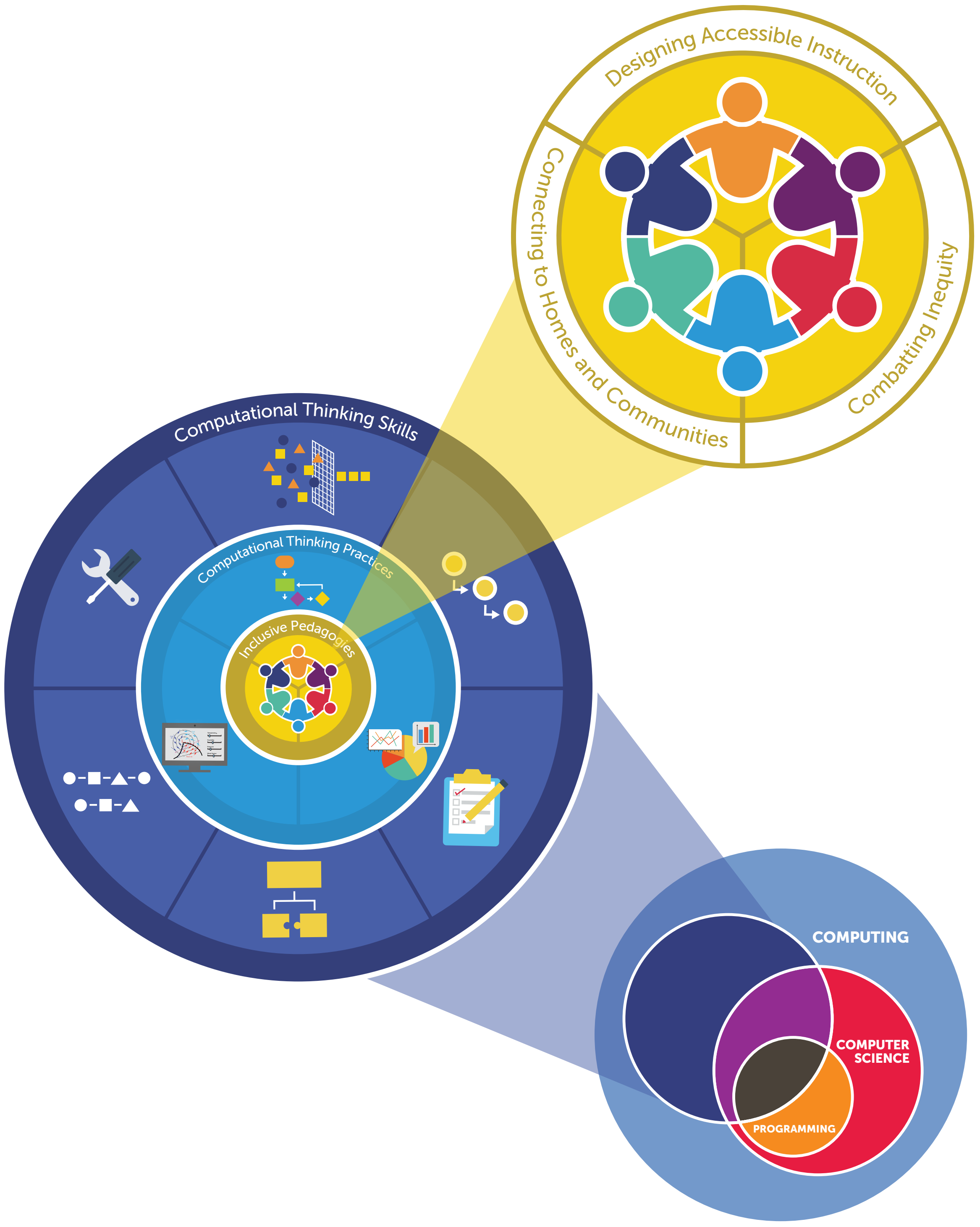
Computational Thinking for an Inclusive World: A Resource for Educators to Learn and Lead

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

December 2021


Designing Accessible Instruction
Connecting to Homes and Communities
Combatting Inequity
Computational Thinking Skills
Computational Thinking Practices
Inclusive Pedagogies
COMPUTING
COMPUTER SCIENCE
PROGRAMMING

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.* Digital Promise.
https://doi.org/10.51388/20.500.12265/138

Acknowledgments

This report was developed under the guidance of Kelly Mills, Merijke Coenraad, Pati Ruiz, Quinn Burke, and Josh Weisgrau of Digital Promise.

The authors thank the experts interviewed for the content of this paper and those who reviewed and offered comments.

- Natalie Araujo Melo, Northwestern University
- Steve Azeka, Robin Hood Learning & Technology Fund
- Joan Beatrice Ladaban, New York City College of Technology
- Jessica Bibbs-Fox, Compton Unified School District
- Lautaro Cabrera, University of Maryland, College Park
- Jacqueline DiSanto, Hostos Community College, City University of New York
- Ximena Dominguez, Digital Promise
- Julie Evans, Project Tomorrow
- Jamie Ewing, Public School 277 in Bronx, New York
- Cheri Fancsali, Research Alliance for New York City Schools
- David Gomez, Project Tomorrow
- Carlos Guevara, Hostos Community College, City University of New York
- Christopher Hoadley, New York University
- Maya Israel, University of Florida
- Danae Kamdar, Digital Promise
- Irene Lee, Education Development Center
- Tiffany Leones, Digital Promise
- Amy Lindahl, Fred Hutchinson Cancer Research Center
- Emily Nestor, Talladega County Public Schools
- Babette Moeller, Education Development Center
- Brooke Morgan, Talladega County Public Schools
- Amber Oliver, Robin Hood Learning & Technology Fund
- Aankit Patel, The City University of New York
- Melissa Rasberry, Friday Institute for Educational Innovation
- Meg Ray, The City University of New York
- Jeremy Roschelle, Digital Promise
- Kennan Scott, Bitwise Industries
- Heather Sherwood, Education Development Center
- Joy Straub, Mission Vista High School
- Ronald Summers, CS4All, New York CIty Depar
- Ashleigh Thompson, The City University of New York
- Sara Vogel, New York University

This report 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

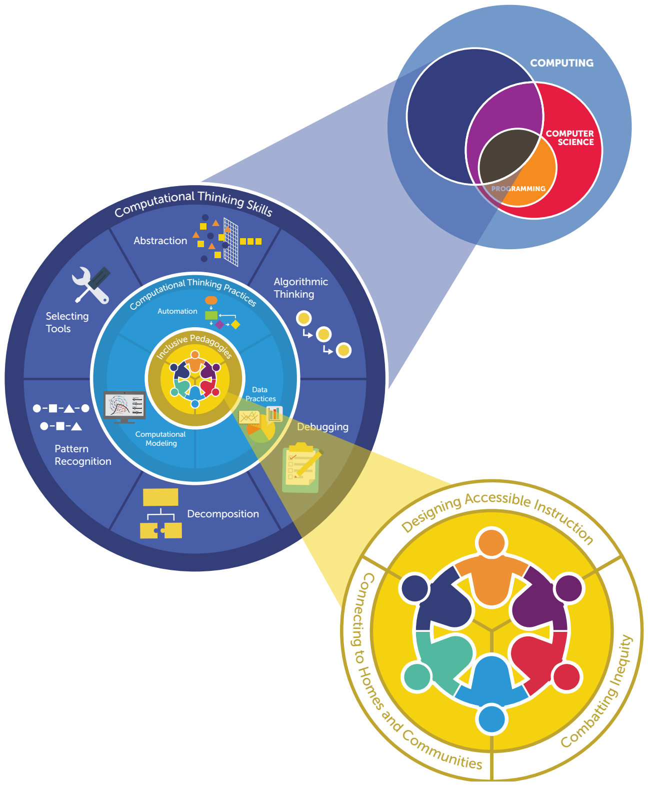
Executive Summary

Technology is becoming more integral across professional fields and within our daily lives, especially since the onset of the pandemic. As such, opportunities to learn computational thinking are important to all students—not only the ones who will eventually study computer science or enter the information technology industry. However, large inequalities continue to exist in access to equipment and learning opportunities needed to build computational thinking skills for students that experience marginalization.

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.

We introduce three elements to clarify computational thinking concepts for educators:

- A Venn diagram illustrating the relationship between computer science, computational thinking, programming and computing.
- A framework for computational thinking integration including foundational computational skills, applied computational practices and inclusive pedagogies.
- Inclusive pedagogies, divided into three categories to emphasize different pedagogical approaches to inclusivity: designing accessible instruction, connecting to students' interests, homes, and communities and acknowledging and combating inequity



Recommendations

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 art, English language arts, math, science and social studies. We recommend that educators use the following strategies to design powerful learning experiences for students:

Leverage synergies between disciplinary learning and computational thinking	Provide opportunities for students to build computational thinking skills in the younger grades	Promote student agency and purpose

Integrating computing thinking into every classroom is not something that can be left to individual educators. Educational leaders must prioritize the initiative and build capacity for teachers to do so. We recommend the following strategies for teacher, building and district leaders to scale and sustain computational thinking integration:

Promote shared leadership among districts, schools and teachers	Develop sustained, individualized professional learning opportunities	Integrate computational thinking into pre-service teacher education

Introduction

Photo by Allison Shelley for EDUimages

In our 2017 report, Computational Thinking for a Computational World, we posed the question, "In a computational world, what is important to know and know how to do?" We made the argument that technology was deeply embedded into our daily lives and learning computing is a pressing need for education. We must equip students with the computational skills to fully participate in the changing nature of the workforce, as well as in their communities and social change more broadly (Angevine et al., 2017).

Since that time, technology has advanced and our world continues to become integrated with and dependent upon computing. In particular, the COVID-19 pandemic upended our lives, changing the nature of our work and socialization. Educators have shifted in-person instruction to remote and hybrid learning. People in many employment sectors have left the office altogether, turning to virtual methods and systems to perform their jobs. For many people, their primary means of communication and entertainment have become almost entirely dependent on digital platforms. Given the newfound need for people to connect and engage in their work and learning, 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).

Specifically, there is a growing need to reframe the power of computing in classrooms as an inherently social and learned set of skills. The end goal is not just providing access to digital devices but developing skills with digital devices for heightened disciplinary learning, critical thinking, and self-expression. Students' use of computers must move beyond consuming information or doing routine tasks like reading, writing, or presenting. Students need opportunities to create, decode, analyze, customize, or otherwise manipulate a computer program or predictive model to solve a problem or support their learning goals. They also need the opportunity to consider the ways that technological systems, including the bias designed into technological systems, are affecting their lives. 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—because technology is now integral across fields and within our daily lives.

As we expand computing opportunities in schools and encourage students to be creators and critical users rather than simply consumers of technology, it is important to recognize that our evolving world of technological innovation is still rooted in and reinforcing systemic injustice. These injustices are caused not only by bias within technical systems as evidenced by examples of algorithmic bias (Noble, 2018; Wachter-Boettcher, 2017), but also the digital learning and inclusion gap (LearnPlatform, 2021). This gap contributes to the lack of diversity in the technology workforce (Bureau of Labor Statistics, 2018; 2019). 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).

Photo by Allison Shelley/The Verbatim Agency for EDUimages

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. In school, discrepancies exist not only in access to computing devices, but how those devices and the internet are used. In low-socioeconomic (SES) schools, students are more likely to use computers for drill and practice or to develop familiarity with software applications rather than for more creative, complex or critical thinking purposes (Margolis et al., 2008; Warschauer & Matuchinak, 2010). The shift to distance learning in the spring of 2020 only exacerbated these disparities. Students experiencing marginalization had disproportional access to learning opportunities based on their access to computing devices and the internet (Herold, 2020; Hodges et al., 2020).

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, needs 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 computational 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 report is a resource for educators, from classroom teachers to building administrators to district leaders, to learn about and build capacity for students to engage in computational thinking. In our previous report, we made a case for learning computational thinking as a core skill for the world we live in, defined computational thinking and its relationship to coding and computer science, and suggested that CT could be integrated across subject areas. In this report we update those definitions and provide strategies to integrate computational thinking into disciplinary learning.

It is divided into three sections. First, we present a brief overview on computational thinking. What do we mean by the term and how do we apply it in practice? Next, we examine the current state of K–12 computing education and the persistent challenges of providing students with access to computational tools and addressing systemic inequity in computing education and the tech world more broadly. To combat these challenges, we propose a framework for computational thinking integration with inclusive pedagogies at the center. Finally, we describe two distinct needs for educators to create inclusive learning opportunities for computing: (1) integrating computational thinking into disciplinary learning and (2) building teachers' capacity for computational thinking. We describe strategies to address each need and highlight examples of educators using each strategy in action. By the end of the report, we hope that readers will have identified concrete next steps to further inclusive computing education in their contexts.

Computational Thinking: A Brief Refresher

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 every-day 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.

Digital Promise illustrated the relationship between computer science (CS), computational thinking, and coding with a Venn diagram in 2017 (Angevine et al., 2017; Figure 1). Initially, the image sought to succinctly capture the intersection and separation of computer science as a discipline, computational thinking as a practice, and coding as a distinct skill. However, it also raised awareness of ambiguities in how educators were using these terms.

We have updated the previous representation of computational terminology with two modifications. First, we added the term "computing," which encompasses skills and practices in both computer science and computational thinking. Additionally, we replaced the term "coding" with the term "programming." In the 2017 report, we defined coding as "developing a set of instructions that a computer can understand and execute." Since that time, the colloquial use of the term "coding" has become manifestly broader, some-times used to describe a highly discrete technical skill and at other times to describe the underpinning of computer science education. Given advancements in computing such as the development of artificial intelligence that can autonomously generate code (Metz, 2021), as well as ever-expansive online libraries of pre-written code, software development is no longer a line-by-line process. Rather the skill set is shifting to modeling, debugging, organizing and applying code, which the term "programming" better captures. While coding is a highly visible term as it relates to K–12 educational advocacy and marketing for computer sci-ence initiatives, we believe programming is now a more precise terminology to reference the wider thought and problem-solving processes involved in the development of instructions for digital devices, as well as the associated considerations of how end-users interact with such devices (Metz, 2021).

K–12 Computing: Then and Now

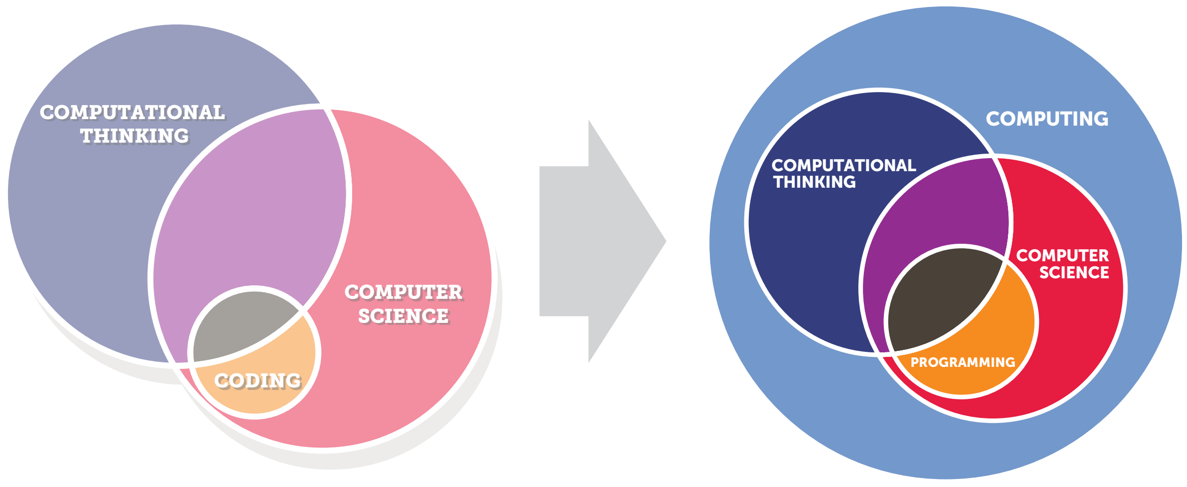


Figure 1. The relationship between computer science (CS), computational thinking (CT) and coding released in Computational Thinking for a Computational World (Angevine et al., 2017) and updated.

- **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.

While computer science is an individual academic discipline, computational thinking is a problem-solving approach that integrates across activities (Angevine et al., 2017). The skills and practices requiring computational thinking are broader, leveraging concepts and skills from computer science and applying them to other contexts, such as core academic disciplines. For educators integrating computational thinking into their classrooms, we believe computational thinking is best understood as a series of interrelated skills and competencies. We describe these interrelated skills and competencies below.

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 (Figure 2) has three concentric circles. In the outermost circle, **computational thinking skills** are the cognitive processes necessary to engage with computational tools to solve problems. In the middle circle, **computational thinking practices** combine many computational skills to solve an applied problem; many of these practices result in the development of a computer program, a data visualization, or a computational model that could be used to solve problems related to one another. In the innermost circle, **inclusive pedagogies** are those 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. Table 1 provides descriptions for each concept in the outer and middle circles.

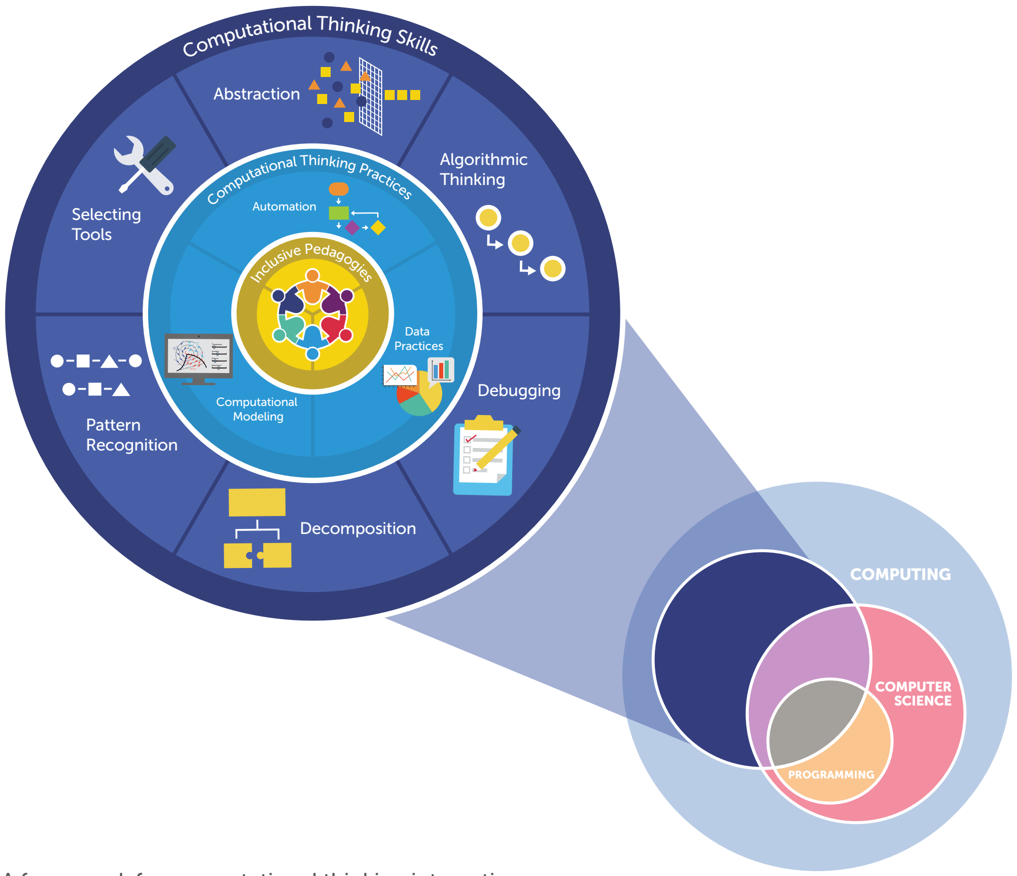
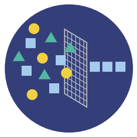
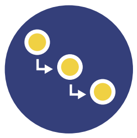


Figure 2. A framework for computational thinking integration.

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

We recommend applying the computational thinking integration framework above in light of wider child development considerations to determine what and how to teach to students of different ages and cognitive capacities. In younger grades (K–3), students will likely be engaging primarily in computational skills in application to relevant problems in the classroom.

In these younger years, the applications may be plugged (with a computational device) and unplugged (without a computational device). "Unplugged" activities are not reliant upon digital devices. At this age group, computational thinking skills can also be meaningfully applied to appropriate computational devices, such as Beebots or Codeapillar, which rely on simple yet readily programmable buttons to customize handheld toys for specific activities and experiments right there on the classroom floor and tabletops. 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.

In older grades (4–12), students apply multiple computational skills to engage in computational practices. Computational practices require students to use a technological tool to (1) automate a procedure or phenomenon, (2) use or create a computational model or (3) collect, analyze or visualize data. An example of students applying computational skills to engage in data practices is described below.

For instance, students might identify a problem with how much food waste is generated during their lunch at school and elect to investigate how to reduce the amount of food wasted.

To learn more about the causes, the teacher might facilitate students to engage in ***data practices***. The students should make decisions about what data to collect and how to collect it. For instance, they could use a computational tool (e.g. chromebook, ipad) to design and administer a survey to collect data about how much food is thrown away, what is part of the food waste, and the reasons students give for throwing away food. Then, the students could compile and analyze the data using computational methods as described below.

As they collect and analyze data, students may define categories of food that were thrown away by their peers (e.g. protein, vegetable, dessert, beverage) using ***abstraction***, filtering trash production by overarching categories.

As students begin to identify trends and consider potential solutions, they might also use ***decomposition***, dividing problems into smaller parts. Does the cafeteria offer certain unpopular foods on certain school days? Is it necessary for the cafeteria worker to add the "mystery" cream sauce to each chicken cutlet served, regardless of student preferences? Decomposition identifies potential "bottlenecks" within a wider process and begins to reduce a wider issue by thinking in terms of specific steps.

Finally, students can ***recognize patterns*** to make conclusions about the food waste in their school. Do the day-to-day patterns mimic each other or differ? Do the patterns of their own lunch room waste potentially correspond to those of other schools?

Depending on the amount of data they collect, students will likely need to ***select a tool*** to help them filter, analyze, or present their data, such as Common Online Data Analysis Platform (CODAP) (Erickson et al., 2019). Later, they might be able to use this tool to analyze food waste in a slightly different situation, for example, at their school's sporting or performance events. Such a tool also can serve as a wider platform to share results and hypotheses.

After analyzing their data, students might use ***data practices*** to communicate their findings through a data visualization presented to students, teachers, and administrators from the school to begin a conversation about decreasing food waste.

To maximize learning, it is also important to reflect. For example, a teacher could provide students with opportunities to reflect on their decisions about how they are engaging in computational practices (e.g. what data are they collecting? How will they organize and analyze it? What tool are they using?) Reflection is an essential step for students to engage in authentic computational problem solving.

As evidenced in this learning sequence, computational skills are applicable to learning in many disciplines. For example, allowing students to collect and analyze data related to a problem they identify is applicable to mathematics Common Core standards to "Represent and interpret data" in grades 1–5 and "Statistics and probability" in grades 6–12 (Common Core State Standards Initiative, 2021). Additionally, providing students the opportunity to interpret their findings in order to write about the data they collect can address common core English Language Arts standards such as to "Read with sufficient accuracy and fluency to support comprehension" in grades 2–5 (Common Core State Standards Initiative, 2021). The lesson sequence integrates inclusive practices because it connects to the interests and experiences of students and provides context for learners to apply math and reading skills, as opposed to an isolated worksheet or assignment.

The Current State of Computing Education

Computing education has expanded rapidly as an educational initiative. In the past eight years, 36 states have passed new policies to establish computing education as a fundamental part of K–12 education (Code.org et al., 2020). New York City has been a leader in this work, launching Computer Science for All (CS4ALL) in 2015, an $81 million investment between public and private partners to ensure all New York City students K–12 receive high quality learning opportunities in computer science and computational thinking (Villavicencio et al., 2018).

While all fifty states have some policy in place promoting computer science (Code.org et al., 2020), the role of computational thinking within these initiatives is ambiguous. The K–12 Computer Science framework (2016), which many states use to develop standards, integrates computational thinking into four of the seven core practices, reasoning that "the most effective context and approach for developing computational thinking is learning computer science; they are intrinsically connected" (2016, p. 69). Consequently, most data points for participation and achievement in computing education are directly related only to computer science, such as enrollment in elective computer science classes in high school and associated test scores (Code.org et al., 2020). Below, we report on the current state of computing education, acknowledging the data relies heavily on computer science as a proxy for computational thinking.

Photo by Allison Shelley for EDUimages

Despite rapid growth in computing education initiatives, disparities in computing education participation and achievement persist today, exacerbated in the past year by inequitable access to tools and learning opportunities due to the COVID-19 pandemic (Martin et al., 2020). Fewer and weaker opportunities for computing learning are presented to students in under-resourced schools, compared to their peers in wealthier schools. Less than half (47%) of high schools in the United States offer computer science, 38% of middle schools, and 26% of elementary schools offer instruction in computer programming (Banilower et al., 2018; Code.org et al., 2020). High poverty schools (26%) and smaller high schools (23%) were less likely to offer computer science learning opportunities than low-poverty schools (46%) and larger high schools (43%) (Banilower et al., 2018; NASEM, 2021). Students that experience marginalization due to their race or ethnicity are more likely to be enrolled in a school that does not offer a computer science course (Flapan, Ryoo, Hadad, & Knudson, 2021). Villavicencio et al. (2018) suggested that schools serving students of color and those experiencing poverty may be prioritizing more pressing challenges such as absenteeism, limited resources, and low achievement in math and reading. The time and resources required for a stand-alone computer science program are not feasible. Students in rural areas, neurodiverse students, students identified as English language learners, and students experiencing poverty are also less likely to have access to computer science resources and opportunities (Code.org et al., 2020).

Among the students that have the opportunity to enroll in computer science, students experiencing marginalization disproportionately elect not to enroll (Villavicencio et al., 2018). For example, students identifying as girls or gender non-binary, particularly girls of color, are disproportionately not electing to participate in computing. In 2019, girls comprised only 29 percent of students that took the AP Computer Science exam. Additionally, students who identified as Black/African American, Native Hawaiian/Pacific Islander, Latinx, Alaska Native/Native American comprised less than 25 percent, even though they account for about 44% of the total student population (Code.org et al., 2020). These statistics reflect the lack of gender and racial diversity in the current CS workforce (National Center for Women in Computing, 2018).

UKBlackTech Stock Photos: https://ukblacktech.com/stock-photos/ licensed under CC BY 2.0

The lack of diversity in computing is not due to the lack of ability nor interest in computing among women and people of color (Google Inc. & Gallup Inc., 2016; Pinkard et al., 2017; Ryoo, 2019). Rather, social and structural barriers including stereotypes about who should excel in computing lead to inequitable tracking of young women and students of color out of computing courses (Margolis et al., 2008; Ryoo, 2019; Wang & Moghadam, 2017). The National Center for Women & Information Technology (NCWIT) has identified counselors as influencers who can advise and encourage students in their education and career aspirations and created the Counselors for Computing (C4C) initiative to provide professional school counselors with information and professional development opportunities that prepare them to expose all students to occupations in technology and computer science (National Center for Women & Information Technology, 2021a).

Broadening Access to Computing with Computational Thinking

While diverse goals can motivate computing integration (Blikstein, 2018; Brennan, in press; Santo et al., 2019), most programs in PreK–12 public education are in alignment with the traditional computer science pipeline, with the primary goal for students to enter STEM and computer science degree programs and workforce (Tissenbaum et al., 2021). As previously stated, the success or uptake of varying computing initiatives is often measured by enrollment in AP Computer Science and/or success on the AP Computer Science exam (Code.org et al., 2020). These traditional motivations for CS education are largely disconnected from the lives of learners who experience marginalization. For example, advanced placement scores couch "success" solely in terms of college credit; while higher education preparedness is certainly a worthwhile goal, AP CS scores are far too discipline-specific (and notably late in the K–12 timeline) to stand alone as a school district's vision of "success" for its students. At best, it represents a barometer but limited metric. At worst, such a singular focus on AP scores reinforces the longstanding perception of CS as meant for a select few, perpetuating the systemic inequity in computing education and the big-tech world more broadly (Vakil, 2018; Washington, 2020). Educators need more robust strategies to attend to computational learning opportunities beyond participation and achievement in elective coursework.

In our increasingly technological society, computational thinking skills and practices are fundamental to virtually every career path and generally in civic life. We argue that all students must be equipped with computational skills, whether they enroll in a computer science course or not. Researchers have suggested that integrating computational thinking into existing curriculum and school subjects (e.g., arts, ELA, math, science, social studies) could increase opportunities for students to be exposed to foundational computing concepts (Lee & Malyn-Smith, 2020; K. Rich, Yadav et al., 2019). 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.

Photo by Allison Shelley for EDUimages

Reimagining Computational Thinking to be More Inclusive

Researchers have identified many innovative strategies to increase access to computational thinking learning opportunities for all students. These strategies are being adopted by schools and districts. Notably, these strategies 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). Examples of such projects have taken advantage of the integrative nature of CT, leveraging computing as a means to engage in more familiar topics, such as music, dance, or storytelling, which may build students' attitudes and confidence toward computer science (e.g., Allen-Handy et al., 2020; Magerko et al., 2016; Pinkard et al., 2017; Weston & Biin, 2013). Other projects have utilized computing as a means for students to interrogate and counteract dominant cultural norms and inequality within society (e.g., Lee & Soep, 2016; Proctor & Blikstein, 2019; Tissenbaum et al., 2017; Vakil, 2014). Re-envisioning computer science pedagogy to be inclusive of student voices; relevant, and meaningful to students' interests and experiences; and connected to social issues within their communities is imperative to break the stereotype of who can and should participate in computer science (Ryoo, 2019).

So, what teaching practices promote inclusive computational thinking and what does inclusive computational thinking look like in a classroom? Several computing education leaders have contributed frameworks

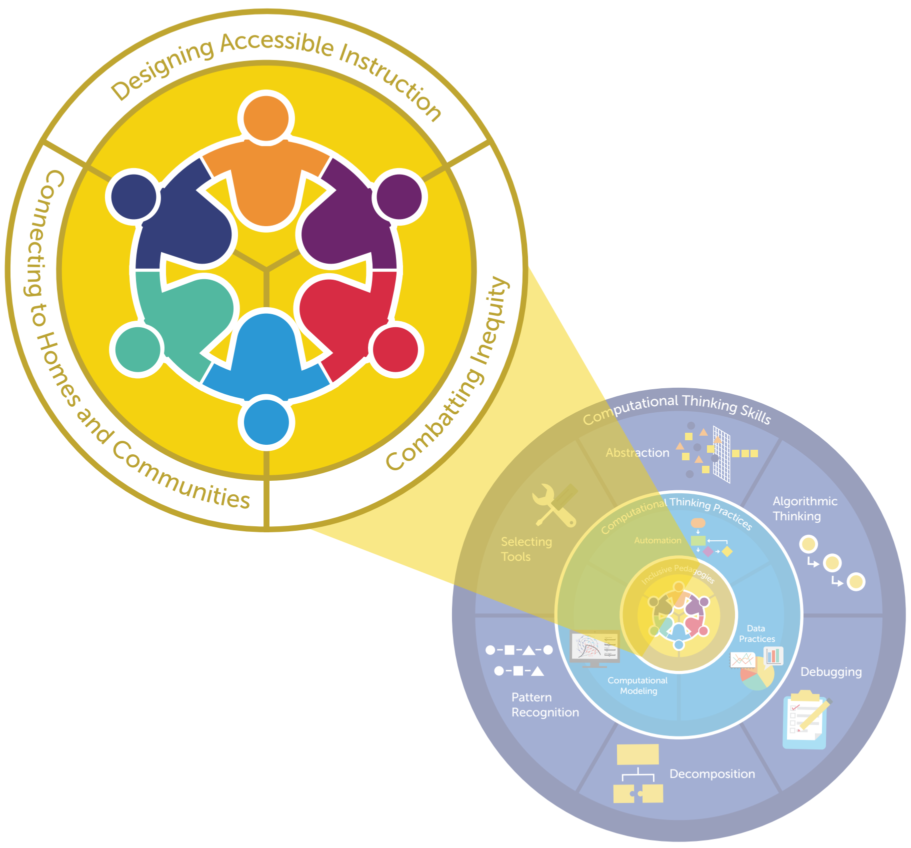


Figure 3. Inclusive pedagogies centered within the computational thinking framework

to operationalize inclusive computing pedagogy (Israel et al., 2017; Kapor Center, 2021; Madkins et al., 2020; National Center for Women & Information Technology, 2021b). In the chart below, we provide examples of inclusive computing pedagogies in the classroom. Inclusive pedagogies, practices, and routines include equity pedagogies such as culturally relevant, responsive, and sustaining pedagogies as well as pedagogies that focus on equal participation of neurodiverse students such as Universal Design for Learning and on practices that can make computing more inclusive and equitable such the recruitment and retention of students that experience marginalization in computing. While many of the practices listed in the chart align to inclusive pedagogies outside of computing, here we provide those practices most relevant to environments in which computing is occurring, whether that is a standalone computer science class or an integrated computational thinking lesson, and highlight examples of using those practices while including computational thinking 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 opportunities that are connected with their homes, communities and interests 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.

Together these pedagogical approaches promote a more inclusive computational thinking classroom environment, life-relevant learning, and opportunities to critique and counter inequalities. Computational thinking provides students with skills and practices that allow them to understand technology and computing processes. This equips students with the background knowledge to discuss and critique how computing and technological systems impact social equality with authentic examples of bias in algorithms and artificial intelligence. For example, students might discuss the implications of gender biases in natural language processing systems such as Gmail Smart Compose (Swearingen, 2018) or higher error rates for marginalized groups in facial and speech recognition programs such as those used in smart speakers and home assistants. Additionally, students can use these understandings to explore large real-world data sets and to identify patterns of inequity between groups of people, such as disparities between demographic groups in diagnosis and recovery from certain diseases. Importantly, these discussions and learning opportunities do not need to wait until high school.

In the example about food waste above, the teacher engages in multiple inclusive pedagogies while teaching upper-elementary aged students. For example, students are **provided with a choice** about the topic they want to explore and select a topic that **connects with their interests and community**. Students are invited to **engage critically** with food waste, a practice that can perpetuate injustices within society and contributes to climate change. Students are also provided with a **choice of computational tool** with which to analyze and present their data. This not only provides students with choice over their learning, but also aligns with the principles of **Universal Design for Learning**. Finally, they **present their findings** and suggestions to their school community, providing an authentic end point for the work they are doing.

Educators should attend to each of the three approaches as they plan and teach lessons, especially related to computing. We reference these practices of inclusivity in case studies throughout the report.

[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).

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)

[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).

What Districts Should Focus On: Integrate Learning and Develop Capacity

Reimaging computing education to be more inclusive also involves revising systems for where and how computing learning opportunities are offered. To reach all learners, particularly those experiencing marginalization, we need to broaden the opportunities that students have to engage in computational thinking beyond elective coursework or extension programs. Today, computational thinking mostly exists as optional content with no formal assessment. It is challenging for educators to prioritize computational thinking as the tested subjects are given priority. Districts, schools, teachers, and other stakeholders must tackle two pressing needs in order to increase inclusive computational thinking learning opportunities into K–12 systems. The first is for teachers to integrate computational thinking into the core disciplinary learning topics already in the curriculum, such as arts, ELA, math, science and social studies. This is a big change from the status quo and this effort must be prioritized by district leaders. Supporting this first need, the second need is for educational leaders to develop systems of professional learning educating teachers on how to integrate computational thinking in their practice and address challenges to inclusivity and equity. We elaborate below.

Need: Integrate 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. Integrating computational thinking can be beneficial both for the expansion of computing opportunities and disciplinary understanding. From a computing perspective, computational thinking integration conceives computing as a tool for understanding other disciplines and topics and, therefore, as an inherently relevant skill. From a disciplinary content perspective, computational thinking can deepen disciplinary learning (diSessa, 2018). Some integrations will place a greater focus on computing while others will place the greatest focus on disciplinary concepts; ideally an integration balances computing and disciplinary learning to promote both (CodeVA, in press). Then, CT can become a value-add to disciplinary learning and not an add-on to an already overscheduled school day. We discuss three strategies for educators to integrate computational thinking into disciplinary learning to promote computational thinking learning opportunities for all students: leverage synergies between disciplinary learning and computational thinking, develop computational thinking skills in the younger grades, and promote student agency and purpose. Illustrative examples of each strategy follow.

Strategy #1: Leverage Synergies Between Disciplinary Learning and CT

School districts and teachers need to identify the best ways to integrate CT into core subject areas. There are robust examples and frameworks of computational thinking integration in science and math (Basu et al., 2016; K–12 Computer Science framework, 2016; Lee & Malyn-Smith, 2020; Malyn-Smith et al., 2018; Nesiba et al., 2015; Sengupta et al., 2013; Settle et al., 2012; Waterman et al., 2020; Weintrop et al., 2016). For example, Project GUTS is a middle school science curriculum in which students' leverage computational models to explore scientific concepts, such as population ecology, water conservation, and epidemiology (Lee et al., 2011). In middle and high school mathematics, Bootstrap offers a curriculum in which learners program a video game or analyze data, integrating concepts of algebra, physics and data science (Schanzer et al., 2015). There are also compelling examples of CT integration in arts, ELA and social studies. The

Bootstrap: Data Science curriculum offers data related to historical events, providing students opportunities to interpret primary source datasets. Additionally the Integrated Computational Thinking (iCT) project has worked with teachers to identify synergistic CT integrated practices in arts, ELA and social studies, such as critical analysis of computational texts and the creation of computationally enhanced and inspired art (Integrated Computational Thinking, 2021).

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). Examples of integrated computational thinking lessons are described below the chart. While the examples are integrated into middle school and high school science classes, prompting questions for each example promotes readers to make connections between the inclusive pedagogies, computational thinking practices and their own classroom across many disciplines.

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

Example: Integrating Computational Modeling into Middle School Science for Bilingual Students

Students in Ms. Silfa's class designed computational models in Scratch about hurricanes, a topic relevant to the lives of many students. Throughout the unit, there were many opportunities for students to speak and learn in multiple languages and connect with their families and communities.

Think about:

- **What is Translanguaging? Are there opportunities for students to use translanguaging in your classroom?**
- **What opportunities do students have to connect to their families and communities? Where can you make these connections in your classroom?**
- **Are there topics, processes, or phenomena in your classroom that students could automate in Scratch?**

Inclusive Pedagogies	Computational Modeling	Automation

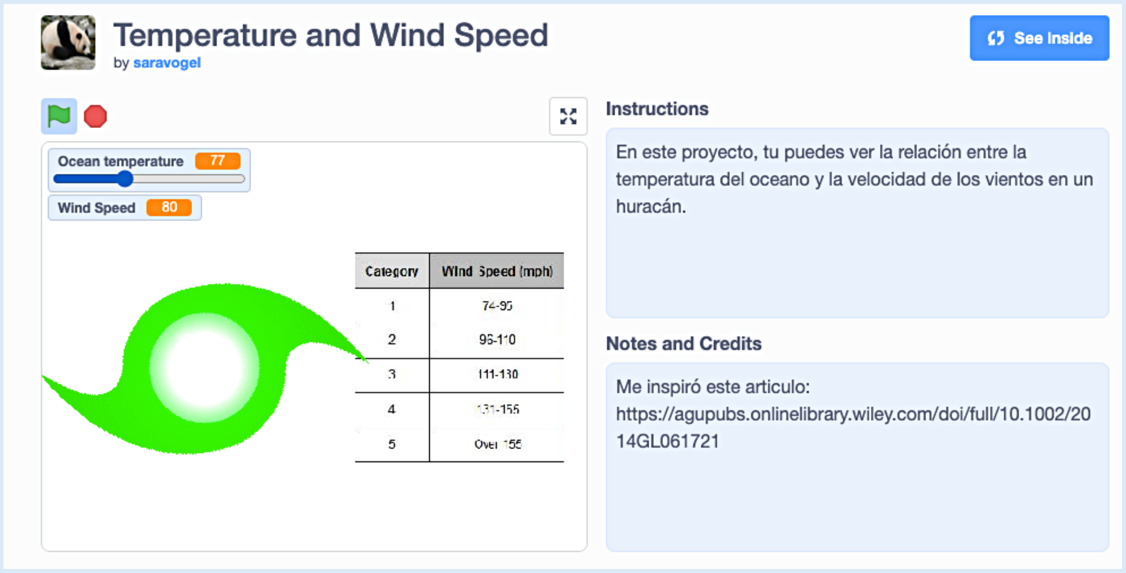
In New York City, Karen Silfa integrated computational modeling in a middle school bilingual science class in a curricular unit about Huracán María (Hurricane Maria), a topic that impacted the lives of most students (James et al., 2020).

As she designed and implemented the unit, she was intentionally experimenting with an approach called Translanguaging Pedagogy, which encourages students to talk, read and write using all of their language practices, including oral and written language in English, Spanish, and beyond; gestures, code, objects, and models. Teachers build on all of students' language practices, including those practices that schooling tends to marginalize (García et al., 2014).

María was a severe category 5 hurricane that devastated Puerto Rico in September 2017. Many of the learners in Ms. Silfa's class were from the Dominican Republic and had arrived in the United States less than three years ago. More than half of the students were Multilingual Learners / English Language Learners and 90% of the students qualified for free and reduced price lunch. They had experienced the damaging effects that hurricanes and tropical storms had on their home countries and were very concerned about the impacts of Huracán María.

Ms. Silfa identified an opportunity to integrate computational thinking and inclusive pedagogies in the Next Generation Science Standards for grades 6–8. She integrated the science practice of developing and using scientific models (in this case, **computational modeling**) into the disciplinary topic of weather and climate and included discussions about cause and effect. The learning activities included:

- Conducting background research about Hurricane María in both Spanish and English to identify short-term and long-term effects of the disaster. Students interviewed Puerto Rican activists and their parents to learn more about the impacts of hurricanes.
- Exploring how **computational models** can be used to predict the path and severity of hurricanes (Figure 4). Students considered the purpose of modeling and different methods for creating models (e.g. physical, computational).
- **Automating** one effect of hurricanes in Scratch by remixing teacher-created projects. As they worked, students tracked their progress using a multilingual checklist.
- Presenting models to students' parents, educators, administrators, and professionals from the Association of Latinos in STEM.



Category	Wind Speed (mph)
1	74-95
2	96-110
3	111-130
4	131-155
5	Over 155

Instructions

En este proyecto, tu puedes ver la relación entre la temperatura del oceano y la velocidad de los vientos en un huracán.

Notes and Credits

Me inspiró este articulo: https://agupubs.onlinelibrary.wiley.com/doi/full/10.1002/2014GL061721

Figure 4. Screenshot of a computational model that students explored in Scratch. Translated instructions: In this project, you can see the relationships between the temperature of the ocean and the wind speed of a hurricane (James et al., 2020).

Students had opportunities to connect previous experiences with hurricanes to science concepts and computational modeling. Importantly, there were inclusive pedagogies embedded throughout, including opportunities for students to translanguage. Additionally, parents and other Latinx community members contributed to students' learning throughout the unit. This unit illustrates the idea that computer science learning opportunities can engage syncretic literacies (Gutiérrez, 2014)—in other words, can bring together literacies from across domains including students' communities, computing, and school disciplines in ways that spark new conversations and new literacies.

Teaching Tips and Resources:

Tip	Resource
Create models for students to remix	Short term effects of Hurricane María Scratch Studio Long term effects of Hurricane María Scratch Studio
Provide a checklist for students to track their progress	Multilingual checklist for students to self-regulate their progress
Create authentic opportunities for students to share their work outside of the classroom	Document to support students with making public presentations
Get inspired by Ms. Silfa's unit	Description of the unit Video case study of this unit

Table 4. Teaching tips and resources to learn more about how Ms. Silfa integrated computational thinking into middle school science.

Example: Exploring Health Disparities Using Data Science in High School Biology

Students in Ms. Lindahl's class analyzed demographic disparities in rates of cancer diagnosis, treatment, and outcomes.

Think about:

- **What topics that you teach are connected to social justice?**
- **Is there data that could reveal societal inequities related to a topic you teach?**
- **How could you encourage students to speak about and improve injustices?**

Inclusive Pedagogies	**Data Practices**

Using data science to explore health disparities in science is something that Amy Lindahl integrated into a unit on cellular division and cancer (Lindahl, 2012). She had taught the topic for years, yet something seemed to be missing. Ms. Lindahl realized that discussing the science of cancer, without acknowledging the harm it has caused students and their families was a missed opportunity.

Ms. Lindahl listened to the heartbreaking stories of her students, some of which illustrated with disturbing clarity how cancer disproportionately affects Black, Native American, and Latinx communities. She felt she may be perpetuating these disparities by not acknowledging them in her classroom. She decided that she needed to make space to integrate these critical discussions into science.

First, Ms. Lindahl introduced students to how societal in-equities apply to cancer. Students engaged in the following activities:

- **Is Cancer an Equity Issue?** Investigate cancer risks, to what degree these risks are preventable, and what demographic comparisons reveal about cancer screenings, diagnosis, and treatment
- **Cancer Case Studies** Explore case studies, such as the true story or Marissa Thomas (Figure 5), to help students imagine how the inequities shown in cancer data impact individuals, families, and communities.

Figure 5. Marissa Thomas's true story as a breast cancer survivor is featured in one of the case studies from Ms. Lindahl's class. Ms. Thomas is a co-founder and CEO of the activist group The Breast of Us.

- **The Social Determinants of Health** Use this public health framework to consider how structural factors lead to cancer inequities

Then, students explored cancer statistics and disparities between different groups in the following lessons:

- **Identify, Intersectionality and Cancer.** Manipulate data on how aspects of identity connect with cancer risk, health care experiences, treatment, and more
- **Cancer Inequities in our Community** Use databases to discover correlations between cancer outcomes and local factors

Finally, students designed strategies to decrease disparities in cancer rates in the following lesson sequence:

- **Local Health Activism.** Learn from local health activists about ideas for student cancer equity actions
- **Develop Cancer Equity Actions.** Design an action to reduce cancer inequities in their community
- **Equity Action Share Out and Reflection.** Share out their cancer equity actions and reflect on needed local and national structural reforms

In this example, students had opportunities to explore data that illustrated societal inequities related to disciplinary learning. In order to design a related learning sequence, teachers need to locate a data set appropriate for students, and support them to visualize and make sense of these large data sets. Finally, equipping students to acknowledge and grapple with inequities and injustice the data reveals is essential. This learning sequence is currently being revised and updated, and will be available at the Science Education Partnership website, an initiative of the Fred Hutchinson Cancer Research Center. Below, we suggest some related resources to support teachers and students with these practices.

Teaching Tips and Resources:

Tip	Resource
Locate data relevant to disciplinary learning and students' lives and interests	Health Equity Tracker CDC Epi Info
Prompt students to evaluate and think critically about data	AACR Cancer Disparities Progress Report 2020 Analyzing Health Disparities (Learning for Justice)
Facilitate a discussion about societal inequities	Resources for Having Social Justice Conversations

Table 5. Teaching tips and resources to learn more about how Ms. Lindahl integrated computational thinking into high school biology.

Strategy #2: Develop Computational Thinking Skills in the Younger Grades

It is particularly important for young children to have opportunities to engage in computational thinking in order to develop an interest in and a foundational understanding of computing. As educators seek to integrate computational thinking into all grade bands, including younger grades, they must make strategic decisions as to what are appropriate learning goals for different age groups, and approximate how these learning goals are connected as part of a learning progression. Several researchers have proposed frameworks and activities for computational thinking in PreK–12 education (e.g. Angeli et al., 2016; Duncan & Bell, 2015; Lee, et al., 2014; Seiter & Foreman, 2013; Weintrop et al., 2016). However, limited research evidence is available to inform learning trajectories that drive curricular decisions (K. Rich, Strickland et al., 2019; K. Rich et al. 2017; 2018).

Understanding developmentally appropriate computational thinking practices is particularly challenging in the younger grades (PreK–5) because traditionally computing education does not start until middle school. While there are many computational devices designed to support young learners in learning to program (e.g., Beebots and Scratch Jr.) computational thinking skills are relevant to much more than just programming. Emerging research suggests that there may be benefits to leveraging computational thinking skills as a general problem-solving approach, especially in the younger grades. These skills can be applied to plugged (with a computational device) and unplugged (without a computational device) activities across any discipline in the classroom. Yadav, Ocak and Oliver (under review) have suggested that computational thinking facilitates metacognition, the ability to monitor and control cognitive processes, when solving problems.

The following examples highlight two projects that have co-designed learning experiences with teachers of young children over several years. Both projects emphasize the development of computational thinking skills (e.g. abstraction, decomposition, pattern recognition) as a problem solving approach. These skills provide students with language and processes to identify how to solve a problem and discuss why that approach is appropriate.

Example: Integrating Life-Relevant Computational Thinking into Preschool

Preschool children connect computational thinking skills to learning and everyday activities such as getting ready for school, using playdough, and going to the grocery store.

Think about:

- **How do the CT integrated preschool activities connect school and home?**
- **How might young children engage in both plugged and unplugged computational thinking skills?**

Inclusive Pedagogies	**Problem Decomposition**	**Abstraction**

Digital Promise has partnered with teachers and families to co-design integrated CT activities (both unplugged and plugged) for preschool children.

To intentionally connect CT skills to young children's (ages 3–5) experiences, homes, and communities, the co-design team included teachers and families from culturally diverse public preschool programs. Together, learning scientists, curriculum designers, and media developers identified life-relevant CT skills that can be meaningfully linked to math and science learning in preschool classrooms and can strengthen early learning more broadly.

In the following two activities, young children were able to engage in the CT skill of **problem decomposition**; they learn to break down problems or tasks into smaller parts, which makes solving the problems or tasks more manageable. This process also helps children work together with classmates to solve problems.

- Getting Ready for School. Children break down an everyday routine into smaller parts and make a mini-book to tell the story of their very own routine. Breaking this larger task (getting ready for school) invites children to independently identify smaller tasks (e.g., brush teeth, get dressed) and helps them more easily or efficiently complete the routine. Making the mini-books about their routines also gives teachers and classmates an opportunity to learn about and celebrate each other's unique stories and home traditions.
- Playdough Workshop. Children identify the large task of making a playdough tree and can work with a classmate to identify and complete the smaller tasks together, such as making a trunk, leaves, and apples. This activity also links to math learning as children are invited to notice the shape of the leaves or count the number of apples on the tree. Examples of playdough figures children might decompose are in Figure 6.

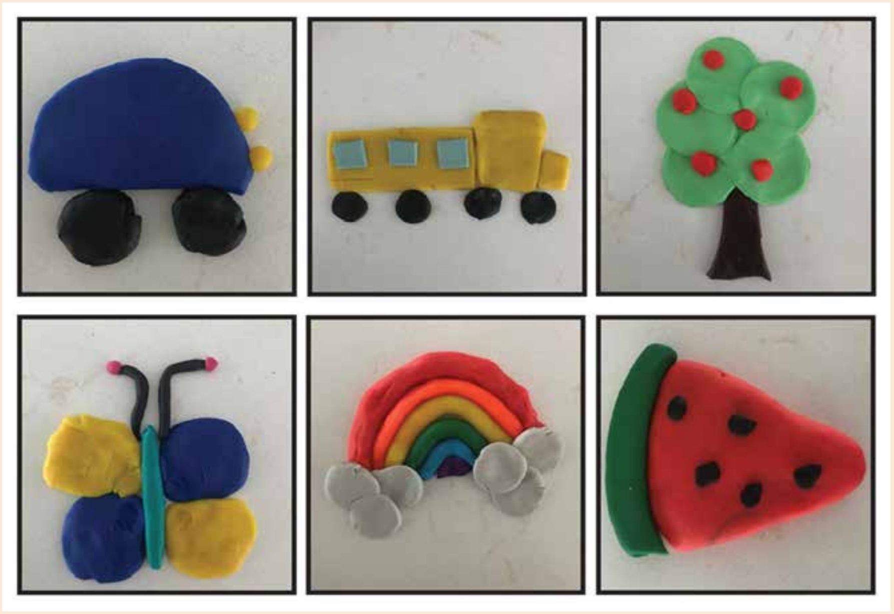


Figure 6. Examples of playdough figures. Young children can use problem decomposition to identify and complete smaller tasks of making playdough figures.

Another CT skill that aligns with common preschool interests and activities is **abstraction**. When children sort objects based on observable characteristics, they are encouraged to highlight necessary information while ignoring irrelevant or unnecessary information, as described in the following activities:

- Grocery Store. Children engage in sorting and labeling related to food groups. Children are given a large set of different foods and encouraged to decide how to sort the foods into separate baskets and then label accordingly, using pictures or simple words. For instance, they may decide to sort and label by type of food such as fruits and vegetables. This activity promotes the skills of hiding details and paying attention to only the most important information, in this case the type of food. They can also engage in guided discussion about how sorting food in a particular way can be helpful to achieve their goals. For instance, sorting fruit by type (e.g. citrus, melon, berry) can help customers easily find what they need at the market.
- STEM-tastic Adventures app includes a game known as Better Building that can help children practice sorting and labeling. Through their gameplay, children observe and sort objects by color, shape and size labeling the sorted groups to help the robot build structures more efficiently.

Tip	Resource
Make connections with childrens' homes and families to design CT opportunities that extend in and out of the classroom	Preschool Computational Thinking
Get inspired by some PreK CT activities	PreK CT activities
Explore a CT app for PreK	STEM-tastic adventures app

Table 6. Teaching tips and resources to learn more about integrating computational thinking into PreK.

Example: Integrating Computational Thinking into Elementary English Language Arts

Students in Ms. Jenkins English language arts class used computational thinking skills such as problem decomposition and abstraction to identify the theme of a story and compose paragraphs.

Think about:

- **How could computational skills be applied to practices that you teach?**
- **Could explicitly teaching computational thinking terminology provide students with language and processes to approach problems in your classroom?**

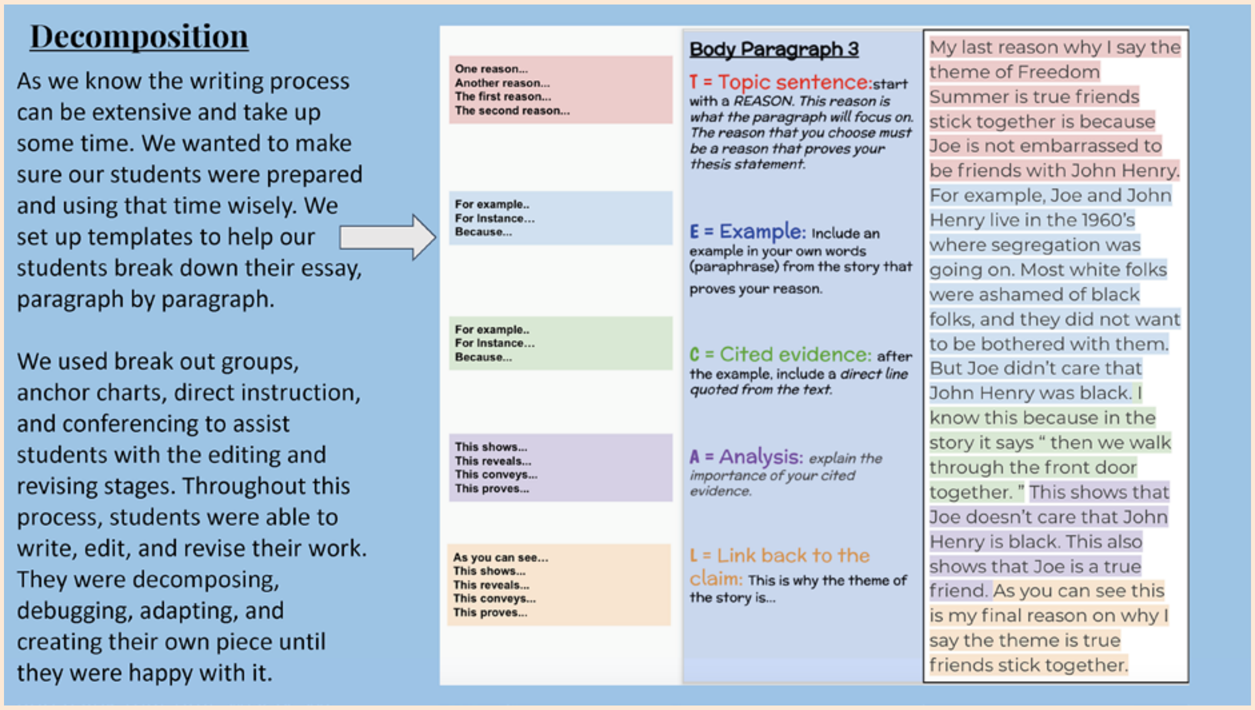
Inclusive Pedagogies	**Problem Decomposition**	**Abstraction**	**Pattern Recognition**	**Debugging**	**Automation**

Ms. Jenkins is a 4th grade teacher in Staten Island, a borough of New York City. She intentionally introduced computational thinking with her students as a problem solving approach, particularly relevant to her English Language Arts instruction. She integrated computational thinking vocabulary in her instruction (e.g. slide presentations, worksheets/activities) to facilitate students to identify where computational thinking concepts connected to reading and writing practices. For reading, she made the following connections:

- In order to Identify the theme of a story, teachers used **problem decomposition** to break down the parts of the story to consider what happens in each smaller part. Then, they used abstraction to consider what aspects were the most important. Finally, they thought about what the bigger ideas were.
- Based on students' prior knowledge or experiences with other literature, they **recognized patterns** to make predictions about upcoming events. For instance, how a character reacted to a certain situation.
- When presented with unknown or unfamiliar words, students **decomposed** these words into chunks or other smaller portions that were easier to figure out. These smaller chunks included common beginnings and endings, vowel combinations or root words that were familiar to students. This often comprised the first steps of an algorithm students followed for decoding unknown words.

For writing, she made the following connections:

- Ms. Jenkins supported students to break down, or **decompose**, the contents of each paragraph to ensure they included a Topic sentence, Example, Evidence, Analysis, and Link back to the claim (Figure 7).



Decomposition

As we know the writing process can be extensive and take up some time. We wanted to make sure our students were prepared and using that time wisely. We set up templates to help our students break down their essay, paragraph by paragraph.

We used break out groups, anchor charts, direct instruction, and conferencing to assist students with the editing and revising stages. Throughout this process, students were able to write, edit, and revise their work. They were decomposing, debugging, adapting, and creating their own piece until they were happy with it.

One reason...
Another reason...
The first reason...
The second reason...

For example..
For instance...
Because...

For example..
For instance...
Because...

This shows...
This reveals...
This conveys...
This proves...

As you can see...
This shows...
This reveals...
This conveys...
This proves...

Body Paragraph 3

T = Topic sentence: start with a *REASON. This reason is what the paragraph will focus on. The reason that you choose must be a reason that proves your thesis statement.*

E = Example: Include an example in your own words (paraphrase) from the story that proves your reason.

C = Cited evidence: after the example, include a *direct line quoted from the text.*

A = Analysis: *explain the importance of your cited evidence.*

L = Link back to the claim: This is why the theme of the story is...

My last reason why I say the theme of Freedom Summer is true friends stick together is because Joe is not embarrassed to be friends with John Henry. For example, Joe and John Henry live in the 1960's where segregation was going on. Most white folks were ashamed of black folks, and they did not want to be bothered with them. But Joe didn't care that John Henry was black. I know this because in the story it says " then we walk through the front door together. " This shows that Joe doesn't care that John Henry is black. This also shows that Joe is a true friend. As you can see this is my final reason on why I say the theme is true friends stick together.

Figure 7. Class algorithm for writing including topic sentence, example, cited evidence, analysis and link back to claim.

As students completed their writing pieces, they enhanced their work through a process Ms. Jenkins named **debugging**. When debugging their work and their classmates' work, Ms. Jenkins made sure students were aware that they were using computational thinking skills such as **pattern recognition** to look for common errors in punctuation and **abstraction** to identify points in their writing that could use more support or detail.

Once students produced a completed writing piece, they were provided with options to elicit their creativity and apply their computational thinking skills through the use of technology.

One of the options presented to students was to use **automation** to design and build a Scratch coding project to share information based on their learning and research.

Ms. Jenkins integrated computational thinking terminology consistently throughout the school year and applied the terms to different lessons. Students were then able to pick up these computational thinking skills as a problem solving approach in English language arts and beyond.

Tip	Resource
Identify ways to highlight computational thinking with plugged and unplugged activities.	[Sample Lesson Plan- 4th Grade Main Idea]()
Use relatable stories and texts to help students apply computational thinking to deepen their understanding of character's differing viewpoints.	[Sample Lesson Plan- 4th Grade Point of View]()

Table 7. Teaching tips and resources to learn more about how Ms. Jenkins integrated computational thinking into elementary English Language Arts.

Strategy #3: Promote Student Agency and Purpose

It is important to remember that not all opportunities to integrate computational thinking will provide students with the same opportunities to build computational skills in ways that are equitable, including ways that promote student agency and deepen disciplinary learning (Coenraad et al., 2021; Waterman et al., 2020). Drawing on the previous example, a student could engage in data practices if the teacher instructed them to tally how much trash is thrown away and provided the methods for both data collection and data analysis using a computational tool. The student collects and analyzes data, but is not well positioned to use data practices for future purposes. Alternatively, and described in the example, the teacher might encourage learners to design a data collection method based on their own curiosities about trash in the lunchroom and select their own computational tool for analysis. Providing student-centered learning experiences, so that students are driving decisions about what tool to use, how to use it, and for what purpose, provides them with opportunities to gain experience, autonomy, and confidence in computing they can take outside of the classroom.

To illustrate this point, we have developed a spectrum to illustrate that integrations of computational thinking exist on a continuum that ranges from only enhancing learning to a true transformation of the learning experience (Terada, 2020). The spectrum builds on the concept that CT can be integrated at different levels (CodeVA, in press; Lee et al., 2021; Waterman et al., 2020). By "enhancement," we mean students are instructed how to use data, a program, or code and by "transformation" we mean students make their own decisions about learning goals and the computational tools and processes they will use to achieve them. In the middle ground, students may modify a program that they are provided with or make decisions with limited scope.

In their first year integrating computational thinking, a teacher might feel most comfortable beginning to integrate on the enhancement side of the spectrum, slowly providing students more agency as they become more comfortable with tools and practices to facilitate computational thinking. While this type of lesson enhancement is a place to start with computational thinking integration, it is certainly not an endpoint. Within and between grade levels, the goal should always be for teachers to start where they are, but move towards transformative learning. The affordances of computational thinking provide students agency to explore questions and design solutions that are connected to students' interests and experiences, increasing inclusion and engagement. For example, students can interact with complex real-world problems, such as analyzing large data sets or modeling systems. Research has illustrated that sustained professional learning about computational thinking promoted teachers to develop student-centered lessons (Ketelhut et al., 2020).

In Table 8, we illustrate how these depths of integration can occur in each of three domains: automation, computational modeling, and data practices. This spectrum does not apply skills in isolation, and therefore is only applicable to students engaging in computational practices, approximately grades 5–12, as described in the framework for integration.

Computational Thinking Practice	Integration Opportunity		
	Enhancement ------------------------------------►		Transformation
Automation	Use a provided computer program as instructed to achieve a specific result.	Select a computer program to solve a problem.	Design a computational artifact of choice (e.g. computer program, robot, app) in response to an authentic problem that students identify.
Computational Modeling (Lee et al., 2021)	Use a simulation to explore relationships between parts of a system.	Analyze ("Decode") a simulation to reveal the relationships between parts of a system and how the simulation is like or unlike the real world.	Modify or create a computational model using code to include cause and effect relationships between parts of a system.
Data Practices	Collect a small, predetermined amount of data. Students create a data visualization as they are instructed, such as a bar graph.	Use an online survey platform to collect data and use a spreadsheet to create a data visualization.	Analyze a large secondhand data set using a computational tool such as CODAP. Students make their own decisions about the best filter, make sense and visualize the data.

Table 8. Computational thinking practices operationalized as a spectrum of transformation.

Example: Students Identify Social Justice Community Need to Inform App Design

Students in Mr. Scott's class designed apps in response to concerns of local community members to promote social justice.

Think about:

- **What is an empathy interview? How did students use these interviews with community members to inform the design of the app?**
- **Mr. Scott says, "whatever they wanted it to be, whatever they wanted it to do, was up to them." Is this sentiment ever applicable in your classroom?**

Inclusive Pedagogies

Automation

Kennan Scott, a middle school teacher in Oakland, California leveraged design thinking and Agency by Design to provide a platform for his students to expose and address injustices faced by individuals in the community using inclusive computing pedagogies. By using empathy interviews with family and community members, students in Mr. Scott's middle school class designed apps that supported the safety of people in their community.

In order to develop meaningful contexts for design, Mr. Scott decided to focus on social justice causes or issues that revolve around community needs.

- Students first talked to different members of the community about issues in the community and their needs. They spoke with a member of their family (e.g. parents, grandparents), somebody from the middle school community (e.g. staff, students) and an individual from the broader community (e.g. somebody from the local market or church).
- Based on the discussions, students designed a mobile app using automation through programming to address a problem that they identified.
- Students had the agency to select the topic they wanted to cover. Mr. Scott explained, "whatever they wanted it to be, whatever they wanted it to do, was up to them."

Through this process, students gained perspective about challenges in the community and were able to address the needs of real people through the design of an app. Mr. Scott explained, **"It drove home the idea that this is a transformative act, that you can use the creativity and power of computational thinking to actually make a change."**

Based on what they learned in their interviews, students created a variety of apps. For example:

- **Quick access to help**. Students developed an app to quickly access help on their phone if something was amiss when they were out, such as walking home from their friend's house. The app was designed as four quadrants on the screen that students could press to signal different actions. One quadrant sounded an alarm, others called emergency contacts, another called 911. Based on the situation, students could quickly take action to stay safe.
- **Safe routes home from school**. Another app that students developed was a map that provided the user with up-to-the-minute crime reports data overlaid on Google Maps. Therefore, students could re-route their way home from school to home based on reported incidents that had been identified in a moment.

Mr. Scott recalled that the completion and success rate for this project was much higher because students were engaged. Starting with conversations with community members, each student could tell the story of a person that they were designing for. Therefore, students produced high quality apps because they were able to dive deeper into the cause and reasoning for what they were doing.

Tip	Resource
Students conduct empathy interviews with members of the community to inform design choices.	Liberatory Design Card deck
Students identify a community relevant problem to design a computational artifact.	Design thinking equityXdesign

Table 9. Teaching tips and resources to learn more about how Mr. Scott used design thinking and social justice in app design.

Example: Promoting Student Agency in Data Practices

Students in Ms. Bibb-Fox's class wrangled large data sets to inform their own inquiries about the ongoing COVID-19 pandemic.

Think about:

- **Where can you locate data that is relevant to students' lives and interests?**
- **How can you support students to pose their own questions that can be answered with data and make sense of large data sets?**
- **Do you make space for students to consider bias in a data source or identify misleading graphics?**

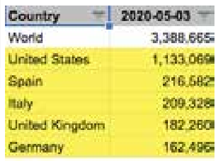
Inclusive Pedagogies	**Data Practices**

Jessica Bibbs-Fox, a teacher in Compton Unified School District, promoted student agency in the design of a project-based learning experience at the onset of the COVID-19 pandemic in her virtual middle school science class. Just a few weeks after the nationwide emergency switch to remote learning in April 2020, Bibbs-Fox designed a unit in which students examined the accuracy of information related to the ongoing COVID-19 pandemic .

First, students posed their own questions based on their curiosities about the pandemic. However, many of the initial questions students developed could not be explored using data. Bibbs-Fox prompted students to revise their questions so they could be answered using data. For example, one student initially posed the question, "Why is the coronavirus so deadly?" which was revised to "Compared to other coronaviruses, is COVID-19 resulting in more deaths?"

Next, students used raw data from various organizations such as Our World in Data and the Center for Disease Control (CDC) to inform their questions, all the while considering the relevance, validity, and bias of each data source. Bibbs-Fox encouraged students to use the Common Online Data Analysis Platform (CODAP) to analyze their data, which offers student-friendly scaffolding for data analysis and visualization. She provided a graphic organizer (Figure 8) to encourage students to use specific data moves to analyze their data (Erickson et al., 2019). Bibbs-Fox reported that this activity helped students become more confident working in spreadsheets and were proud of their work.

Question: What country has the highest number of confirmed cases?

Data Move(s) Dive deeper into the data set by combining moves	**Sketch or describe how you organized or illustrated the data:**	**What does this tell you about the relationships/patterns within your dataset?**	**What new questions do you have?**
Sorting	Sorted from most recent date by highest to lowest 	The United States has the highest number of cases, followed by Spain, Italy, the United Kingdom, and Germany.	What continent has the highest number of cases? (See analysis below)

Country	2020-05-03
World	3,388,665
United States	1,133,069
Spain	216,582
Italy	209,328
United Kingdom	182,260
Germany	162,496

Figure 8. Students analyzed data sets using data moves (Erickson et al., 2019). Resource by Digital Promise, Collecting, Analyzing, and Evaluating Data

Finally, students communicated their findings to community members, particularly those who were uncertain about safety practices or skeptical about social distancing. To discourage students from reusing data visualizations that were created from other sources, Bibbs-Fox presented misleading graphics related to the pandemic and asked her students to spot errors. Students were then motivated to design an original infographic, public service announcement, or magazine article. They shared drafts of their visualizations with members of the community and modified the design based on their interpretations.

In reflection, Bibbs-Fox expressed that this was a very valuable learning experience for students. She explained, "This project relies on questions that do not have answers readily available to them. Students have to rely on their skills to be successful."

Tip	Resource
Locate data relevant to disciplinary learning and students' lives and interests	Our World in Data
Scaffold learning opportunities for students to make sense of large data sets	Data Practices: Computational Thinking for Next Generation Science
Utilize student-friendly data analysis platforms	CODAP Bootstrap

Table 10. Teaching tips and resources to learn more about how Ms. Bibbs-Fox designed her high school data science class.

Need: Develop Capacity for Computational Thinking

There are strong examples of computational thinking, such as the examples in the previous section, frequently 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 this section, we discuss strategies to scale and sustain computational thinking integration within districts and schools, including promoting shared leadership between districts and schools and designing professional learning experiences for pre- and in-service teachers. For each strategy, we provide an overview followed by illustrative examples of the strategy in action.

Strategy #1: Promote Shared Leadership Among Districts, Schools and Teachers

While statewide computing initiatives continue to grow with great success (Code.org et al., 2020), state policymakers are too removed from school-level contexts to make implementation recommendations at the grain size that educators need. The role of district leaders is essential in establishing the three "Cs" in system-wide K–12 pathways: **consistent** across classrooms, **cumulative** from year to year, and **competency-based**. Yet, we believe that district leaders cannot do this work alone and must partner with building and teacher leaders at the school level to identify and develop computational thinking pathways. This shared leadership allows for the development of consistent, cumulative, and competency-based pathways that take into account school needs and have teacher buy-in.

Due to the inherent inequity of implementing computational thinking, often as an after-school activity or extension in advanced classes, Digital Promise recommends a district-wide competency-based approach to computational thinking implementation. Burke, Roschelle, et al. (2019) described a progression of CT integration in districts and schools ranging from **tools**, to **themes**, to **competencies**. With tools, districts provide technology and programs to teachers such as Scratch, Code.org, and Project Lead the Way. With themes, they embed these tools into integration approaches such as maker learning or STEAM. With competencies, districts focused on student knowledge and abilities, such as algorithms, data, and computational modeling. Competency-based pathways are an effective strategy to define computational thinking while providing educators flexibility to select appropriate tools and methods of instruction for their students.

In our CT pathways project[5], we have partnered with three school districts from across the United States to iteratively develop and implement inclusive K–12 computing pathways. These districts represent unique contexts and challenges. Iowa City Community School District is a college-town district serving 14,000 students and is rapidly becoming more urban and diverse. Indian Prairie School District is a suburban school district serving 28,000 students in the Naperville, Aurora, Bolingbrook, and Plainfield communities outside of Chicago. Talladega County Schools is a 7,000-student district in rural Alabama.

Prior computing experiences offered across these districts were typical of many districts, where offerings are fragmentary with tool or theme orientation, rather than consistent, cumulative, and competency based. For instance, previous computing opportunities in each district were extracurricular (e.g., robotics clubs), tool-based (e.g., makerspaces, kits), or elective (e.g., computer science courses in high school). This fueled

[5] Developing Inclusive K-12 Computing Pathways for the League of Innovative Schools, supported by the National Science Foundation under Grant No. 1837386. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

the equity challenge identified by each district: the current offerings were excluding a demographic of students experiencing marginalization in computing. While each district targeted different demographics of students as the equity challenge in their district, all three focused on dismantling barriers to computing education participation. The districts engaged in our project to intentionally create inclusive opportunities for computing that disrupt educational systems that reproduce the existing and persistent inequities in computing education.

Within the districts, **shared leadership** included **district leaders with a strong commitment** to the initiative and inclusivity mission working closely with teacher leaders at each school building who then guided **educators from across the district** throughout **both the creation and implementation** of the CT Pathways. These teams were championed by superintendents, led by district-level curriculum and technology leaders, and included teachers, teacher leaders, building administrators, and curriculum specialists. Importantly, representatives from across these groups were instrumental throughout the entire pathway creation process; they were not just stakeholders brought in to give comments a few times or pilot a process developed by others. Once the pathway was developed, the building-level educators played a pivotal role in implementation. Participating in the development team and process provided professional learning opportunities for the participating teachers and garnered buy-in from them to be champions within their school buildings. Below, we spotlight how the district and schools developed shared leadership when creating an inclusive K–12 computing pathway in Talladega County Public Schools.

Example: Designing K–12 Computational Thinking Pathways in Rural Alabama

Talladega County Schools shared leadership among district, school, and teacher leaders throughout the development of a K–12 computational thinking pathway that is individualized for their district and student needs.

Think about:

- **What group of students who experience marginalization in computing should your district prioritize?**
- **What is your district vision for computational thinking?**
- **How can leadership be shared among district, building, and teacher leaders where you work?**

Talladega County Schools is a rural district in central Alabama comprising seventeen schools and serving around 7,000 students. Within the district, 71% of students have been identified as low income, 33% of students are Black, and 2% of students are Latinx. Over the past three years, the district has worked to develop the capacity to integrate CT in K–12 disciplinary learning. Dr. Brooke Morgan, Talladega's Coordinator of Innovative Learning worked closely with superintendent Dr. Suzanne Lacey and the Digital Promise team to consider how the district could develop activities, curricula, and assessment that appealed to a broader range of students, with a focus on promoting inclusivity among students from low socioeconomic households and, more broadly, female students. They leveraged existing teacher leaders across schools to assemble a pathway development team that shared leadership between district and building personnel with a three-fold process:

Establishing the Foundation for Computational Thinking. First, Talladega clarified its vision for its own K–12 computing pathway. Over a period of two months, a pathway development team of teacher and district leaders met to identify the district's vision, strengths, interests, and existing resources—as well as its gaps per grade level and across schools. As a rural district, the Talladega community valued computing learning opportunities for students to thrive in the technologically evolving workforce and promote economic opportunity in the region. They resolved to integrate CT in core disciplinary learning (e.g. science, ELA, math) in each grade K–8.

Developing K–12 Computational Thinking Pathways. Next, the Talladega pathway development team defined what new learning opportunities would be created across grade levels, courses, and schools. Here, Talladega tapped into Alabama's Digital Literacy and Computer Science (DLCS) standards and relevant disciplinary standards to specifically identify CT competencies (e.g. data practices, algorithm) that were in alignment with their existing initiatives. Then, they defined opportunities to integrate CT by grade level and within the context of specific subjects. Talladega developed a "competency map" linking CT competencies to relevant standards and curricular specific activities and resources within disciplinary learning (e.g., Storytelling with Scratch in 4th grade ELA), as illustrated in Table 11.

Grade K: By the end of Grade K, what will ALL students know and be able to do?				
Relevant Standards (From Alabama DLCS)	What do the standards mean? (Unpack/ restate in your own words.)	Key Vocabulary (Students will KNOW / understand...)	What does it look like in class? (Students will be able to DO...)	Opportunities to learn (Lessons, resources, etc.)
ALGORITHMIC THINKING				
DLCS 1. List the sequence of events required to solve problems. Examples: tying shoes, making a sandwich, brushing teeth.	I can identify the order of events related to a specific task. I can identify what comes next or if a step is out of order. I can tell the order of events for a specific task. I can identify what comes next for specific tasks. I can identify a step that is not in the correct order.	**Algorithm:** A precise sequence of instructions for processes that can be executed by a computer **Bug:** Part of a program that does not work correctly **Debugging:** Finding and fixing problems in an algorithm or program **Sequence:** To arrange in a particular order	ELA - Write informational or explanatory text, such as how-to articles. - Create/draft outlines for writings or projects. - Express a routine as a sequence of step-by-step instructions. - Map or outline a story. - Create decision trees. Math - List the steps to solving math problems. -Determine when a task is not in the correct order. -Order a sequence of events. Science/SS - List steps for a process. - Create if/then statements for concepts. -Order a sequence of events related to an experiment.	https://www.kodable.com/ Students use basic coding skills to follow a sequence. Beebot Challenge Cards Lesson where students follow directions to get the Beebot from point A to point B. Nearpod Lesson: Room on the Broom Story sequencing Seesaw: Sequence the Story Activity for story sequencing that can easily be assigned to students Debugging: Unspotted Bugs A lesson to help students understand the step involved in debugging.

Table 11. One page of Talladega's competency map, connecting computational thinking practices to relevant standards and classroom practice.

Designing Inclusive Learning Opportunities. Talladega's third stage was most ambitious: addressing the nuts and bolts of school- and classroom-level change. Talladega teacher leaders worked in as a committee to develop inclusive assessment tools for their pathway. They decided to focus on three competencies for this work: Algorithmic Thinking, Data Collection and Analysis, and Creating Models and Simulations. "Look for" documents and rubrics were developed for each of the three "focus" competencies. The Inclusive CT Pathways Committee worked to ensure that these documents integrated inclusive pedagogies by validating unconventional ways of doing and knowing computing such as providing learners the flexibility to express their interests and experiences.

Throughout all three phases of the process, district-level leaders worked closely with building administrators and teacher leaders to develop the pathway together through **shared leadership**. Within Talladega County Schools, this collaboration and building a pathway unique to their district using resources identified by teachers has been credited by the district leaders as essential for the development of a strong pathway and the success that they have had implementing it thus far.

Figure 9. District leader and teacher designing K–12 opportunities for CT integration in Talladega county

One of the things we've really tried to focus on with these documents is that they're living documents… they're not just one set and it's just going to stay like that forever. We think internally like that here in Talladega. There's always been new developments.

– Elementary teacher from Talladega County Schools

Tip	Resource
Develop a competency map for your district	CT Pathways Toolkit Talladega's competency map
Conduct empathy interviews to learn more about students' experience with marginalization	Empathy interview protocol

Table 12. Resources to learn more about how Talladega County Schools designed an inclusive K–12 computing pathway

Example: Developing School Capacity Using the CT Integration Framework

The computational thinking integration framework and self-assessment tool guide schools as they integrate computational thinking across subject areas. The framework offers opportunities for shared leadership and explicit considerations of inclusive computing pedagogies as CT is integrated schoolwide.

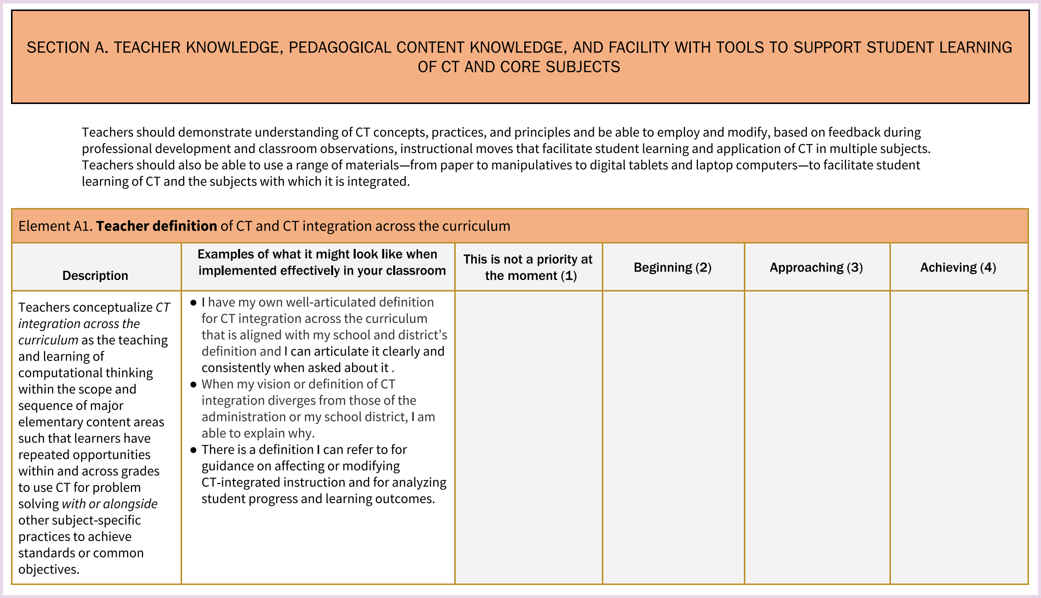
Think about:

- **How does each of the six focus areas play a role in developing CT integrations?**
- **Which of the six focus areas does your school need to prioritize?**
- **How could your school use the self-assessment tool to learn where you are and set goals for future work?**

A collaborative team from the Education Development Center, University of Florida and The Research Alliance for New York City Schools has worked with multiple stakeholders to develop school-wide implementation plans for computational thinking integration within elementary curricula. That is, learning opportunities integrated with computational thinking are offered across subject areas and extend across multiple grades, not just in standalone technology or computer science courses. Through their framework, the development and on-going offering of such learning opportunities requires **shared leadership**. School-wide computational thinking integration involves ongoing coordination, collaboration, and sustained efforts of many stakeholders, including the leadership team, in-school coaches, teachers, and families.

To support school-wide implementation of computational thinking that considers stakeholders across the school building, the team developed the computational thinking integration framework and self-assessment. This tool is designed to assist school leaders and teachers in evaluating their readiness for and guiding the creation and implementation of a plan for integrating CT schoolwide. There are six categories in the framework, each with their own individual elements, that can influence schoolwide CT integration. The six focus areas for are:

- Teacher knowledge, technological pedagogical content knowledge, and facility with tools to support student learning of CT and core subjects
- Teacher supports
- Curriculum features and lesson/unit planning
- CT assessment
- Student impact
- Families



SECTION A. TEACHER KNOWLEDGE, PEDAGOGICAL CONTENT KNOWLEDGE, AND FACILITY WITH TOOLS TO SUPPORT STUDENT LEARNING OF CT AND CORE SUBJECTS

Teachers should demonstrate understanding of CT concepts, practices, and principles and be able to employ and modify, based on feedback during professional development and classroom observations, instructional moves that facilitate student learning and application of CT in multiple subjects. Teachers should also be able to use a range of materials—from paper to manipulatives to digital tablets and laptop computers—to facilitate student learning of CT and the subjects with which it is integrated.

Element A1. **Teacher definition** of CT and CT integration across the curriculum

Description	Examples of what it might look like when implemented effectively in your classroom	This is not a priority at the moment (1)	Beginning (2)	Approaching (3)	Achieving (4)
Teachers conceptualize *CT integration across the curriculum* as the teaching and learning of computational thinking within the scope and sequence of major elementary content areas such that learners have repeated opportunities within and across grades to use CT for problem solving *with or alongside* other subject-specific practices to achieve standards or common objectives.	• I have my own well-articulated definition for CT integration across the curriculum that is aligned with my school and district's definition and I can articulate it clearly and consistently when asked about it . • When my vision or definition of CT integration diverges from those of the administration or my school district, I am able to explain why. • There is a definition I can refer to for guidance on affecting or modifying CT-integrated instruction and for analyzing student progress and learning outcomes.				

Figure 10. An example of an element from the Computational Thinking Integration Framework.

The framework is intended as a starting point for a school or a group of teachers to think about how to begin to integrate computational thinking. The framework should be used initially as a self assessment tool and later to set goals and priorities for CT integration in the school. After setting those goals, the school should create a short-term, mid-term, and long-term action plan to address each of them and regularly evaluate the progress towards meeting each goal. Through this process, the school should revisit its priorities and set new goals periodically.

Woven throughout the self-assessment tool and framework are elements that promote inclusive computing pedagogies such as a focus on meeting the needs of academically and culturally diverse learners, supporting collaboration between students, the importance of differentiation in lesson plans, and connecting with families.

Teams of educators have used the self-assessment tool and associated framework to help identify and think strategically about the factors that are likely to have an impact on their CT integration efforts and successfully designed many computational thinking learning activities for their students as a result.

The framework help[s] teachers and administrators think about how do we implement this across our school, what are the components that we need to look at to really integrate computational thinking throughout all of the content areas or with all of our grade levels…it's a way of identifying our strengths, our weaknesses. Are we ready to implement this and where are those places that we can build on, where are the places that we really need some work? To me the framework is a really good starting point for a school or a group of teachers that are looking to think about how to begin with computational thinking.

– Kristen Beck, Computer Science Specialist/Instructional Coach, Chicago Public Schools

Tip	Resource
Develop school-wide implementation plans for computational thinking integration	Computational Thinking Integration Framework

Table 13. Resources to learn more about how educators designed school-wide CT Implementation programs in their school.

Strategy #2: Develop Sustained, Individualized Professional Learning Opportunities

When teachers have access to robust professional learning opportunities, it can increase their confidence and efficacy implementing computing in their classrooms (P. Rich et al., 2017). There is a critical need to develop systems that support and recognize educators for the knowledge of how to integrate computational thinking. Despite promising approaches to integrating computational thinking into core content areas, challenges persist. For instance, teachers have indicated that challenges to integrating computational thinking into their practices include limited planning and instructional time and difficulty identifying connections between computing concepts and core content (Israel, 2015; Ketelhut et al., 2020; Rich, Yadav & Schwartz, 2019).

A lack of teacher credentialing pathways and professional development leave teachers ill-equipped to integrate CT, particularly in innovative ways that engage students who experience marginalization from computing. Most educators do not have strong computer science backgrounds. Historically, teacher education programs have not included a computing methods course or opportunities to engage in computational thinking practices, which remains true for most programs (Blikstein, 2018; Koshy et al., 2021; Yadav, Stephenson, & Hong, 2017). Policymakers increasingly describe the need to require or expect teachers to integrate CT into core subjects (e.g., Code.org et al., 2020; NGSS Lead States, 2013). To do this, teachers need support to recognize and realize opportunities that integrate CT in ways that enhance disciplinary learning. Not only that, professional learning opportunities need to support teachers to do so using inclusive computing pedagogies. As schools and districts continue to face the challenges caused by the COVID-19 pandemic and the need for online learning, teachers also need support and learning opportunities focused on using computational thinking tools and teaching computational thinking skills and practices in online environments.

District leaders, curriculum specialists, and practitioner experts have partnered with Digital Promise to facilitate professional learning experiences for educators across the nation. One hundred and twenty collaborating teachers completed a survey to provide feedback about how to best support them to seamlessly integrate computational thinking into their instruction. Teachers were asked the open-ended question, "What additional resources would allow you to more seamlessly integrate computational thinking into your instruction?" The responses indicated that they would like continued support with professional learning, clarification of grade-appropriate terminology, access to exemplar lesson plans, strategies to assess student work and recognition for computational competencies. A description and examples of each of these supports is described below in Table 14.

Teacher Support	Description	Examples
Opportunities for ongoing Professional Learning	Most teachers requested more planning time, indicating that teachers may require additional support beyond “sit and get” professional learning to make connections between computational thinking and classroom practice.	• Time to identify integration opportunities in existing lessons • Collaborative planning time • A coach for just-in-time support
Clarification of grade appropriate terminology	Teachers sought additional information about identifying and implementing computational thinking practices, particularly operationalizing grade appropriate terminology.	• Everyday language to describe computing terminology • Examples of computational thinking terminology applied in different lessons
Access to exemplar lesson plans	Teachers expressed a need to access exemplar lesson plans and units illustrating how to integrate computational thinking into their grade and subject area, especially those that address inclusive pedagogies.	• Curated lesson repository for inclusive, disciplinary integrated CT
Strategies to assess student work	Without more effective and consistent assessment measures that are tied to articulated and grade-appropriate standards, teachers are challenged to determine the effectiveness of CT interventions in their schools and communities.	• Looks fors/evidence statements of students engaging in CT practices • Rubrics • Formative assessments
Recognition of computational competencies	Teachers want recognition for what they know and can do. This requires developing teacher-facing measures to assess a teachers’ knowledge of computing (NASEM, 2021). To a greater extent, we lack methods to recognize and provide feedback on teachers integrating novel approaches of computational thinking in their practice.	• Micro-credentials can be used to recognize teacher competency in computing. A micro-credential is a digital certificate or badge that verifies an individual’s competence in a specific skill or set of skills. To earn a micro-credential, teachers submit evidence of their practice including artifacts such as student work, lesson plans, video of classroom implementation, and reflections.

Table 14. Support requested from teachers after participating in traditional professional development.

Example: Differentiated, Readiness-Aligned Professional Learning for Elementary Educators

Project Tomorrow provides individualized professional development to teachers in New York City. Based on an initial assessment of teacher readiness, teachers receive individual coaching and participate in a professional learning community as they learn to integrate computational thinking into their classrooms.

Think about:

- **What do teachers in your district already know about computational thinking integration? What do they need to learn?**
- **What is your professional learning community of teachers who are integrating computational thinking? If you don't have one, is there a way to create one for yourself within your district or state?**

Since January 2020, Project Tomorrow has been implementing computational thinking professional learning with 10 public elementary schools in New York City.

Each partner school has more than 80% enrollment of students experiencing exclusion in the United States (Asian, Hispanic/Latinx, Black or Multi-racial).

The cohort currently consists of 120 teachers in grades 3–5. Overall, the project seeks to build teacher capacity to give students exposure to computational thinking and equip them with skills to be critical thinkers and confident problem solvers.

Most teachers had not participated in any prior CT or CS professional learning. Project Tomorrow designed and implemented a model for professional learning based on individual teacher needs. Key elements of this model included:

- **Assessment of individual teacher readiness.** Each teacher completes a survey to indicate their competency, confidence, and comfort with CT integration.
- **Individual coaching.** An individual professional learning plan is created based on the results of the survey. A coach meets with each teacher twice a month to review lesson plans, provide feedback, and identify additional resources for ongoing learning. CT coaches supported teachers to make connections between CT and the curriculum in core content areas and supported teachers to use technology tools that enable CT learning experiences.
- **Professional learning community.** Teachers from different schools are connected through an online professional learning community, which includes an online repository for resources and annual meetups for teachers to learn from each other through Edcamps.

An individualized learning plan for each teacher allows the coach to understand individual contexts specific to each classroom. This is particularly important for the unique assets and challenges that are relevant to students experiencing marginalization. After integrating computational thinking in their classrooms, teachers have expressed that the CT problem solving approach is flexible and universal, and can be used to help students that are at different levels, enhancing differentiated instruction. The individualized approach to learning is something that Project Tomorrow encourages teachers to take with their own students.

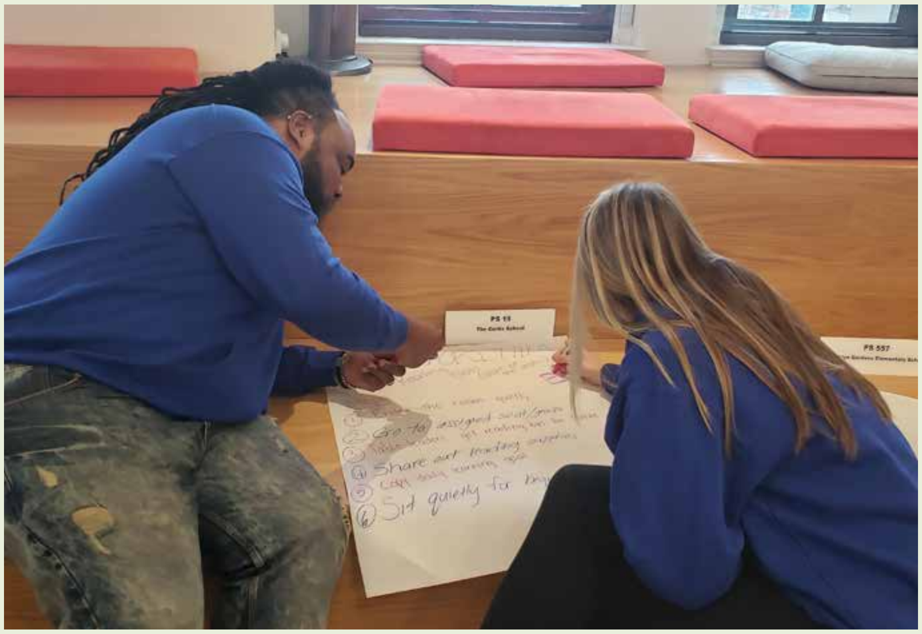


Figure 11. Teachers creating a poster renaming an established classroom routine to transition to reading class as an algorithm

Preliminary results suggest that this may be a sustainable, replicable model for teacher professional learning on CT integration. Teachers' understanding of CT concepts increased after participating in the professional learning during the first six months of the project (Evans, 2021). During the first academic semester following the PD, 16% of new teachers to the project identified having a good understanding of CT concepts and practices. This improved to 94% of teachers by the end of that academic year. This improved understanding of CT enabled 71% of teachers to use CT concepts to supplement a specific unit or lesson and 23% of teachers to integrate CT practices into disciplinary learning.

In order to scale and sustain the model of professional learning beyond the initial cohort, Project Tomorrow emphasized the need to leverage support from administrators and parents, in addition to identifying a CT teacher-leader within the school. Plans are in place to replicate this model in 20 schools in New York City during the 2022–23 school year and to identify new project locations outside of New York City interested in implementing a localized version of this model.

Tip	Resource
Develop teachers' computational thinking schools through coaching	CS Coaching Toolkit by Cornell Tech

Example: Professional Learning to Integrate Computational Thinking into Elementary Classrooms

Maker Partnership Program utilized maker learning approaches in order to build teachers' capacity to integrate computational thinking to their science instruction.

Think about:

- **What frameworks or initiatives does your district use that could support computational thinking integration?**
- **How can you integrate the characteristics of the professional development that were successful into your classroom teaching or professional development at your school?**

The Maker Partnership Program (MPP) is a research practice partnership between the Research Alliance for New York City Schools, MakerState, and Schools That Can. MPP developed and tested a new model for building teachers' capacity to integrate computational thinking (CT) into regular science instruction using Maker pedagogy (Fancsali et al., 2019; 2021). The project explored facilitators and barriers to CT integration in elementary schools, as well as outcomes for MPP teachers and students.

The Maker approach is based on the engineering design cycle—which includes defining a problem; researching, planning, prototyping, and testing solutions; and then refining those solutions.

The hands-on, interdisciplinary nature of Maker activities makes them promising for integrating CT into learning and for designing instruction that is accessible, collaborative, and empowering to students. MPP teachers often paired students to brainstorm together, test each others' prototypes, and provide peer feedback that was then used to improve designs. In MPP's "circle meetings," students presented and explained their work to their peers and shared insights about what they learned—making their thinking visible, normalizing struggle and mistakes, and facilitating processing and deeper understanding of the ideas or concepts being taught. Student rubrics clarified expectations and provided feedback on students' mastery of skills.

Research showed that the Maker framework helped teachers embrace a pedagogical shift—from instructors transferring knowledge to facilitators of learning. Teachers found Maker pedagogy to be an effective approach for integrating CT because it allows multiple entry points for students with a wide range of CT skill levels. They also observed that MPP activities were engaging for students, encouraged and provided various opportunities for peer collaboration and feedback, and improved problem-solving skills, suggesting promising outcomes for students, particularly those who have been marginalized from computing such as Black, Latinx, and female students.

MPP yielded important insights about building teachers' capacity for CT integration and the use of Maker pedagogy. Teachers found a number of characteristics of MPP's professional development (PD) to be particularly effective, including:

- **Sustained engagement in PD** (e.g., over the course of two years), with sessions spread throughout the year. This allowed teachers to learn new concepts, approaches, and skills in a session, try them out with their students, and then come together again to review their successes and challenges and to get feedback and support. It also allowed teachers to provide formative feedback on the PD, which helped improve future sessions and bolster the support being provided by the program.
- **Hands-on PD activities** (e.g., teachers learned CT concepts and skills by creating their own Scratch programs). The PD modeled lessons and pedagogical approaches, with teachers experiencing the lessons as students would. PD sessions also offered guided practice in using new CT skills, and time for planning and collaboration with other teachers and the MPP coaches.
- **CT resources and materials** (e.g., lesson plans and units, model projects and skill-building videos for students, student assessment rubrics). These materials were key to helping teachers integrate CT into their science instruction and saved them time and effort.
- **Structured planning time.** Even with access to high-quality resources, teachers needed to modify or adapt most lessons to differentiate supports and align activities with the specific science topics they planned to teach. Providing time during MPP PD sessions—as well as the guidance and support of MPP coaches and teacher colleagues—were seen as especially valuable to teachers as they prepared integrated lessons.

Figure 12. Teachers collaborating on a hands-on STEM/CT maker learning activity during a professional development session

Teachers also reported challenges to integrating CT into their science instruction:

- **Time constraints and scheduling.** Teachers reported that a lack of time to collaborate with each other and plan lessons was a barrier to integration, and they sometimes struggled to find space in their class or lesson schedule for incorporating CT.
- **Differentiated Instruction.** Teachers found that students were diverse in terms of prior CT experience (for example, in Scratch coding), with some requiring foundational instruction in order to implement integrated CT science lessons, while others benefitted from extension activities.
- **Logistics.** Issues such as hardware and internet access, setting up Scratch accounts, and getting students logged in, had to be addressed for successful integration of CT into science. Teachers benefitted from specific support and assistance troubleshooting and establishing routines and procedures to mitigate these challenges.

Engaging school leaders in MPP was also important, because they are in a position to support teachers in overcoming common barriers to integration. For example, school leaders could facilitate class scheduling to allow for integration of CT, carve out planning time so teachers could collaborate with others, allow teachers release time so that they can attend PD, and ensure that teachers have access to adequate internet, hardware, software, and other needed materials.

Tip	Resource
Integrate Maker learning strategies to facilitate inclusive pedagogies	Descriptions of Maker strategies Maker Learning Leadership framework
Design hands-on learning experiences and structured time for work for teachers	Example professional development agendas STIGCT: Science Teaching Computational Thinking Inquiry Group
Provide teachers with resources and materials	Example CT integrated lessons projects that integrate science and CS/CT
Example rubrics for CT assignments	Assessments

Table 15. Teaching tips and resources to learn more about helping teachers to integrate computational thinking into elementary classrooms

Strategy #3: Integrate Computational Thinking into Pre-Service Teacher Education

While professional development for in-service teachers serves a critical role in creating capacity to integrate CT into schools, the sustainability and longevity of this pedagogical innovation depends on our ability to make CT a part of every new teacher's toolbox. Embedding computing instruction within pre-service education programs would help create cohorts of teachers who arrive at the classroom ready to introduce students to computing and leverage CT for disciplinary learning. However, few teacher education programs are prepared to provide their students with opportunities to learn about computing and consider how CT skills and practices are relevant to their classroom practice. Today, not many programs preparing elementary and middle school teachers cover computer science or CT as a set of skills that can be integrated into other areas of instruction (Vegas & Fowler, 2020). Teacher educators themselves need support in order to design courses instructing pre-service teachers to integrate computational thinking into their future classrooms. Moreover, universities are struggling to prepare and credential teachers to teach computer science (CSTA, 2013). To begin countering these deficiencies, Yadav et al. (2017) suggested that successful initiatives that integrate computational thinking into teacher education should:

- Build **partnerships between computer science educators and teacher educators** to develop curricula integrated with computational thinking skills and practices.
- Introduce computational thinking skills in **educational technology courses** (e.g. Chang & Peterson, 2018; Mouza et al., 2017; Yadav et al., 2014).
- Apply computational thinking practices to various subject areas through **methods coursework** (e.g. Jaipal-Jamani & Angeli, 2017; Ketelhut, 2020; Patel et al., 2021).

Institutions, like the City University of New York, that have successfully integrated computational thinking into pre-service teacher education emphasize the following implications for designing CT learning experiences in teacher education:

- **Resources and tools to define computational thinking**—such as frameworks, look-fors and rubrics—support teachers and faculty members in identifying opportunities to integrate computational thinking into their classroom and/or reflect on their practices (Patel et al., 2021). Cabrera et al. (under review) designed a teacher-oriented framework for computational thinking integration into elementary science at the elementary level. This resource lowered the language barrier of CT frameworks with technical computing terminology and focused on the CT practices that are developmentally appropriate for elementary school students.
- Provide opportunities for pre-service teachers to **apply computational thinking skills to disciplinary embedded problem solving**, as opposed to discrete computer science learning opportunities, such as coding in isolation. Patel et al., (2021) found this pedagogical approach promoted pre-service teachers' ability to apply computational thinking into their teaching contexts. Mills et al. (2020) leveraged this pedagogical approach in a science methods course, with learning activities such as using computational models to explore ecosystem dynamics (see Waterman et al., 2020) and designing algorithms to identify animals (Coenraad et al., 2020).
- **Provide coaching for the application of CT practices into classroom practice.** Teacher educators have noted that faculty members and pre-service teachers alike benefitted from ongoing coaching about the application of computational thinking practices into their coursework or classrooms. Patel et al. (2021) provided faculty of methods courses with regular coaching sessions with a computational thinking expert to provide feedback and guidance about course activities and assignments. Mills et al. (2020) met with pre-service teachers individually to provide feedback about their computational thinking integrated lesson plans, a capstone assignment of the course.
- **Engage stakeholders across higher education and local school districts.** Patel et al. (2021) emphasized the need to include clinical faculty in the design team to ensure that CT is integrated into the student teaching experience, such as in lesson templates and observation rubrics. Additionally, they recommend partnering with schools and teachers that have CT experience to provide strong CT mentorship during the student teaching experience. Ketelhut et al. (2020) & Killen et al. (2020) developed an innovative model for professional learning with mentor teachers and pre-service teachers learning computational thinking together sustained over the duration of the student teaching experience.

With the demand for teachers who are proficient in computational thinking, having such a focus in pre-service education can make graduating teachers more employable. The City University of New York (CUNY) has been working to develop a pipeline of teachers knowledgeable in computational thinking. A few of these programs are described in the examples below.

Example: Computational Thinking and Universal Design for Learning in Introductory Education Coursework

Hostos Community College integrated Universal Design for Learning and computational thinking into an instructional technology course. The course models the inclusive pedagogies taught while providing project-based learning opportunities for pre-service teachers. To expand computational thinking in the teacher education course, the faculty members are currently receiving 1:1 coaching to integrate it into their materials.

Think about:

- **What are the synergies between Universal Design for Learning and computational thinking that can be leveraged for integration and to promote student learning?**
- **What do teaching faculty at colleges and universities need to know in order to support pre-service teachers as they learn to integrate computational thinking in their classrooms?**

Hostos Community College in New York City has integrated Universal Design for Learning alongside computational thinking into an instructional technology course to meet the new state Computer Science and Digital Fluency learning standards. EDU226, Introduction to Instructional Technology, is an intermediate-level education course, which can be taken as an elective for students in Early Childhood Education or Math Education degree concentrations. It is also an option for teachers and paraprofessionals, and for students participating in programs offered by Continuing Education & Workforce Development.

This revised course contains four modules:

- Module 1: Digital Literacy and Citizenship
- Module 2: Unplugged Computational Thinking & Instructional Design
- Module 3: Accessibility and Universal Design for Learning
- Module 4: Applications of Computational Thinking & Digital Ethics

The instruction of the modules was designed using project-based and flipped learning approaches and applied frameworks such as Community of Inquiry, Universal Design for Learning, 5E Model of Instruction and the Hostos Course Development Guidelines. Within each module, students participate in learning opportunities that model the inclusive pedagogies we encourage pre-service teachers to integrate into their own practice.

Each module was designed to be taught independently; therefore, they can be integrated in any of the required EDU courses, thereby giving students more opportunities to learn about inclusivity and computational thinking. To scale and sustain the integration of computational thinking into the teacher education program, eleven faculty members are currently engaging in 1:1 coaching sessions so they are able to produce computational thinking materials in their courses.

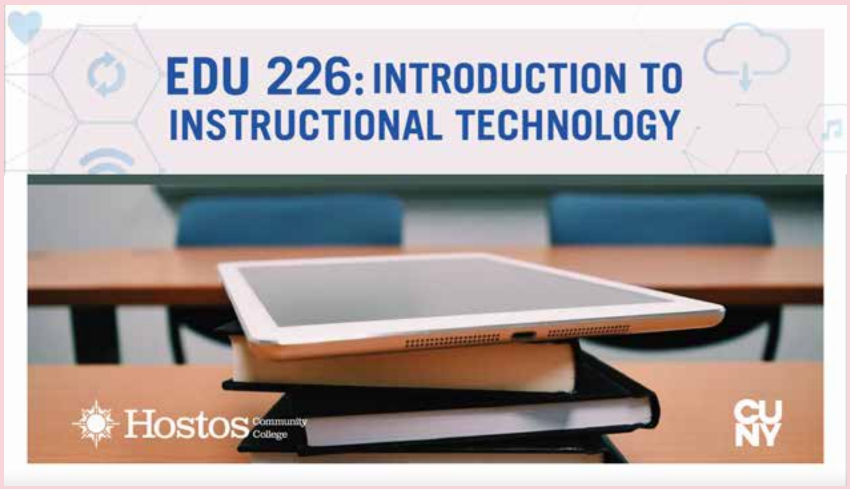


Figure 13. Banner for EDU226: Introduction to Instructional Technology, a course for pre-service teachers that integrates computational thinking and Universal Design for Learning

The students enrolled at Hostos Community College are about 67% female and 66% Hispanic. The pre-service teachers may have not had many opportunities to develop the skills and confidence to integrate CT into classroom practice. Therefore, the redesign of the course intentionally centered the marginalized identities of both the pre-service teachers and the students they are working with. For instance, students examined an exemplary lesson developed by PiLa CS that leverages computational thinking and translanguaging practices, and created their own version of the project for their own classrooms.

Example: Integrating Computational Thinking into Methods Courses and Student Teaching Experiences

Queens College integrated computational thinking into both the course work and the field experiences of pre-service teachers in their teacher education program.

Think about:

- **When you work with pre-service teachers, how could you support them to practice integrating computational thinking?**
- **How can pre-service teachers work with their mentor teachers to integrate computational thinking?**

In collaboration with International Society for Technology in Education (ISTE) and partnering school districts, Queens College, part of CUNY, integrated computational thinking into the coursework and clinical experiences of pre-service teachers. To integrate CT into methods courses, faculty participated in summer professional development and identified opportunities for CT integration in each teacher education course. The methods course instructors received additional ongoing coaching to recognize and assess CT integration in practice and provide feedback to support pre-service teachers to develop robust CT learning opportunities.

To meaningfully integrate CT into pre-service teachers' student teaching experience, CUNY integrated it into field experiences for pre-service teachers — and thus they changed how they place pre-service teachers and how they support pre-service teachers during student teaching. The college partnered with a local school district. The superintendent selected three elementary schools as placement sites for CT integration and collaborated with principals and other building leaders to move the initiative forward. Additional clinical faculty provided additional support by adapting these lesson templates and observation rubrics to include CT integration. Cooperating mentor teachers participated in one day of professional development prior to the beginning of the students' internship experience, with the option of extending professional learning during the semester. During their student teaching, pre-service teachers designed and implemented 2–4 CT integrated lessons for grades 1–5. Additionally, they had the opportunity to earn three ISTE micro-credentials designed to acknowledge competency in integrating computational thinking into elementary instruction.

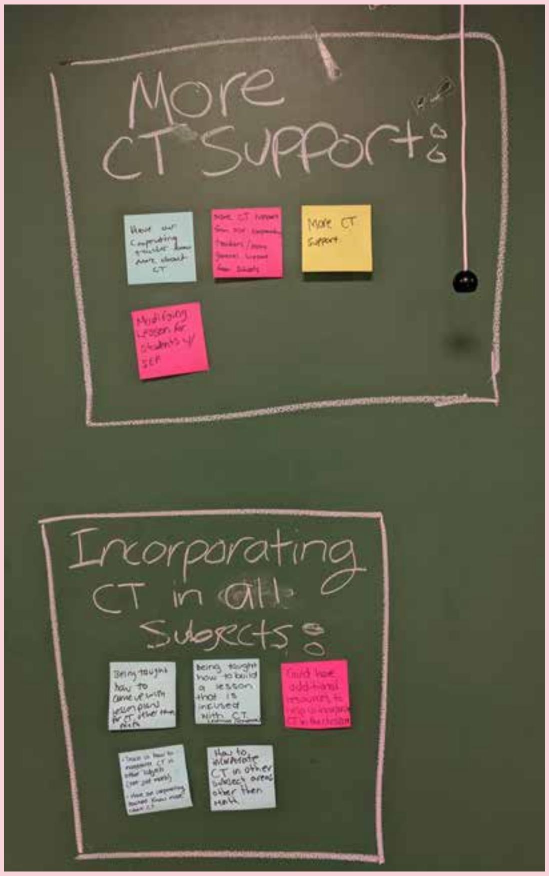

More CT Support:
More CT Support
Modifying Lesson for students w/ IEP
Incorporating CT in all Subjects:
Being taught how to come up with lesson plans for CT other than Math
Being taught how to build a lesson that is included with CT
How to incorporate CT in other subject areas other than Math

Figure 14. Pre-service teachers reflect on their computational thinking learning experiences with desired support for learning about computational thinking

CUNY began this work in order to advance equitable access to computing education in New York City. CT integration was piloted in a generalist childhood education program in order to reach pre-service teachers who will be able to shape their students' computing abilities and their computing identities. With the most diverse teacher education graduates in New York State, CUNY pre-service teachers, many of whom come from New York City public schools and share racial, national origin, or linguistic backgrounds with New York City public school students. The pre-service teachers in the project were representative of the CUNY teacher education population and the student population in the clinical placement partner schools were representative of the citywide student population. Both populations are primarily made up of groups who have experienced marginalization in computing, math, and science. Pre-service teachers' social and emotional relationship to computing is critical to their future students' relationship to computing. The project engaged this diverse set of pre-service teachers and gave them the tools to integrate CT in science, math and literacy lessons using inquiry-driven pedagogy; however, upon reflection, CUNY realized the equity imperative driving the project was left in the background of the course integration.

Future iterations will place equity in the foreground in faculty training, the integration design process, and clinical placements. For example, in summer 2021 a cross-university collaboration provided faculty training in partnership with Michigan State University. The training included discussions on how scholars and tech designers are leveraging ideas from other disciplines to raise issues about technology's impact on society and a session about the translanguaging potential of computationally rich activities by a researcher from the Participating in Literacies and Computer Science project.

Going forward, CUNY plans to scale integration of computing to all 15 colleges with education programs. They will continue to explore how the integration design process could ask faculty to identify integration outcomes that include pre-service teachers making specific cultural connections between computing and their learners, and clinical seminars could include support for the pre-service teachers to learn to lead critical conversations about racial disparities through the lens of the impact of technology on society. Increasing access for students begins with well-trained teachers who themselves know about computational thinking and can identify synergistic integration points, but going beyond access requires preparing teachers to have difficult conversations around computing and society and to identify connections between the lives of the students they are teaching and computational thinking.

Conclusion

Computational skills are essential for everyone to participate in our increasingly computational world. However, long-standing inequities in the representation, participation and achievement of people that identify as Black, Native American, and Latinx; students with disabilities; girls; non-binary students; and students experiencing poverty persist in computing today, as consequences of previously existing social structures and inequities, and exacerbated by our response to the pandemic. In this report, we issued two calls for action for educators to design inclusive computing learning opportunities for students: (1) integrate computational thinking into disciplinary learning, and (2) build capacity for computational thinking with shared leadership and professional learning. Inspired by the frameworks, strategies, and examples of inclusive computational thinking integration throughout the report, readers can take away practical implications to reach learners in their contexts. Now is the time to integrate computational thinking with inclusive pedagogies into every classroom. 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.

References

Allen-Handy, A., Ifill, V., Schaar, R., Rogers, M., & Woodard, M. (2020). Black girls STEAMing through dance: Inspiring STEAM literacies, STEAM identities, and positive self-concept. In K. Thomas & D. Huffman (Eds.), *Challenges and opportunities for transforming from STEM to STEAM education* (pp. 198–219). IGI Global. https://doi.org/10.4018/978-1-7998-2517-3.ch008

Angeli, C., Voogt, J., Fluck, A., Webb, M., Cox, M., Malyn-Smith, J., & Zagami, J. (2016). *A K–6 computational thinking curriculum framework: Implications for teacher knowledge. Journal of Educational Technology & Society, 19*(3). https://www.researchgate.net/publication/305140678_A_K-6_Computational_Thinking_Curriculum_Framework_Implications_for_Teacher_Knowledge

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

Banilower, E. R., Smith, P. S., Malzahn, K. A., Plumley, C. L., Gordon, E. M., & Hayes, M. L. (2018). *Report of the 2018 NSSME+*. Horizon Research, Inc. https://files.eric.ed.gov/fulltext/ED598121.pdf

Basu, S., Biswas, G., Sengupta, P., Dickes, A., Kinnebrew, J. S., & Clark, D. (2016). Identifying middle school students' challenges in computational thinking-based science learning. *Research and Practice in Technology Enhanced Learning, 11*(1), 13. https://doi.org/10.1186/s41039-016-0036-2

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

Bureau of Labor Statistics. (2018). Labor force statistics from the current population survey: Table 11. https://www.bls.gov/cps/#tables

Bureau of Labor Statistics. (2019). Persons with a disability: Labor force characteristics—2018. https://www.bls.gov/news.release/archives/disabl_02262019.pdf

Blikstein, P. (2018). *Pre-college computer science education: A survey of the field.* Google LLC. https://goo.gl/gmS1Vm

Brennan, K., & Resnick, M. (2012, April). New frameworks for studying and assessing the development of computational thinking. In *Proceedings of the 2012 Annual Meeting of the American educational research association,* Vancouver, Canada (Vol. 1, p. 25). https://web.media.mit.edu/~kbrennan/files/Brennan_Resnick_AERA2012_CT.pdf

Brennan, K. (in press). A case for why: Society, school, self. In S.-C. Kong and H. Abelson (Eds.), *Computational thinking education in K–12.* MIT Press.

Brigham Young University (2021, September 10). Tech With kids: Computational thinking and coding badges. https://iptedtec.wixsite.com/techwithkids/badges

Burke, Q., Roschelle, J., Angevine, C., O'Donnell, K. A., Smith, K., & Weisgrau, J. (2019, March). Developing inclusive K–12 computing pathways for the League of Innovative Schools. [Poster presentation] *2019 Research on Equity and Sustained Participation in Engineering, Computing, and Technology (RESPECT)* Conference, March 27th, Minneapolis, MN. http://respect2019.stcbp.org/conference-program/

Cabrera, L., Ketelhut, D. J., Mills, K., Coenraad, M., Killen, H., Byrne, V., & Plane, J. (Under Review). Designing a framework for teachers' integration of computational thinking into elementary science.

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

Chang, Y. H., & Peterson, L. (2018). Pre-service teachers' perceptions of computational thinking. *Journal of Technology and Teacher Education, 26*(3), 353–374. https://www.learntechlib.org/primary/p/181433/

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

CodeVA. (in press) *Integration planning guide.*

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.

Coenraad, M., Plane, J., Ketelhut, D.J., Cabrera, L., Mills, K., & Killen, H. (2020). STIGCT: The Science Teacher Computational Thinking Inquiry Group [White paper]. https://education.umd.edu/stigct

Common Core State Standards Initiative (2021). *Preparing America's Students for Success.* http://www.corestandards.org/

Computer Science Teachers Association (CSTA) (2013). *Bugs in the system: Computer science teacher certification in the U.S* [White paper]. https://www.csteachers.org/documents/en-us/3b4a70cd-2a9b-478b-95cd-376530c3e976/1

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

DiSessa, A. A. (2018). Computational literacy and "the big picture" concerning computers in mathematics education. *Mathematical thinking and learning, 20*(1), 3–31. https://doi.org/10.1080/10986065.2018.1403544

Duncan, C., & Bell, T. (2015). A pilot computer science and programming course for primary school students. 39–48. ACM. https://doi.org/10.1145/2818314.2818328

Erickson, T., Wilkerson, M., Finzer, W., & Reichsman, F. (2019). Data moves. *Technology Innovations in Statistics Education, 12*(1). https://escholarship.org/uc/item/0mg8m7g6

Everyday Computing. (2021). *Action Fractions.* http://everydaycomputing.org/lessons/action-fractions/

Evans, J. (2021). Personalizing elementary teacher professional learning on computational thinking integration based upon understanding teacher readiness. Presented at the American Education Research Association, virtual, April 8–12, 2021.

Fancsali, C., Klevan, S., & Mirakhur, Z. (2021). Making research practice partnerships work: An assessment of The Maker Partnership. Paper presented at Research in Equity and Sustained Participation in Engineering, Computing, and Technology (RESPECT). May 23–27, 2021. Virtual.

Fancsali, C., Mirakhur, Z., Klevan, S., & Rivera-Cash, E. (2019). "Making" science relevant for the 21st century: Early lessons from a research-practice-partnership. In *Proceedings of FabLearn 2019* (pp. 136–139). https://doi.org/10.1145/3311890.3311910

Fancsali, C., Mirakhur, Z., Klevan, S., & Rivera-Cash, E. (2021). The Maker Partnership Program: Lessons Learned and Recommendations from a Teacher Capacity-Building Effort. Research Alliance for NYC Schools

Flapan, J., Ryoo, J. J., Hadad, R., & Knudson, J. (2021). Preparing school leaders to advance equity in computer science education. *Journal of Computer Science Integration, 4*(1), 2. http://doi.org/10.26716/jcsi.2021.10.8.33

García, O., & Wei, L.. (2014). *Translanguaging: Language, bilingualism and education.* Palgrave Macmillan Pivot. https://doi.org/10.1080/15235882.2014.965361

Google Inc., & Gallup Inc. (2016). Diversity gaps in computer science: Exploring the underrepresentation of girls, Blacks and Hispanics. https://services.google.com/fh/files/misc/diversity-gaps-in-computer-science-report.pdf

Gutiérrez, K. D. (2014). Integrative research review: Syncretic approaches to literacy learning: Leveraging horizontal knowledge and expertise. In P. J. Dunston, L. B. Gambrell, K. Headley, S. K. Fullerton, & P. M. Stecker (Eds.), *63rd Literacy Research Association Yearbook* (pp. 48–60). http://www.academia.edu/28876369/Syncretic_Approaches_to_Literacy_Learning-_Leveraging_Horizontal_Knowledge_and_Expertise.pdf

Herold, B. (2020, April 10). The disparities in remote learning under coronavirus (in charts). *EducationWeek* https://www.edweek.org/technology/the-disparities-in-remote-learning-under-coronavirus-in-charts/2020/04

Hestness, E., Jass Ketelhut, D., McGinnis, J. R. & Plane, J. (2018). Professional knowledge building within an elementary teacher professional development experience on computational thinking in science education. *Journal of Technology and Teacher Education, 26*(3), 411–435. Waynesville, NC USA: Society for Information Technology & Teacher Education. https://www.learntechlib.org/primary/p/181431/.

Hodges, C., Moore, S., Lockee, B., Trust, T., & Bond, A. (2020, March 27). The difference between emergency remote teaching and online learning. *Educause Review.* https://er.educause.edu/articles/2020/3/the-difference-between-emergency-remote-teaching-and-online-learning

Integrated Computational Thinking (2021). https://projects.ctintegration.org/

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

Israel, M., Pearson, J. N., Tapia, T., Wherfel, Q. M., & Reese, G. (2015). Supporting all learners in school-wide computational thinking: A cross-case qualitative analysis. *Computers & Education, 82*, 263–279. https://doi.org/10.1016/j.compedu.2014.11.022

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

James, S., Silfa, K., Vogel, S., Ascenzi-Moreno, L., Hoadley, C., & Ma, J. (2020, Oct). Classroom case study: Modeling the impacts of Hurricane María in Puerto Rico. *Participating in Literacies and Computer Science (PiLaCS)*. https://nyuscholars.nyu.edu/en/publications/classroom-case-study-modeling-the-impacts-of-hurricane-mar%C3%ADa-in-p

Jaipal-Jamani, K., & Angeli, C. (2017). Effect of robotics on elementary preservice teachers' self-efficacy, science learning, and computational thinking. *Journal of Science Education and Technology, 26*(2), 175–192. https://link.springer.com/article/10.1007/s10956-016-9663-z

K–12 computer science framework. (2016). http://www.k12cs.org.

Kafai, Y. B., & Burke, Q. (2014). Connected code: Why children need to learn programming. MIT Press.

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

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. https://link.springer.com/article/10.1007/s10956-019-09798-4

Killen, H., Coenraad, M., Byrne, V., Cabrera, L., & Ketelhut, D. J. (2020) Reimagining computational thinking professional development: Benefits of a community of practice model. In P*roceedings of the International Conference of the Learning Sciences (ICLS 2020)*, Nashville, Tennessee, USA. ISLS. https://par.nsf.gov/servlets/purl/10204993

Koshy, S., Martin, A., Hinton, L., Scott, A., Twarek, B., & Davis, K., (2021). *The computer science teacher landscape: Results from a nationwide survey* [White paper]. The Kapor Center and Computer Science Teachers Association. https://www.kaporcenter.org/the-computer-science-teacher-landscape-results-of-a-nationwide-teacher-survey/

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

LearnPlatform. (2021). National EdTech Equity Dashboard. https://learnplatform.com/equity-dashboard

Lee, C. H., & Soep, E. (2016). None but ourselves can free our minds: Critical computational literacy as a pedagogy of resistance. *Equity and Excellence in Education, 49*(4), 480–492. https://doi.org/10.1080/10665684.2016.1227157

Lee, I., & Malyn-Smith, J. (2020). Computational thinking integration patterns along the framework defining computational thinking from a disciplinary perspective. *Journal of Science Education and Technology, 29*(1), 9–18. https://doi.org/10.1007/s10956-019-09802-x

Lee, I., Malyn-Smith, J., & Peterson, K. (2021). *Framing computational modeling and simulation as a social justice issue.* 2021 Stem For All Video Showcase. https://stemforall2021.videohall.com/presentations/2101?display_media=video

Lee, I., Martin, F., & Apone, K. (2014). Integrating computational thinking across the K–8 curriculum. *ACM Inroads*, *5*(4), 64–71. https://doi.org/10.1145/2684721.2684736

Lee, I., Martin, F., Denner, J., Coulter, B., Allan, W., Erickson, J., Malyn-Smith, J., & Werner, L. (2011). Computational thinking for youth in practice. *ACM Inroads, 2*(1), 32–37. https://doi.org/10.1145/1929887.1929902

Lindahl, A. (2012). Facing cancer: Social justice in biology class. *Rethinking Schools, 26*(4), 14–18. https://rethinkingschools.org/articles/facing-cancer-social-justice-in-biology-class/

Luke, C., & Young, V. (2020). *Integrating micro-credentials into professional learning: Lessons from five districts* [White paper]. Digital Promise Global. https://digitalpromise.org/wp-content/uploads/2020/10/Integrating-Micro-credentials.pdf

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

Magerko, B., Freeman, J., McKlin, T., Reilly, M., Livingston, E., McCoid, S., & Crews-Brown, A. (2016). EarSketch: A STEAM-based approach for underrepresented populations in high school computer science education. *ACM Transactions on Computing Education, 16*(4), 14:1–14:25. https://doi.org/10.1145/2886418

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

Martin, A., Koshy, S., Hinton, L., Scott, A., Twarek, B., & Davis, K. (2020). T*eacher perspectives on COVID-19's impact on K–12 computer science instruction* [White paper]. Kapor Center. https://www.kaporcenter.org/teacher-perspectives-on-covid-19s-impact-on-K-12-computer-science-instruction/

Malyn-Smith, J., Lee, I., Martin, F. G., Grover, S., Evans, M. A., & Pillai, S. (2018). Developing a framework for computational thinking from a disciplinary perspective. In S. C. Kong, D. Andone, G. Biswas, T. Crick, H. U. Hoppe, T. C. Hsu, & J. Vahrenhold (Eds.), *Conference Proceedings of the International Conference on Computational Thinking Education 2018* (pp. 182–186). The Education University of Hong Kong.

Metz, C. (2021, September 9). A.I. can now write its own computer code. That's good news for humans. *The New York Times*. https://www.nytimes.com/2021/09/09/technology/codex-artificial-intelligence-coding.html

Mills, K., Coenraad, M., Cabrera, L., Killen, H., Ketelhut, D., & McGinnis, J. R. (2020, Apr 17–21). A design-based research approach to integrating computational thinking into elementary science teacher education [Roundtable Session]. AERA Annual Meeting San Francisco, CA

Mouza, C., Yang, H., Pan, Y. C., Ozden, S. Y., & Pollock, L. (2017). Resetting educational technology coursework for pre-service teachers: A computational thinking approach to the development of technological pedagogical content knowledge (TPACK). *Australasian Journal of Educational Technology, 33*(3). https://doi.org/10.14742/ajet.3521

National Academies of Sciences, Engineering, and Medicine (NASEM) 2021. *Cultivating interest and competencies in computing: Authentic experiences and design factors.* The National Academies Press. https://doi.org/10.17226/25912

National Center for Women & Information Technology. (2018). *By the numbers.* https://www.ncwit.org/resources/numbers

National Center for Women & Information Technology. (2021a) *Counselors for computing (C4C)*. https://ncwit.org/program/counselors-for-computing/

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/

Nesiba, N., Pontelli, E., & Staley, T. (2015). DISSECT: Exploring the relationship between computational thinking and English literature in K–12 curricula. *2015 IEEE Frontiers in Education Conference (FIE)*, 1–8. http://doi.org/ 10.1109/FIE.2015.7344063

NGSS Lead States. (2013). Next Generation Science Standards: For states, by states (Vol 1). https://www.nap.edu/catalog/18290/next-generation-science-standards-for-states-by-states

Noble, S. U. (2018). *Algorithms of oppression.* New York University Press.

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

Patel, A., Thompson, A., Williams, H., Abell, O., & Sykora, C. (2021). Introducing pre-service teachers to computational thinking at scale [White Paper]. City University of New York (CUNY) & International Society for Technology in Education (ISTE). https://academicworks.cuny.edu/cgi/viewcontent.cgi?article=1022&context=oaa_pubs

Pinkard, N., Erete, S., Martin, C. K., & McKinney de Royston, M. (2017). Digital youth divas: Exploring narrative-driven curriculum to spark middle school girls' interest in computational activities. *Journal of the Learning Sciences, 26*(3), 477–516. https://doi.org/10.1080/10508406.2017.1307199

Proctor, C., & Blikstein, P. (2019). Unfold studio: Supporting critical literacies of text and code. *Information and Learning Science, 120*(5–6), 285–307. https://doi.org/10.1108/ILS-05-2018-0039

Rich, K .M., Yadav, A., & Schwarz, C. V. (2019). Computational thinking, mathematics, and science: Elementary teachers' perspectives on integration. *Journal of Technology and Teacher Education, 27*(2), 165–205. https://www.learntechlib.org/primary/p/207487/.

Rich, K. M., Strickland, C., Binkowski, T. A., & Franklin, D. (2019). A K–8 Debugging Learning Trajectory Derived from Research Literature. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education* (pp. 745–751). https://dl.acm.org/doi/10.1145/3287324.3287396

Rich, K. M., Strickland, C., Binkowski, A., Moran, C., & Franklin, D. (2017). K–8 Learning trajectories derived from research literature: Sequence, repetition, conditionals. In ICER '17: *Proceedings of the 2017 Association for Computing Machinery (ACM) Conference on International Computing Education Research* (pp. 182–190). ACM. https://doi.org/10.1145/3105726.3106166

Rich, K. M., Binkowski, T. A., Strickland, C., & Franklin, D. (2018). Decomposition: A K–8 Computational Thinking Learning Trajectory. In ICER '18: *Proceedings of the 2018 Association for Computing Machinery (ACM) Conference on International Computing Education Research* (pp. 124–132). ACM. https://doi.org/10.1145/3230977.3230979

Rich, P. J., Jones, B. L., Belikov, O., Yoshikawa, E., & Perkins, M. (2017). Computing and engineering in elementary school: The effect of yearlong training on elementary teacher self-efficacy and beliefs about teaching computing and engineering. *International Journal of Computer Science Education in Schools, 1*(1), n1. https://doi.org/10.21585/ijcses.v1i1.6

Ruiz, P., Iwatani, E., Burke, Q., Tackett, T. (2021, April 5). Adding a computational boost to your K–8 Lessons. https://digitalpromise.org/2021/04/06/adding-a-computational-boost-to-your-k-8-lessons/

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

Santo, R., Vogel, S., & Ching, D. (2019). CS for what? Diverse visions of computer science education in practice. CUNY. https://academicworks.cuny.edu/gc_pubs/562/

Schanzer, E., Fisler, K., & Krishnamurthi, S. (2018). Assessing bootstrap: Algebra students on scaffolded and unscaffolded word problems. In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education* (pp. 8–13). https://doi.org/10.1145/3159450.3159498

Seiter, L., & Foreman, B. (2013). *Modeling the learning progressions of computational thinking of primary grade students.* 59–66. ACM. https://doi.org/10.1145/2493394.2493403

Sengupta, P., Kinnebrew, J. S., Basu, S., Biswas, G., & Clark, D. (2013). Integrating computational thinking with K–12 science education using agent-based computation: A theoretical framework. *Education and Information Technologies, 18*(2), 351–380. https://doi.org/10.1007/s10639-012-9240-x

Settle, A., Franke, B., Hansen, R., Spaltro, F., Jurisson, C., Rennert-May, C., & Wildeman, B. (2012). Infusing computational thinking into the middle- and high-school curriculum. *Proceedings of the 17th ACM Annual Conference on Innovation and Technology in Computer Science Education - ITiCSE '12.* https://doi.org/10.1145/2325296.2325306

Swearingen, J. (2018, November). Unable to avoid bias, Gmail stops using gender in its automated replies. *New York Magazine*, 1–4. https://nymag.com/intelligencer/2018/11/unable-to-stop-bias-gmail-avoids-gender-in-ai-replies.html

Terada, Y. (2020, May 4). A powerful model for understanding good tech integration. https://www.edutopia.org/article/powerful-model-understanding-good-tech-integration

Tissenbaum, M., Sheldon, J., Seop, L., Lee, C. H., & Lao, N. (2017). *Critical computational empowerment: Engaging youth as shapers of the digital future*. IEEE Global Engineering Education Conference (EDUCON), 1705–1708. https://doi.org/10.1109/EDUCON.2017.7943078

Tissenbaum, M., Weintrop, D., Holbert, N., & Clegg, T. (2021). The case for alternative endpoints in computing education. *British Journal of Educational Technology, 52*(3), 1164–1177. https://doi.org/10.1111/bjet.13072

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

Vakil, S. (2014). A critical pedagogy approach for engaging urban youth in mobile app development in an after-school program. *Equity and Excellence in Education, 47*(1), 31–45. https://doi.org/10.1080/10665684.2014.866869

Vakil, S. (2018). Ethics, identity, and political vision: Toward a justice-centered approach to equity in computer science education. *Harvard Educational Review, 88*(1), 26–52. https://doi.org/10.17763/1943-5045-88.1.26

Vegas, E., & Fowler, B. (2020, August 4). What do we know about the expansion of K–12 computer science education? *Brookings*. https://www.brookings.edu/research/what-do-we-know-about-the-expansion-of-k-12-computer-science-education/

Villavicencio, A., Fancsali C., Martin, W., Mark, J., & Cole, R. (2018). *Computer science in New York City: An early look at teacher training opportunities and the landscape of CS implementation in schools.* The Research Alliance for New York City Schools. https://research.steinhardt.nyu.edu/scmsAdmin/media/users/ks191/CS4All/CS4All_Report.pdf

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

Wachter-Boettcher, S. (2017). Technically Wrong: Sexist apps, biased algorithms, and other threats of toxic tech. W. W. Norton & Company.

Wang, J., & Moghadam, S. H. (2017). Diversity barriers in K–12 computer science education: Structural and social. Proceedings of the 2017 *ACM SIGCSE Technical Symposium on Computer Science Education*, 615–620. https://doi.org/10.1145/3017680.3017734

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

Washington, A. N. (2020). When twice as good isn't enough: The case for cultural competence in computing. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education* (pp. 213–219). https://doi.org/10.1145/3328778.3366792

Waterman, K. P., Goldsmith, L., & Pasquale, M. (2020). Integrating computational thinking into elementary science curriculum: An examination of activities that support students' computational thinking in the service of disciplinary learning. *Journal of Science Education and Technology, 29*(1), 53–64. https://link.springer.com/article/10.1007/s10956-019-09801-y

Weintrop, D., Beheshti, E., Horn, M., Orton, K., Jona, K., Trouille, L., & Wilensky, U. (2016). Defining computational thinking for mathematics and science classrooms. *Journal of Science Education and Technology, 25*(1), 127–147. http://ccl.northwestern.edu/2015/Weintrop%20et%20al.%20-%202015%20-%20Defining%20Computational%20Thinking%20for%20Mathematics%20an.pdf

Weston, M., & Biin, D. (2013). The ancestor project: Aboriginal computer education through storytelling. *IADIS International Conference on Cognition and Exploratory Learning in Digital Age, CELDA 2013*, 85–92. https://doi.org/10.1007/978-3-319-05825-2_20

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

Yadav, A., Gretter, S., Good, J., & McLean, T. (2017). Computational thinking in teacher education. In P. Rich & C. Hodges (Eds.), *Emerging Research, Practice, and Policy on Computational Thinking, Educational Communications and Technology: Issues and Innovations* (pp. 205–220). Springer. https://doi.org/10.1007/978-3-319-52691-1_13

Yadav, A., Mayfield, C., Zhou, N., Hambrusch, S., & Korb, J. T. (2014). Computational thinking in elementary and secondary teacher education. *ACM Transactions on Computing Education (TOCE), 14*(1), 1–16. https://w3.cs.jmu.edu/mayfiecs/pubs/2014_Yadav_CT.pdf

Yadav, A., Ocak, C., Oliver, A. (under review). *Computational Thinking and Metacognition*.

Yadav, A., Stephenson, C., & Hong, H. (2017). Computational thinking for teacher education. *Communications of the ACM, 60* (4), 55-62. https://doi.org/10.1145/2994591