacy and beliefs about teaching computing and engineering. *International Journal of Computer Science Education in Schools, 1*(1), 1–20.

Sands, P., Yadav, A., & Good, J. (2018). Computational thinking in K-12: In-service teacher perceptions of computational thinking. In M. S. Khine (Ed.), *Computational thinking in the STEM disciplines: Foundations and research highlights* (pp. 151–164). Springer International Publishing.

Sullivan, A. (2019). *Breaking the STEM stereotype: Reaching girls in early childhood.* Rowman & Littlefield Publishers.

Tinkering Studio. (n.d.). *Our work at the tinkering studio.* Exploratorium. https://www.exploratorium.edu/tinkering/about

Wing, J. (2006). Computational thinking. *Communications of the ACM, 49*(3), 33–35.

Yadav, A., Krist, C., Good, J., & Caeli, E. N. (2018). Computational thinking in elementary classrooms: Measuring teacher understanding of computational ideas for teaching science. *Computer Science Education, 28*(4), 371–400.

Yang, D., & Bers, M. U. (2024). *Coding as another language: Impact on math and literacy achievement in early CS* [Conference session]. American Educational Research Association (AERA) Annual Meeting, Philadelphia, PA.