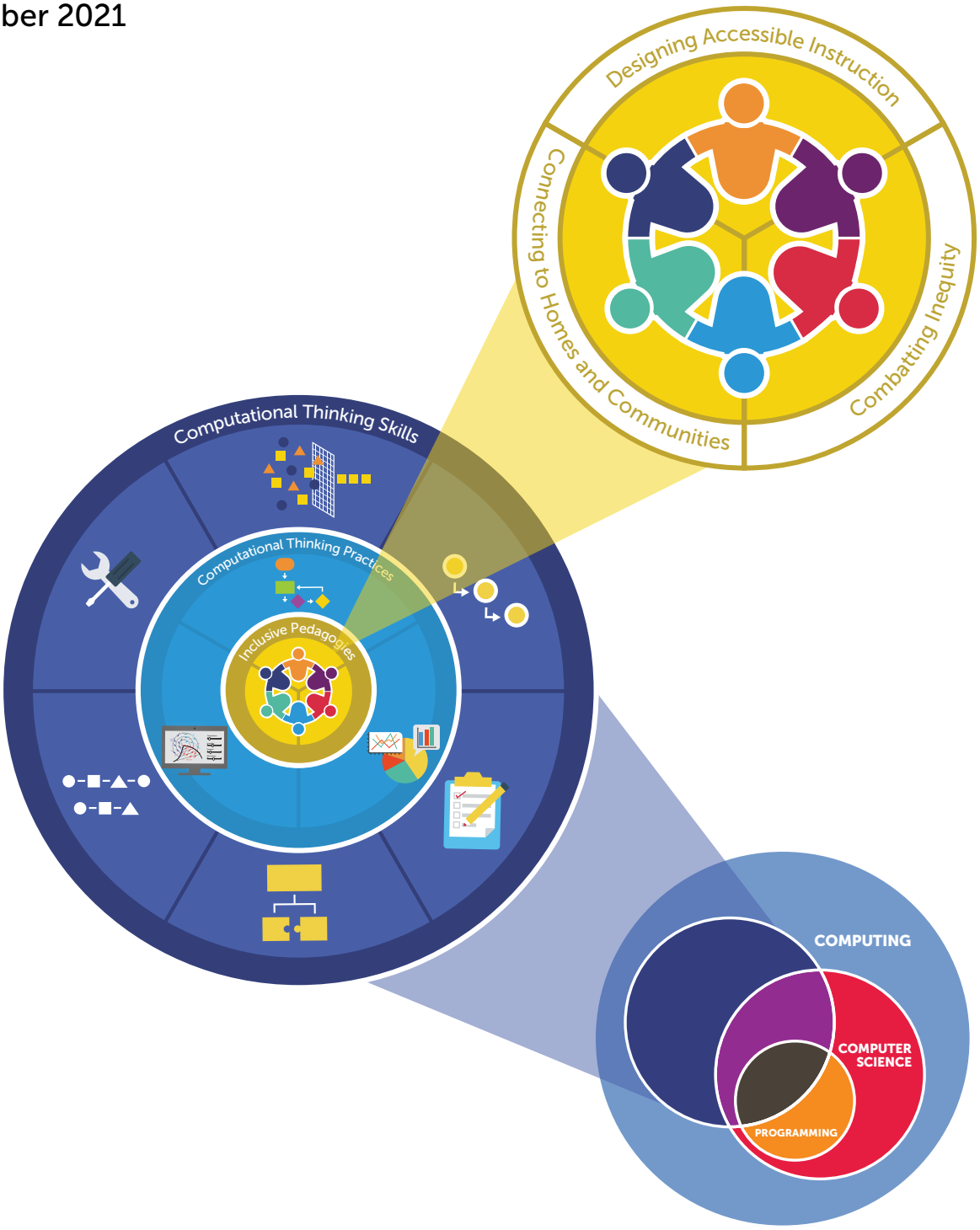
Computational Thinking for an Inclusive World: A Resource for Educators to Learn and Lead

Quick Start and Discussion Guide

Kelly Mills, Merijke Coenraad, Pati Ruiz, Quinn Burke, and Josh Weisgrau

December 2021



Digital Promise
Accelerating Innovation in Education

Suggested Citation

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, Quick start and discussion guide*. Digital Promise. https://doi.org/10.51388/20.500.12265/140

Acknowledgments

This quick start and discussion guide was developed under the guidance of Kelly Mills, Merijke Coenraad, Pati Ruiz, Quinn Burke, and Josh Weisgrau of Digital Promise. It is based on the larger report, *Computational Thinking for an Inclusive World: A Resource for Educators to Learn and Lead.*

This guide was made possible by the generous support of Robin Hood Learning and Technology Fund.

Contact Information

Digital Promise:

Washington, DC:
1001 Connecticut Avenue NW, Suite 935
Washington, DC 20036

San Mateo, CA:
2955 Campus Dr., Suite 110
San Mateo, CA 94403
https://digitalpromise.org/

Table of Contents

Introduction

Technology is becoming more integral across professional fields and within our daily lives, especially since the onset of the pandemic. Now more than ever, it is essential to equip students with the skills to substantially contribute to and genuinely connect within our computational world (Jackman et al., 2021). These opportunities to learn will be important to all students—not only the ones who will eventually study computer science or enter the information technology industry.

While jobs increasingly require the complementary partnership of the processing power of computers and the creativity and expertise of humans (Warschauer & Matuchniak, 2010), our education system has historically prepared only some students with these skills (Margolis, Goode, & Ryoo, 2015) and has systematically excluded certain student populations (Margolis et al, 2008). Large inequalities continue to exist in access to equipment and learning opportunities needed to build computational skills for students that experience marginalization (Code.org et al., 2020). When we say students that experience marginalization in computing, we are referring to Black, Native American, and Latinx students; students with disabilities; girls and non-binary students.

If we are to equip every student in the next generation with the skillset to participate in our technological society, all educators, across disciplines and grade bands, need to provide opportunities for students to engage in computational skills and practices.

> *We call all educators to integrate computational thinking into disciplinary learning across K-12 education, while centering inclusivity, to equip students with the skills they need to participate in our increasingly technological world and promote justice for students and society at large.*

Integrating computing thinking into every classroom is not something that can be left to individual educators. Educational leaders at the district and school levels must prioritize the initiative and build capacity for teachers to do so. This quick start and discussion guide is a resource for educators to learn about and build capacity for students to engage in computational thinking.

Computational Thinking: A Primer

So, what is computational thinking? Many educators find this term mystifying. Although computational thinking has previously been defined as using computational methods to solve interdisciplinary and everyday problems (Barr & Stephenson, 2011; ISTE & CSTA, 2011; Wing, 2006), many educators find this definition offers insufficient guidance for practical classroom implementation. This could be in part because there are many terms used to address skills related to computing. Figure 1 is a representation of the relationship between computer science, computational thinking, programming and computing.

K–12 Computing: Then and Now

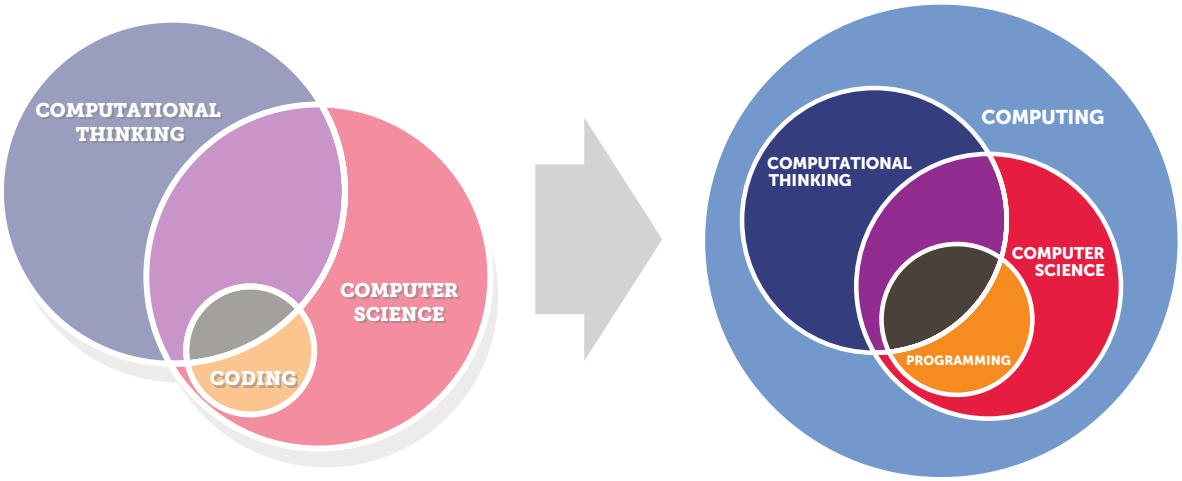


Figure 1. The relationship between computer science (CS), computational thinking (CT) and coding released in Computational Thinking for a Computational World (Angevine et al., 2017) and updated.

- **Programming** is the practice of developing a set of instructions that a computer can understand and execute, as well as debugging, organizing, and applying that code to appropriate problem-solving contexts. It lies among computer science and computational thinking. It entails technical skill (coding), in addition to problem-solving (e.g., debugging), design aesthetics (e.g., concise lines), and documentation.
- **Computer science** is "the study of computers and algorithmic processes, including their principles, their hardware and software designs, their applications, and their impact on society" (Tucker et al., 2003).
- **Computational thinking** is "a way of solving problems, designing systems, and understanding human behavior that draws on concepts fundamental to computer science… a fundamental skill for everyone, not just computer scientists." (Wing, 2006).
- **Computing** meanwhile is any activity or area of study that leverages computational methods, models, or systems, such as information management, computer engineering, artificial intelligence, data science, entertainment media and more (CC20 Taskforce, 2020). Computing involves any skill or practice from computer science and/or computational thinking.

Reflect:

What experiences have you had to engage in programming, computer science and/or computational thinking? What did you do and how did it make you feel?

Computational Thinking Skills and Practices: A Framework for Integration

In order to integrate computational thinking into K-12 teaching and learning, educators must define what students need to know and be able to do to be successful computational thinkers. Our recommended framework has three concentric circles.

- **Computational thinking skills**, in the outermost circle, are the cognitive processes necessary to engage with computational tools to solve problems. These skills are the foundation to engage in any computational problem solving and should be integrated into early learning opportunities in K-3 either plugged (with a computational device) and unplugged (without a computational device). Although both plugged and unplugged activities are valuable learning experiences for students to build computational thinking skills, leveraging technological tools where appropriate can support learners to connect and apply these skills to engage in computational thinking practices in the older grades.
- **Computational thinking practices**, in the middle circle, combine multiple computational skills to solve an applied problem. Students in the older grades (4-12) may engage in these practices to develop computational artifacts such as a computer program, data visualization, or computational model.
- **Inclusive pedagogies**, in the innermost circle, are strategies for engaging all learners in computing, connecting applications to students' interests and experiences, and providing opportunities to acknowledge, and combat biases and stereotypes within the computing field.

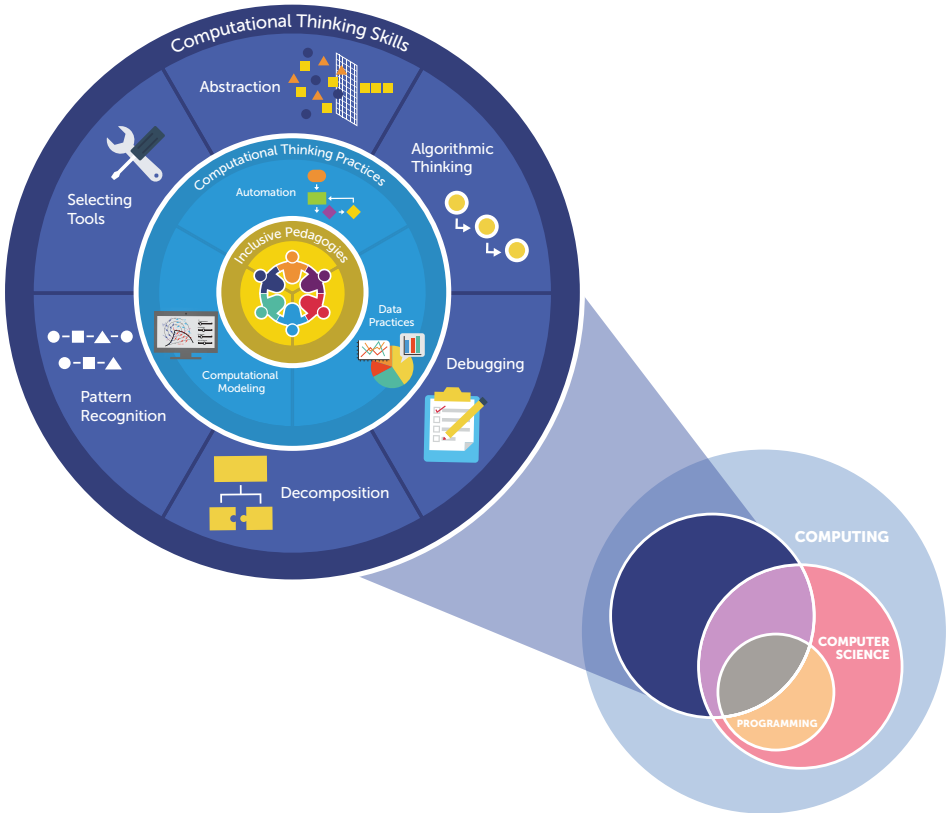
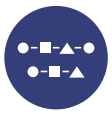


Figure 2. A framework for computational thinking integration.

CT Skills	Description	Example
Abstraction 	Filtering aspects of a problem or phenomenon for what is most important	Preschool students sort buttons based on categories they define (e.g., color, number of buttonholes, shape).
Algorithmic Thinking 	Organizing steps in an ordered sequence	Second graders put different parts of a story (or even a comic strip) in the correct order (Ruiz et al., 2021) and identify the key images and words that serve as the operational "hinges" pointing to a specific sequence.
Debugging 	Iteratively testing, finding errors, and fixing them	Fourth grade students might be given a block-based coding project to calculate the area of a rectangle that uses an addition operator rather than a multiplication one. Students must debug the algorithm to correctly calculate the area based on user input (Everyday Computing, 2021).
Decomposition 	Dividing problems into smaller parts	After reading a book, first grade students might decompose the storyline into beginning, middle, and end or introduction, rising action, climax, falling action, and resolution.
Pattern recognition 	Recognizing recurrent features, data, or relationships	Students in Kindergarten might draw a tree at different times of each year and then pool their different pictures to arrive at the tell-tale signs of a particular season by the pooled imagery.
Selecting tools 	Choose a computational device with the appropriate affordances to complete a task	Fifth grade students might choose between a digital thermometer and a programmable probe while designing an experiment monitoring temperature changes over time.

CT Practices	Description	Example
Automation 	Developing a systematic set of instructions for a computer to carry out a task more often, more efficiently, and/or more accurately than a human.	High school students may create a mobile application that helps community members to find the nearest stores with fresh fruits and vegetables and promotes healthy eating.
Computational modeling 	Representing relationships within complex systems using a computational tool, particularly phenomena that cannot be otherwise thoroughly examined due to constraints of time, size, and/or visibility	Middle school students might explore phases of matter by using a simulation to manipulate temperature in a solid, liquid or gas and observe the speed and distance of molecules in each phase.
Data practices 	Leveraging computational models and methods to collect, analyze, and visualize data.	High school students could choose a locally-relevant data set from their city's open data portal, analyze the data using spreadsheet software, and create a data visualization aimed at an audience of community members.

Table 1. Description of computational skills and practices

Reflect:

Which of these terms are you already familiar with?
Which terms would you like to learn more about?

Reimagining Computational Thinking to be More Inclusive

The push to integrate computational thinking to support disciplinary learning provides an opportunity for schools to reimagine and redefine computing education for students, especially those that have experienced exclusion, knowing these skills will equip them for success in a computational world, regardless of their career choice. Notably, strategies to increase access to computational thinking learning opportunities for all students not only aim to increase opportunities to engage students who are excluded from computing, but also seek to change how and why students are engaging in computing. While computer science instruction has traditionally focused on skill acquisition (e.g., coding), the redesign of inclusive learning experiences must intentionally integrate connections to students' interests, homes and communities and acknowledge and challenge inequity at the outset of the learning rather than as an afterthought (Kafai et al., 2019).

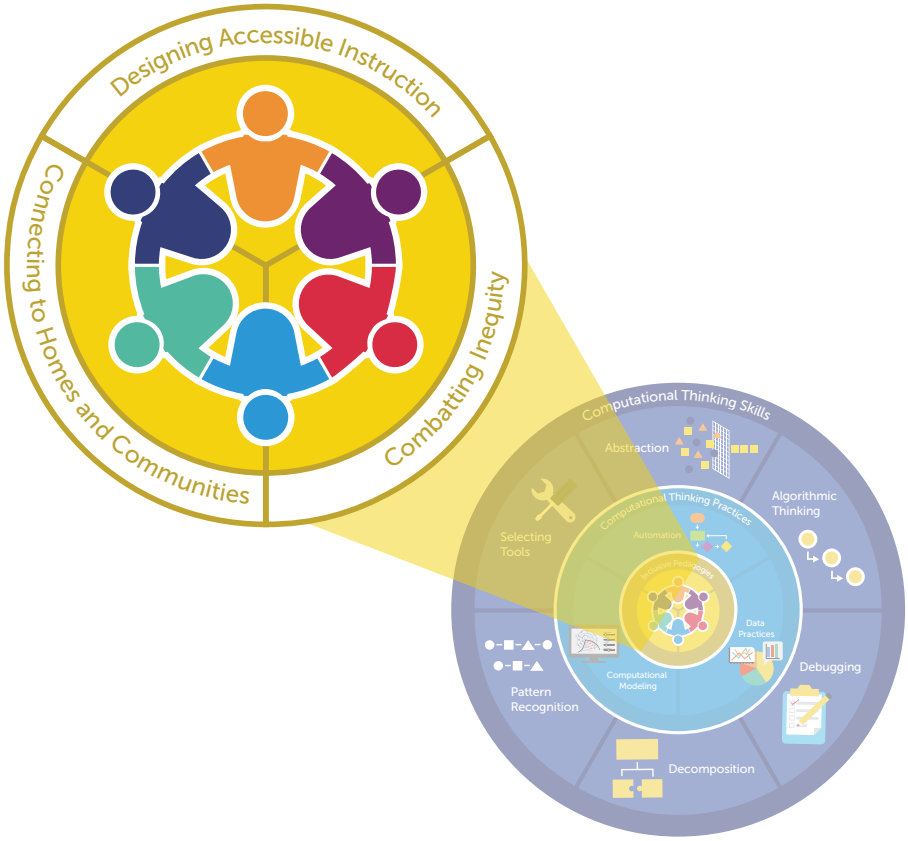


Figure 3. Inclusive pedagogies centered within the computational thinking framework

So, what teaching practices promote inclusive computational thinking and what does inclusive computational thinking look like in a classroom? In the chart below, we provide examples of inclusive computing pedagogies in the classroom. The pedagogies are divided into three categories to emphasize different pedagogical approaches to inclusivity. **Designing Accessible Instruction** refers to strategies teachers should use to engage all learners in computing. **Connecting to Students' Interests, Homes, and Communities** refers to drawing on the experiences of students to design learning experiences that are connected with their homes, communities, interests and experiences to highlight the relevance of computing in their lives. **Acknowledging and Combating Inequity** refers to a teacher supporting students to recognize and take a stand against the oppression of marginalized groups in society broadly and specifically in computing.

Designing Accessible Instruction
• Utilize the principles of Universal Design for Learning[1] and provide multiple means of engagement, representation, and action and expression, such as using a physical representation of code blocks to model computing or providing students with starter code. • Facilitate well-structured **collaborative learning** opportunities, such as pair programming[2] and Process Oriented Guided Inquiry Learning (POGIL)[3]. • **Make expectations explicit**, such as providing a checklist for students to check their progress as they develop computational artifacts. • **Normalize and encourage making mistakes**, getting stuck and redoing activities when developing and refining computational products. One way to do this is through think-alouds where teachers (or students) say what they are thinking and doing when programming. • **Provide students choices** in their creation of computational artifacts by curating and modeling a variety of product outputs (e.g., interactive art, digital stories, e-textiles, video games, and simulations) that align with learning objectives.
Connecting to Students' Interests, Homes, and Communities
• Give students the opportunity to **develop and express their interests and experiences** through computational thinking activities and computing tools. • **Connect with family and community** members to incorporate their perspectives into computing activities. • **Promote linguistic pluralism** in class and computing projects by encouraging translanguaging[4], allowing students to create projects in their language of choice, or utilizing the translated block options on Scratch. • Showcase students' work with technology related **organizations in the community** to connect, celebrate, and legitimize their expertise.
Acknowledging and Combating Inequity
• **Acknowledge power dynamics** associated with race and representation in computing, such as the overrepresentation of white male computer scientists, and actively combat those power dynamics in the classroom (e.g. amplifying the voices of students experiencing marginalization). • **Recruit students who experience exclusion from computing to computer science classes** (e.g., Black, Native American, and Latinx students; students with disabilities; girls; non-binary students). • **Celebrate the multiple, overlapping identities** of students experiencing marginalization and validate the unique perspectives they hold about technology. • **Challenge stereotypes** about who a computer scientist is and who belongs in the technology field, such as celebrating and giving examples of the work of people that have experienced marginalization based on race, gender, or ability. • Teach students about **biases in technology**, such as in algorithms and artificial intelligence. • Provide opportunities for students to **speak about and improve injustices** through computational means, such as creating an app, digital story, or game that showcases and counters inequities.

Table 2. Examples of inclusive computing in classroom practice (Adapted from Israel et al., 2017; Kapor Center, 2021; Madkins et al., 2020; National Center for Women & Information Technology, 2021b; Paris & Alim, 2017; Ryoo, 2019; CSTeachingTips, 2021)

Reflect:

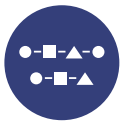
- ***Which inclusive computational thinking pedagogies stood out to you as especially relevant to your learners, community and/or content area? Explain.***
- ***How will you leverage inclusive pedagogies in your classroom, school or district?***

1 Universal Design for Learning is a framework designed to improve teaching and learning for all learners.(CAST, 2018).
2 Pair programming is a technique used in both education and industry. In education it has shown to improve student retention and performance in CS courses (National Center for Women & Information Technology, 2021c).
3 Process Oriented Guided Inquiry Learning is a structured collaborative learning strategy that guides students in constructing their own understanding of key concepts (Kussmaul, Brinkman, & Quinn, 2017).
4 The theory that people fluidly call into action their full meaning-making resources in ways that defy categorization. For example, in a single interaction people may use words from multiple languages as well as gestures, emoji, and other environmental resources (Vogel, et. al., 2019).

Integrating Computational Thinking into Disciplinary Learning

To provide all learners, especially those experiencing marginalization, opportunities to engage in computational thinking, it is essential that educators integrate computational thinking with the topics they already teach, like math and ELA. Then, computational thinking can become a value-add to disciplinary learning and not an add-on to an already overscheduled school day. Educators can design powerful learning experiences by leveraging synergies between disciplinary learning and computational thinking and promoting student agency and purpose. In the younger graders, educators should provide opportunities for students to build computational thinking skills to develop an interest in and a foundational understanding of computing. Table 3 below describes a few examples of how computational thinking can be integrated into core disciplinary learning (arts, English language arts, math, science and social studies) in early learning (PreK–2), upper elementary school (grades 3–5) and secondary school (grades 6–12).

CT Skills Key:


Pattern Recognition


Algorithmic Thinking


Decomposition


Abstraction


Debugging


Selecting Tools

CT Practices Key


Data Practices


Automation


Computational Modeling

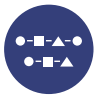
Grade band	Arts	English Language Arts	Math	Science	Social Studies
PreK–2	 Break down the task of creating a large object out of playdough into smaller steps.  Use a repeating pattern to design a bead necklace or paper chain.  Perform a sequence of music notes or dance moves	 Put the plot of a story in the correct order (e.g., We're Going on a Bear Hunt)  Read "How to Code a Sandcastle" and discuss how Pearl broke down the steps for Pascal. Then, write class steps to build a sandcastle or an object that will be familiar to your students.	 Use dice to select an action or sound to make (e.g. animal sound if you roll a 3). Another dice can be used to determine the number of times to repeat that action or sound.  Create graphs of data relevant to your students or classroom (e.g., birthdays, word walls)	 Sort objects according to shared attributes (e.g., food groups)  Draw a tree at different times of the year	 Create a map and provide precise instructions for a robot to travel to different destinations on their map. The robot can be a child following directions or an electronic robot like a Code-a-Pillar or Bee Bot.

Table 3a. Examples of computing integrated into K–12 arts, ELA, math, science, and social studies in grades K-2.

Grade band	Arts	English Language Arts	Math	Science	Social Studies
3–5	 Compose a song using a computational tool such as Scratch	 Collect and analyze data about text features in non-fiction books  Compose an interactive "choose your own adventure" story using a computational tool such as Alice or Scratch.	 Use Scratch to create programs that calculate area and perimeter given user inputs	 Use a simulation to explore scientific phenomena (e.g., how mass affects force when two objects collide).  Code a micro:bit to measure light levels and use their micro:bit light meter to conduct an experiment (e.g., indicate pollution levels in water)	 Develop stories of different historical perspectives using data from primary sources

Table 3b. Examples of computing integrated into K–12 arts, ELA, math, science, and social studies in grades 3-5.

Grade band	Arts	English Language Arts	Math	Science	Social Studies
6–12	 Create a visualization that emphasizes bias or injustice with a dataset  Use e-textiles to create a quilt square.	 Develop an algorithm for decomposing an essay prompt	 Conduct statistical analysis about their own health and wellness habits (e.g. snack habits, stress levels)	 Use, decode, or remix a computational model about a scientific phenomenon (e.g. weather)  Analyze and evaluate bias in secondhand data about socioscientific issues (e.g. cancer, pollution)	 Analyze data critically examining sociopolitical structures within society and technology (e.g., redlining and digital redlining)  Critique how automation in society has contributed to inequitable outcomes for peoples experiencing marginalization.

Table 3c. Examples of computing integrated into K–12 arts, ELA, math, science, and social studies in grades 6-12.

Table 3. Adapted from Canon Lab (n.d.); Coenraad et al., 2021; Computational Thinking for Next Generation Science, EDC Integrated Modules, Integrated Computational Thinking, Preschool Computational Thinking, Tough as Nails Boosters

Reflect:

- ***Are there some computational thinking skills or practices that are common in your classroom, school or district?***
- ***What ideas do you have to integrate computational thinking into disciplinary learning?***

Developing Capacity for Computational Thinking

There are commonly strong examples of computational thinking integrated in pockets within districts. That is, several classrooms or teachers facilitate rigorous computational thinking practices with students. However, this engagement is rarely consistent across classrooms within schools or across schools within districts. This often leads to fewer and less robust computing learning opportunities for students experiencing marginalization. In order to scale and sustain computational thinking integration within districts and schools, educators must promote shared leadership between districts and schools and designing professional learning experiences for pre- and in-service teachers.

Reflect:

How can you engage teacher, school and district leaders to build capacity for computational thinking?

Conclusion

In order to fully participate in our increasingly computational world, all students, especially those experiencing marginalization, must have access to computational thinking and engage with technology in ways that promote justice for the students and for society at large. Now is the time to integrate computational thinking with inclusive pedagogies into every classroom.

To learn more about computational thinking, read our full report.

References

Angevine, C., Cator, K., Roschelle, J., Thomas, S. A., Waite, C., & Weisgrau, J. (2017). *Computational thinking for a computational world* [White paper]. Digital Promise Global. https://digitalpromise.org/wp-content/uploads/2017/12/dp-comp-thinking-v1r5.pdf

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? *Inroads, 2*(1), 48–54. https:doi.org/10.1145/1929887.1929905

Canon Lab. (n.d.). *How to code a sandcastle.* https://www.canonlab.org/copy-of-margaretmoon-1.

CAST. (2018). Universal Design for Learning guidelines version 2.2. http://udlguidelines.cast.org

CC20 Task Force. (2020). *Computing Curricula 2020: Paradigms for global computing education.* Association for Computing Machinery, New York, NY, USA. Retrieved from: https://www.acm.org/binaries/content/assets/education/curricula-recommendations/cc2020.pdf

Code.org, CSTA, & ECEP Alliance. (2020). *2020 State of computer science education: Illuminating disparities.* https://advocacy.code.org/stateofcs

Coenraad, M., Cabrera, L., Killen, H., Plane, J., & Ketelhut, D. J. (2021). Computational thinking integration in elementary teachers' science lesson plans. *ACM Special Issue on K–5 CT*, 11–18.

CSTeachingTips. (2021, September 14). *Tips for reducing bias*. https://www.csteachingtips.org/tips-reducing-bias

International Society for Technology in Education (ISTE) and the Computer Science Teachers Association (CSTA). (2011). *Operational definition of computational thinking for K–12 education.* https://cdn.iste.org/www-root/ Computational_Thinking_Operational_Definition_ISTE.pdf

Israel, M., Lash, T. A., & Jeong, G. (2017). Utilizing the Universal Design for Learning framework in K–12 computer science education. Project TACTIC: Teaching All Computational Thinking through Inclusion and Collaboration. https://www.learningdesigned.org/sites/default/files/udl.pdf

Jackman, J. A., Gentile, D. A., Cho, N. J., & Park, Y. (2021). Addressing the digital skills gap for future education. *Nature Human Behaviour, 5*(5), 542–545. https://www.nature.com/articles/s41562-021-01074-z

Kafai, Y. B., Proctor, C., & Lui, D. (2019). From theory bias to theory dialogue: Embracing cognitive, situated, and critical framings of computational thinking in K–12 CS education. *International Computing Education Research Conference (ICER '19)*, 101–109. https://doi.org/10.1145/3291279.3339400

Kapor Center. (2021). Culturally Responsive-Sustaining Computer Science Education: A Framework [White Paper]. https://mk0kaporcenter5ld71a.kinstacdn.com/wp-content/uploads/2021/07/KC21004_ECS-Framework-Report_v9.pdf

Kussmaul, C., Brinkman, B., & Quinn, B. A. (2017). Exploring inquiry learning: an EngageCSEdu author and a user discuss POGIL. *ACM Inroads 8*, 3 (September 2017), 27–30. https://doi.org/10.1145/3123650

Madkins, T. C., Howard, N. R., & Freed, N. (2020). Engaging equity pedagogies in computer science learning environments. *Journal of Computer Science Integration, 3*(2), 1–27. http://doi.org/10.26716/jcsi.2020.03.2.1

Margolis, J., Estrella, R., Goode, J., Holme, J. J., & Nao, K. (2008). *Stuck in the shallow end: Education, race, and computing.* MIT Press.

Margolis, J., Goode, J., & Ryoo, J. J. (2015). Democratizing computer science. *Educational Leadership, 72*(4), 48–53. https://www.ascd.org/el/articles/democratizing-computer-science

National Center for Women & Information Technology. (2021b, September 14). *Engagement practices framework.* https://www.ncwit.org/engagement-practices-framework.

National Center for Women & Information Technology. (2021c, October 26). Pair programming-in-a-box: The power of collaborative learning. https://ncwit.org/resource/pairprogramming/

Paris, D., & Alim, H. S. (Eds.). (2017). *Culturally sustaining pedagogies: Teaching and learning for justice in a changing world.* Teachers College Press. https://eric.ed.gov/?id=ED580787

Ryoo, J. J. (2019). Pedagogy that supports computer science for all. *ACM Transactions on Computing Education (TOCE)*, 19(4), 36. https://doi.org/10.1145/3322210

Tucker, A., Deek, F., Jones, J., McCowan, D., Stephenson, C., & Verno, A. (2003). *A model curriculum for K–12 computer science. Report of the ACM K–12 Task Force Curriculum Committee*, CSTA. https://doi-org.proxy-um.researchport.umd.edu/10.1145/2593247

Vogel, S., Ascenzi-Moreno, L., Hoadley, C., & Menken, K. (2019). The role of translanguaging in computational literacies: Documenting middle school bilinguals' practices in computer science integrated units. In *SIGCSE 2019 - Proceedings of the 50th ACM Technical Symposium on Computer Science Education*, 1164–1170. https://doi.org/10.1145/3287324.3287368

Warschauer, M., & Matuchniak, T. (2010). New technology and digital worlds: Analyzing evidence of equity in access, use, and outcomes. *Review of Research in Education, 34*(1), 179–225. https://doi.org/10.3102/0091732X09349791

Wing, J. M. (2006). Computational thinking. *Communications of the ACM, 49*(3), 33–35. https://doi.org/10.1145/1118178.1118215