Computational Thinking

Enoch Hunsaker

Image by skeeze on Pixabay.com. CC0 Licence

Learning Objectives

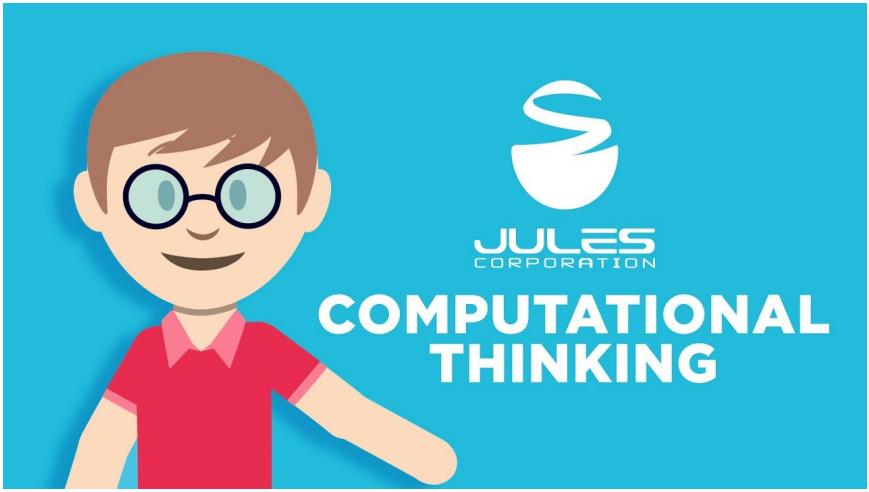
- Define computational thinking (CT);
- Explain the rationale for including CT as part of core curriculum;
- Understand research-based best practices for integrating CT with other core content at your grade level;
- Access a wide variety of resources designed to enable you to integrate CT at your grade level.

In today's high-tech and ever-changing world, it is increasingly clear that students need to be able to think critically and resolve complex and ill-defined problems in order to truly thrive in the environment where they are one day expected to live and work (Schön, 1987; Ventura, Lai, and DiCerbo, 2017). But while few would argue the

utility of teaching critical thinking and problem solving skills in schools, there is less consensus about how to do it, when to start, or what terms to use when teaching these important competencies.

One approach to teaching these skills is to teach computational thinking (CT). CT is particularly useful for the computer age, because it not only teaches critical thinking but also focuses on helping students "develop and employ strategies for understanding and solving problems in ways that leverage the power of technological methods to develop and test solutions" (ISTE, n.d., emphasis added). CT is the bread and butter of computer scientists, but it is also widely applicable for solving many other academic and non-academic problems.

CT is essentially a framework to describe a set of critical thinking and problem-solving skills, and it has gained significant traction as a viable and useful way of thinking about how to teach these skills in formal educational settings. While CT is not the only way to approach these skills, it provides a way of looking at problems so as to produce an automated or semi-automated solution that takes advantage of the unique affordances of computer technologies. It can also be beneficial in providing a common vocabulary, a wealth of resources, and a vibrant community of practice for teachers seeking to focus, coordinate, and improve efforts to guide rising generations in developing problem solving skills.



Watch on YouTube https://edtechbooks.org/-vY

Key Terms

Coding

> a language that a computer can use to complete a task or a set of instructions

Computational Thinking

> a problem solving process; typically broken down into decomposition, pattern recognition, abstraction, and algorithm design

Unplugged

> a coding lesson that does not require a computer

Why Integrate Computational Thinking?

More than ever, we live in a world that is informed and inundated by computer technology. This fact may conjure thoughts of smartphones and personal computers, but increasingly, many everyday and traditionally non-digital objects are being designed to operate via a computer program. Some of these objects include streetlights, car engines, watches, roads, car tires, shoes, and even cereal boxes (Hartigan, 2013).

As computer programs become more widespread, computer programming becomes an increasingly relevant skill, and many political bodies are recognizing this fact. Support for teaching computing in K-12 schools is growing in the U.S. and abroad. Several countries, including England, Finland, South Korea, and Australia, require that children learn computing or computational thinking (Rich, Jones, Belikov, Yoshikawa, and Perkins, 2017). Several U.S. states and districts have similar requirements (Partovi, 2017; EdSurge, 2016). The United States has not yet officially adopted such measures, but appears to be moving in that direction. For example, in 2017 the Trump administration announced a yearly investment of $200 million dollars into STEM education, noting that "the nature of our workforce has increasingly shifted to jobs requiring a different skill set, specifically in coding and computer science" (CNN Wire, 2017, emphasis added). Amazon, Facebook, and other major tech companies have committed a sum of over $300 million (over the period of five years) to the new initiative (Romm, 2017). Thus, increasing attention, interest, and enthusiasm are paid to the role that computer science education should have in our schools (Bers, Flannery, Kazakoff, and Sullivan, 2014; Rich et al., 2017; Sullivan and Bers, 2016; Yadav et al., 2016; Yadav et al., 2017).

But before computer programming - or coding, as it is sometimes called - many believe that today's youth (and adults) need computational thinking (CT) to better solve the problems of the 21st

century. CT may be considered a precursor to learning actual coding or computer programming skills. And while this is certainly true, it can also have a much broader application. The skills, attitudes, and approaches that make up CT are fundamental, universal, transferrable, and particularly appropriate and useful for the computer age. So, while a future computer programmer certainly needs CT, it is not necessarily true that everyone who learns CT should go on to learn coding. Rather, as computer technology becomes more embedded into the fabric of every industry, professionals in every industry need to be able to think in ways that leverage those computers to solve the problems of the future.

Learning computational thinking can benefit students both economically and academically. Each year there are far more computing jobs added than there are computer science graduates, with significant job growth projected for the foreseeable future (Bureau of Labor Statistics, 2018). Furthermore, studies have linked a host of academic benefits to learning CT, including improvement in student engagement, motivation, confidence, problem-solving, communication, and STEM learning and performance (Rich et al., 2017; Yadav et al., 2017).

What Is Computational Thinking?

Stephen Wolfram (2016) stated that the "intellectual core" of computational thinking "is about formulating things with enough clarity, and in a systematic enough way, that one can tell a computer how to do them." After gathering input from over 700 computer science educators, researchers, and practitioners, the International Society for Technology in Education (ISTE) and the Computer Science Teachers Association (CSTA) (2011) issued a joint statement in which they provided an operational definition of computational thinking, which involves both a problem-solving process and a series of dispositions and attitudes.

Computational thinking may imply a certain degree of facility and familiarity with computers, but it is much more than mere tech savviness. It is a combination of disciplined mental habits, attitudes of endurance, and essential soft skills. CT allows us to not merely consume technology, but to create with technology (Yadav, Hong, and Stephenson, 2016). It is not a way of making humans more like computers, but rather of empowering humans to use computers more effectively to solve the problems of the Computer Age (Wing, 2006).

The ISTE/CSTA (2011) definition is thorough, but it may also be useful for teachers to have a few key words to keep in mind when planning lessons, guiding discussions, commenting on student work, etc. The following table is derived from the documentation of various organizations that seek to define and categorize CT in a useful way for educators (CAS Barefoot, 2014; Google, n.d.b; ISTE, 2014). This is not intended to be comprehensive, but it does provide a reasonably complete snapshot of the most crucial components of CT.

Components of CT (CAS Barefoot, 2014; Google, n.d.b; ISTE, 2014)

Skills

- **Decomposition:** Breaking down data, processes, or problems into smaller, manageable parts
- **Pattern Recognition:** Observing patterns, trends, and regularities in data
- **Abstraction:** Making a problem more understandable by reducing unnecessary detail.
- **Algorithm Design:** Developing the step by step instructions for solving this and similar problems
- **Evaluation:** Ensuring that your solution is a good one.

Attitudes

- **Confident:** believing in one's own ability to solve problems
- **Communicative:** willing and able to communicate effectively with others.
- **Flexible:** able to deal with change and open-ended problems

Approaches

- **Tinkering:** experimenting and playing
- **Creating:** designing and making
- **Debugging:** finding and fixing errors
- **Persevering:** keeping going
- **Collaborating:** working together.

Review These Terms on Quizlet

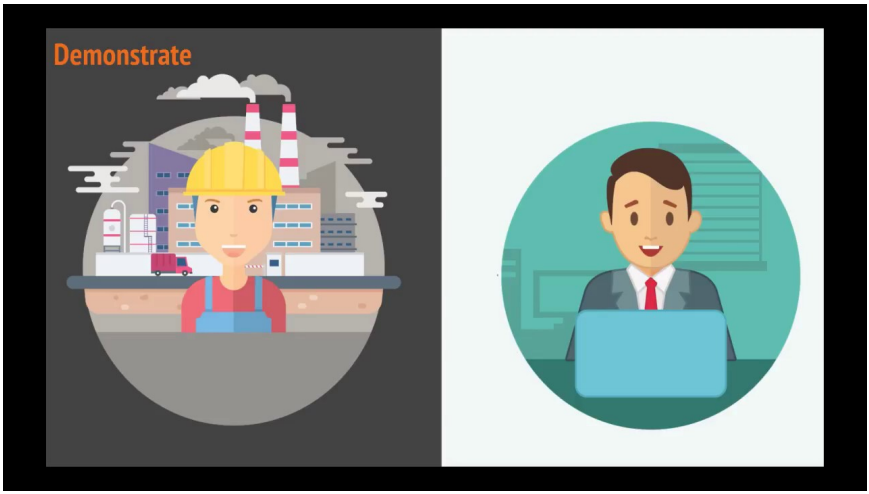
Thought Exercise: Problem-Solving Models

Computational Thinking is an effective model of problem solving, but it is only one model. Others include scientific thinking or the scientific

method (which is used by scientists to answer questions about how and why the world works) and design thinking (which is used by designers and engineers to design objects and experiences). Consider the steps of each of these widely-used problem-solving models:

Computational Thinking	Scientific Thinking	Design Thinking
1. Decomposition 2. Pattern Recognition 3. Abstraction 4. Algorithm Design 5. Evaluation	1. Ask a Question 2. Conduct Research 3. Form a Hypothesis 4. Test the Hypothesis 5. Record & Analyze Data 6. Draw a Conclusion 7. Communicate Results	1. Empathize 2. Define 3. Ideate 4. Prototype 5. Test

Watch this video to better understand these processes:



Questions to Ponder:

- What might be the advantages and disadvantages of each problem-solving model?
- Could any model be applied to any problem? How might the types of results expected from each model differ?
- Are some problems better suited to one method than another?

Why Integrate CT in Early Childhood and Elementary Education

Establishing a way of thinking takes time, so if CT is to be truly grasped by the professionals of the future, they need to be familiarized with these concepts early and often throughout their academic career (Yadav, Mayfield, Zhou, Hambrusch, and Korb, 2014). Computational thinking is "cross-disciplinary" in nature (Yadav et al., 2017), so it makes sense to start teaching it in elementary or even preschool, where all the subjects are naturally blended together for the students within the same environment.

Studies have shown that children as young as preschool-age (approximately 4) have been able to successfully learn basic CT concepts (Sullivan and Bers, 2016; Bers et al., 2014). Studies also show that learning this can be "an engaging and rewarding" experience for the students (Bers et al., 2014).

Technology permeates our world and experience. Bers, Seddighin, and Sullivan (2013) have argued that because technology is an integral part of children's experience, early childhood education should include the study of technology. Teaching computational thinking is one way to do just that. In early childhood education, we often focus on understanding the natural world, which is certainly worth studying, but the man-made world is also worth studying. Most children are more familiar with cell phones than with polar bears, yet teachers are more likely to teach a unit on polar bears than on cell phones. We can and should study both (Bers et al., 2013).

Some early childhood practitioners may question the appropriateness of teaching computational thinking to very young students, due to prevalent and well-founded concerns about giving too much screen time to young children (NAEYC and Fred Rogers, 2012). However, these concerns can be reduced by understanding that (1) there is a wide variety of CT activities that do not require the use of a screen (e.g., unplugged activities, screenless robots) and (2) that even activities that do involve screen time can--and should--be constructed as interactive, rather than non-interactive uses of technology (NAEYC and Fred Rogers, 2012).

Why Integrate CT in Secondary Education

Some secondary educators may understandably feel that, unless they are planning to get an endorsement in information technology education, computational thinking has little to do with them. However, teaching CT concepts in English, history, math, science, second languages, and other core and elective subjects is actually a great way to "support problem solving across all disciplines" (Google, n.d.a) Grover (2018) argues, "Like any skill, CT is best taught and learned in context, and embedded into class subjects."

If CT education is embedded across multiple subject areas at the same school, it has additional advantages, such as helping students to "make powerful connections between their classes and beyond" and "have a rich toolkit to draw from that crosses traditional subject borders" when faced with problems that are difficult to categorize within a traditional subject area (Sheldon, 2017).

Thought Exercise: CT - A 21st Century Literacy?

Many claim that computational thinking is an essential 21st Century Literacy which ought to be taught alongside reading, writing, and arithmetic in our schools. While you don't necessarily have to agree with this assessment, it is important to understand the rationale behind it.

Consider the following statements from CT education proponents, then consider the questions listed below:

> Just as basic literacy in math and science are considered essential for all children to understand how the world works, education must also address the development of knowledge and skills pertaining to computing, which is now so integrally intertwined with every profession (Grover, 2018).
>
> Computational thinking is a fundamental skill for everyone, not just for computer scientists. To reading, writing, and arithmetic, we should add computational thinking to every child's analytical ability. Just as the printing press facilitated the spread of the three Rs, what is appropriately incestuous about this vision is that computing and computers facilitate the spread of computational thinking (Wing, 2006).

Questions to Ponder:

- What is a "literacy"?
- Do you agree that computational thinking is a literacy?
- Do you agree that it is as fundamental as reading, writing, and math in the 21st Century? Why or why not?

How to Effectively Integrate CT into Your Classroom

This section is intended as a reference. Feel free to focus on reading the parts that are most relevant to you.

Research-Based Effective Practice for CT Integration

Teaching computational thinking has traditionally been viewed as a primarily constructionist endeavor (Bers et al., 2014; Buss and Gamboa, 2017). Constructionism posits that "children can learn deeply when they build their own meaningful projects in a community of learners and reflect carefully on the process" (Bers et al., 2014). In particular, the constructionist approach described by Seymour Papert "provides children the freedom to explore their own interests through technologies (Bers, 2008) while investigating domain-specific content learning and also exercising metacognitive, problem-solving, and reasoning skills" (Bers et al., 2014).

Within this broadly constructionist framework, a variety of instructional principles and methods have been identified as effective practices for teaching computational thinking. These practices can be adapted to most grade levels and subject areas.

- **Modeling**. Teachers should set an example of learning by modeling their own understanding, learning, and progress in computational thinking. Especially in the early stages, they should also model the computational thinking process for students so they understand what the learning, reflection, and revision look like (Highfield, 2015).
- **Integrating.** Teachers should collaborate with other teachers to facilitate the completion of interdisciplinary culminating projects (Bers et al., 2014).
- **Releasing Responsibility Gradually.** When teaching CT,

educators should start with direct instruction, move to a simple guided activity, then issue an open-ended challenge or problem (Buss and Gamboa, 2017). Teachers should then continue to guide behavior, even while working/playing as a team (Highfield, 2015).

- **Encouraging.** Insofar as possible, teachers should provide "encouragement and problem-solving hints and tips," rather than outright answers (Buss and Gamboa, 2017).
- **Questioning.** Rather than providing answers directly, teachers should ask "probing questions" before, during, and after learning activities (Buss and Gamboa, 2017; See also Highfield, 2015) These questions should encourage students to reflect on their learning and might begin with phrases like the following (Buss and Gamboa, 2017):
    - "What if you were to..."
    - "How would you..."
    - "Have you considered..."
- **Fostering alternative problem-solving.** Teachers should promote alternative ways of modeling a problem (Buss and Gamboa, 2017), such as
    - Drawing out solutions on paper.
    - Discussing alternative solutions as teams.
    - Relating challenges to more familiar circumstances.
- **Using CT vocabulary across the curriculum** (Yadav et al., 2014). This can reinforce students' understanding of the terms and help them see their applicability across the curriculum and in daily life. For example, a teacher might refer to a set of rules or procedures as an "algorithm"; invite students to create an "abstraction" of how they feel; or emphasize that you are practicing "pattern recognition" skills.

How and When to Use Technology in CT Education

Teachers won't be utilizing technology every time they want to teach CT: they may be simply referencing CT vocabulary, helping students

learn perseverance, or engaging students in an unplugged coding activity. However, since CT does involve "leverag[ing] the power of technological methods" (ISTE, 2014), a progressive program of CT instruction will inevitably lead to some integration of technological devices.

Just as PIC-RAT can be a valuable heuristic for evaluating classroom technology integration and designing technological learning experiences, it can also help guide educators in making decisions about how and when to use technology in the CT education process. In general, teachers should strive to provide learning experiences that guide students toward the creative and transformative ends of the PIC and RAT spectrums.

For example, an elementary teacher wanting to integrate CT into her curriculum might begin by explaining some key CT concepts to her students, such as decomposition and abstraction. She might then introduce a mathematical word problem that requires the students to break the problem into component parts and filter out unnecessary detail. So far, it has not been necessary to use technology, and most uses (e.g., an online worksheet) would likely have been passive or interactive replacements of traditional practice.

However, as the teacher helps her students to learn additional aspects of math and CT, she may see organic ways to integrate technology in creative and transformative ways. For instance, she may feel that the best way to teach shape properties and algorithm design is to bring some codable robots into the classroom and have the students program them to draw regular polygonal shapes. At first, the students may have some interactive time with the robots, simply so they can learn how they function. Eventually, however, their use will become creative as they design an algorithm to meet the teacher's challenge. Such an experience may transform the learning in several ways, such as

- helping the students make connections between math and computer science that they would not have made with mere worksheets;
- deepening the students' perception of the relevance of both math and coding;
- engaging students in content they might otherwise have found routine and boring.

CT in Early Childhood and Elementary Education

In addition to other research-based effective practices, the following ideas, examples and resources may be useful in an early childhood teaching context.

Ideas

- **Teach CT through Coding.** While learning CT does not require learning code, coding can be a particularly effective vehicle to introduce CT to young students, as it can help students to visualize and experience the concepts in a more concrete way. In particular, using "unplugged" activities and codable robots has been particularly effective for this age group.
    - ***Unplugged Activities.*** Unplugged activities are activities that teach coding concepts without involving a computer. Students may use a paper and pencil, manipulatives, or even their own bodies to experience coding principles in a deeper way. These activities naturally allow for conversations about and connections with CT skills, attitudes, and approaches.
    - ***Codable Robots.*** Codable robots can extend the coding and CT experience of young students. Robots provide lots of opportunities to integrate mathematical and engineering concepts into their coding and CT knowledge, and the connections students make can

actually support their learning in traditional core subjects.

Examples

- Students learn about algorithms when the teacher explains what they are using the simple example of the routine students follow when they get up and come to school in the morning. Students then write their own algorithms for planting a seed and test it out with real seeds and soil (Randles, 2017).
- A teacher uses Ozobots (small robots programmable with paper and a marker) to teach her students about states of matter, geography, and coding. The ozobot moves across a map and the students must program it to move slower in cold regions and faster in warm regions. They need to practice communication, debugging, and algorithm design in order to make this work (Randles, 2017).
- Students create a math game with engineering toys and test every circuit before moving on to the next activity. If something doesn't work, they "debug" it. Students learn perseverance and communication skills in working together (Berdik, 2015).
- Students stuck in a difficult problem look toward a teacher for help, but the teacher directs them to "use prior knowledge, explore and work through it." Deep learning occurs as the students learn to persevere, collaborate, and rely on the CT process (Berdik, 2015).
- Students and the teacher together create an "algorithm" for the procedure of leaving the classroom.

Secondary Education

In addition to other research-based effective practices, consider the following ideas/examples for teaching CT in your specific subject area.

Language Arts & Foreign Language

- Students completing a short story unit are learning literary elements (e.g., plot, point of view, irony, etc.). Their assignment is to write a literary analysis in which they explore how a particular literary element influences a work. They utilize many CT skills throughout this unit, such as
    - Representing plot structure through abstraction (i.e., a plot diagram)
    - Logical organization and analysis of data in order to support their thesis.
    - Communicating and collaborating with others in class discussions
    - Th students also relate these skills to what they are learning in other subject areas (Barr, Harrison, and Conery, 2011).
- Students use logic to put together a jumbled story in correct sequence (Grover, 2018).
- Students identify patterns for different sentence types and rules for grammar (Grover, 2018).
- Students use first-order logic to arrive at conclusion based on given facts (Grover, 2018).
- Student construct social networks to analyze stories (Grover, 2018).
- Students program a story with alternative pathways ("Choose your own adventure") (Grover, 2018).
- Students analyze how algorithms affect dialogue and news feeds in social media (Angevine, 2018).
- Student collaborate to build a story, identify any "bugs" in the story, and fix those bugs to give the story a more logical flow. (Google, n.d.c)

Social Studies

- Students compare their modern lifestyle with the lifestyles of

children from another era. They simulate the experience of children from the other era by writing about it in a blog. The teacher calls attention to the fact that they are practicing skills relevant to computational thinking, such as organizing and analyzing data logically, and representing data through an abstraction (Barr et al., 2011).
- Students review data and identify patterns and trends in wars and other historical events. The teacher helps the students recognize that they are practicing the CT skill of "pattern recognition." Students also create visualizations of these patterns and trends, and the teacher refers to them as "abstractions" (Grover, 2018).
- Students "create a simulation to study relationships in social science phenomena such as women's education and health." This is an abstraction (Grover, 2018).
- Students create models or "abstractions" for social systems, social networks, or social choice (Grover, 2018).
- Students use primary source data to study patterns of voting rights in the nation (Angevine, 2018).

Engineering

- Students look at a map of escape routes for the school. They recognize that the map is an "abstraction" and discuss how they could create an algorithm that would define the fastest way out of the school in the event of an emergency (Barr et al., 2011).
- Students compare and contrast the design thinking problem solving process and the computational thinking problem solving process and explore how each method can give them unique insights and solutions for engineering problems. They also discuss how the methods can be melded to provide more complete and better solutions.
- Students use engineering computer software to design structures.
- Students engage in a real-world construction simulation task as

teams. They need to practice the skills of abstraction (drawing a design for the project), decomposition (breaking down the tasks that need to be completed). They also utilize CT approaches such as collaborating, creating, and (possibly) tinkering and debugging.

Music

- Students studying the diatonic scale and the concept of pitch use Scratch (a programming language) to create an "abstraction" of a xylophone. They also develop persistence as they work through a difficult problem (Barr et al., 2011).
- Students use algorithms to study intervals, rhythm, and composition (Angevine, 2018)
- Students explore musical patterns and create algorithms that can write a song (Google, n.d.c)

Mathematics

- Students model functions in algebra through programs (compare them to functions in programs) (Grover, 2018).
- Students write an algorithm (or precise sequence of steps) on how to do matrix multiplication or how to solve a quadratic equation (Grover, 2018).
- Students use decomposition to solve word problems (Grover, 2018).
- Students express generalizations (as algebraic representations) by identifying patterns (Grover, 2018).
- Students interpret and visualize statistics of an athlete's performance (Angevine, 2018).
- Students use robots to create a program that can draw any regular polygon of any regular size. They also explore how slight variations in the program can create fractal shapes.
- Students use basic patterns to label key points on the unit circle in terms of degrees, and then follows a similar process to

relabel these points in terms of radians. Students can then develop an algorithm to convert between degrees and radians based on the patterns they used to count their way around the unit circle. (Google, n.d.c)
- Students use CT concepts to explore the linear association between variables using two sets of data. Students will read data in a spreadsheet and in a graph and identify positive and negative linear association based on the shape of the graph. (Google, n.d.c)

Sciences

- Do a species classification with explicit "If-Then" logic (younger grades) (Grover, 2018).
- Build a computational model of a physical phenomenon (Grover, 2018).
- Instead of playing with or manipulating pre-developed software simulations of scientific phenomenon, create (program) computational models and simulations to study and interrogate phenomena (Grover, 2018).
- Students use computational models and processes to predict the effects of removing a species from the ecosystem (Angevine, 2018).
- Students create simulations and abstractions that model safe and unsafe roller coaster designs (Angevine, 2018).
- STudents model (i.e., abstract) different scientific laws and phenomena using CT concepts and approaches (Google, n.d.c).

Family and Consumer Science

- In a child development course, students engage in metacognition about the computational thinking process, and how it can help them to solve problems and make decisions in their own lives.
- In a sewing class, students observe common patterns in certain

types of clothing. Later on, they create a pattern (i.e., an algorithm) for sewing a shirt. They also include diagrams (abstractions) within their pattern instructions.
- In a foods class, students explore and discuss patterns across cake recipes (e.g., classes of ingredients included, order of steps, baking times and temperatures). Students may also create their own cake recipe (algorithm) and test (evaluate) it based on a set of criteria of their choosing.
- In a personal finance class, students use computer software to track their spending over several months. They then use that data to find patterns and create graphs (i.e., abstractions) of spending patterns that can inform their future decisions.

Dance & Physical Education

- Students learning a variety of dance moves create their own dance (algorithm) by stringing them together.
- Students in P.E. learn about the wide variety of computational resources (e.g., apps, wearables) that can help them monitor and improve their physical wellbeing and personal health habits. They use data they collect from some of these sources to create reports (abstractions) to help them make decisions about what habits they will seek to develop.

CT Learning and Lesson Planning Resources Resources

The following table provides a number of resources for learning more about computational thinking and planning lessons that integrate its components.

Resource	Format	Grade Recommendation		
		PreK-2	3-6	7-12
CT Learning & Leadership				
Computational Thinking Leadership Toolkit (ISTE)	PDF	✔	✔	✔

Resource	Format	Grade Recommendation		
		PreK-2	3-6	7-12
Digital Promise's 10 CT-related micro-credentials	Web	✔	✔	✔
BYU's Understanding Computational Thinking and Teaching Computational Thinking badges	Web	✔	✔	✔
Google for Education - Computational Thinking for Educators free online course.	E- Course	✔	✔	✔
Integration Activities Across the Curriculum				
CAS Barefoot's Computational Thinking page	Web	✔	✔	
Google for Education - Exploring Computational Thinking: CT Materials	Web			✔
Wonder Workshop's Code to Learn Lesson Library	Web	✔	✔	
Computational Thinking Teacher Resources, 2nd Edition (ISTE)	PDF	✔	✔	✔
CT Vocabulary and Progression Chart (ISTE)	PDF	✔	✔	✔
Understanding Developmentally-Appropriate Integration				
NAEYC's Technology & Media website	Web	✔		
Fred Rogers Center Website	Web	✔		
Erikson Institute	Web	✔		
Unplugged Activities				
Code.org's CS Fundamentals Unplugged	Web	✔	✔	
CS Unplugged	Web	✔	✔	
Robotic Coding Activities				
Coding as a Playground : Programming and Computational Thinking in the Early Childhood Classroom by Marina Bers	Book	✔		
Robotics for Young Children: STEM Activities and Simple Coding by An Gadzikowski	Book	✔		
Ozobot Lesson Library	Web	✔	✔	
CAS Barefoot's Bee-bot Activity Guide	PDF	✔	✔	
Wonder Workshop's Learn to Code Curriculum	Web	✔	✔	
Ontario Math curriculum, Grades 1-8 Sphero Lesson Plans	PDF	✔	✔	✔
Block-Based Coding Activities & Tools				
Code.org's Pre-Reader Express Course and courses A-F	Web Games	✔	✔	✔
BootUp Curriculum for Scratch and Scratch Jr.	Web	✔	✔	
Scratch	Web Tool	✔	✔	✔
Kodable	Web Games	✔	✔	

Conclusion

Computational thinking is a method of solving problems that is both widely applicable throughout the K-12 curriculum and increasingly

relevant in the 21st Century. Integrating CT into traditional core and elective subject areas can help students to make important cross-curricular connections, improve their academic performance, and develop important skills for creating solutions in the wide variety of vocations in which they will one day engage. As the popularity and relevance of CT becomes more apparent, many countries, states, and institutions are adopting it into their curriculum, so teachers should be aware of how this affects them, how it may affect them in the future, and the variety of resources they can access as needed. They are also encouraged to become as familiar as they can with CT skills, attitudes, and approaches, and to develop these competencies in their personal and professional lives.

References

Angevine, C. (2018, February 22). Advancing computational thinking across K-12 education. Retrieved from https://edtechbooks.org/-YY

Barr, D., Harrison, J., & Conery, L. (2011). Computational thinking: A digital age skill for everyone. Learning & Leading with Technology, 38(6), 20-23. Retrieved from https://edtechbooks.org/-HQ

Berdik, C. (2015, November 23). How one school district works computational thinking into every grade and class. Retrieved from https://edtechbooks.org/-Cj

Bers, M.U. (2008). Blocks to robots: Learning with technology in the early childhood classroom. New York, NY: Teachers College Press.

Bers, M.U., Seddighin, S., & Sullivan, A. (2013). Ready for robotics: Bringing together the T and E of STEM in early childhood teacher education. Journal of Technology and Teacher Education, 21(3), 355-377.

Bers, M.U., Flannery, L., Kazakoff, E. R., & Sullivan, A. (2014). Computational thinking and tinkering?: Exploration of an early childhood robotics curriculum. Computers & Education, 72, 145-157. https://edtechbooks.org/-HN.

Bureau of Labor Statistics (2018). Occupational outlook handbook. Retrieved from https://edtechbooks.org/-yr

Buss, A., & Gamboa, R. (2017). Teacher transformations in developing computational thinking: Gaming and robotics use in after-school settings. In P.J. Rich & C.B. Hodges (Eds.), Emerging research, practice, and policy on computational thinking (pp. 189-203). Cham, Switzerland: Springer. Retrieved from https://edtechbooks.org/-UN

CAS Barefoot (2014). Computational thinking. Retrieved from https://edtechbooks.org/-Wy.

CNN Wire. (2017, September 25). President Trump announces yearly investment of $200M for STEM expansion. Retrieved from Fox News: https://edtechbooks.org/-tB

EdSurge. (2016). Computer science for all. Retrieved from https://www.edsurge.com/research/special-reports/state-of-edtech-2016/k12_edtech_trends/computer_science

Google (n.d.a). Exploring computational thinking? Retrieved from https://edtechbooks.org/-PU

Google (n.d.b). What is computational thinking? Retrieved from https://edtechbooks.org/-gU

Google (n.d.c). CT materials. Retrieved from https://edtechbooks.org/-tP

Grover, S. (2018, March 13). The 5th 'C' of 21st century skills? Try computational thinking (not coding. Retrieved from EdSurge News:

https://edtechbooks.org/-Pz

Hartigan, M. (2013, August 27). 10 everyday objects that can be programmed to run code. Retrieved from https://edtechbooks.org/-cK

Highfield, K. (2015). Stepping into STEM with young children: Simple robotics and programming as catalysts for early learning. In C. Donohue (Ed.), Technology and digital media in the early years: Tools for teaching and learning (pp. 150-161). New York, NY: Routledge.

ISTE (2014, September 11). Computational thinking for all. Retrieved from https://edtechbooks.org/-yE ISTE. (n.d.). Standards for students. Retrieved from https://edtechbooks.org/-XB.

ISTE, & CSTA. (2011). Operational definition of computational thinking for K-12 education. Retrieved from https://edtechbooks.org/-cV

NAEYC, & Fred Rogers Center for early Learning and Children's Media. (2012). Technology and interactive media as tools in early childhood programs serving children from birth through age 8. Retrieved from https://edtechbooks.org/-zJ

Partovi, H. (2017). Should computer science be a mandatory class in U.S. high schools? Retrieved from https://www.quora.com/Should-Computer-Science-be-a-mandatory-part-of-a-high-school-curriculum/answer/Hadi-Partovi

Randles, J. (2017, January 27). 3 easy lessons that teach coding and computational thinking. Retrieved from https://edtechbooks.org/-Lc

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

Romm, T. (2017, September 26). Amazon, Facebook and others in tech will commit $300 million to the White House's new computer science push. Retrieved from https://edtechbooks.org/-Uu

Schön, D. A. (1987). Educating the reflective practitioner: Toward a new design for teaching and learning in the professions. Ann Arbor, MI: Wiley.

Sheldon, E. (2017) Computational thinking across the curriculum. Retrieved from https://edtechbooks.org/-Qt

Sullivan, A., & Bers, M.U. (2016). Robotics in the early childhood classroom: Learning outcomes from an 8-week robotics curriculum in pre-kindergarten through second grade. International Journal of Technology and Design Education, 26(1), 3-20. https://edtechbooks.org/-LK

Ventura, M., Lai, E., & DiCerbo, K. (2017). Skills for today: What we know about teaching and assessing critical thinking [White paper]. Retrieved March 29, 2018, from Partnership for 21st Century Learning: https://edtechbooks.org/-hH

Wing, J. M. (2006). Computational thinking. Communications of the ACM, 49(3), 33-35. https://edtechbooks.org/-jB

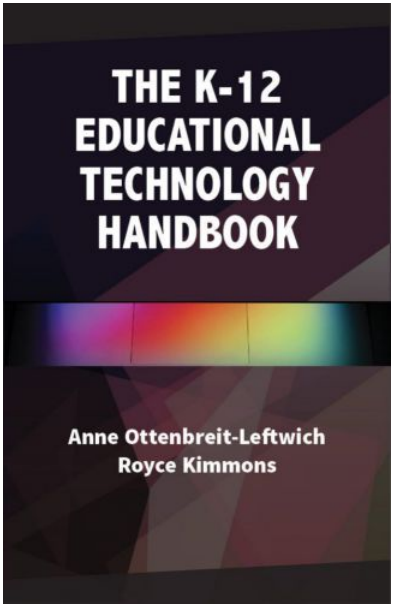
Wolfram, S. (2017, June 16). How to teach computational thinking. Retrieved from https://edtechbooks.org/-jZ

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), 5.

Yadav, A., Hong, H., & Stephenson, C. (2016). Computational thinking for all: Pedagogical approaches to embedding 21st century problem solving in K-12 classrooms. TechTrends, 60(6), 565-568.

https://edtechbooks.org/-vq

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



Hunsaker, E. (2020). Computational Thinking. In A. Ottenbreit-Leftwich & R. Kimmons (Eds.), *The K-12 Educational Technology Handbook*. EdTech Books. Retrieved from https://edtechbooks.org/k12handbook/computational_thinking

are free to do with it as you please as long as you properly attribute it.