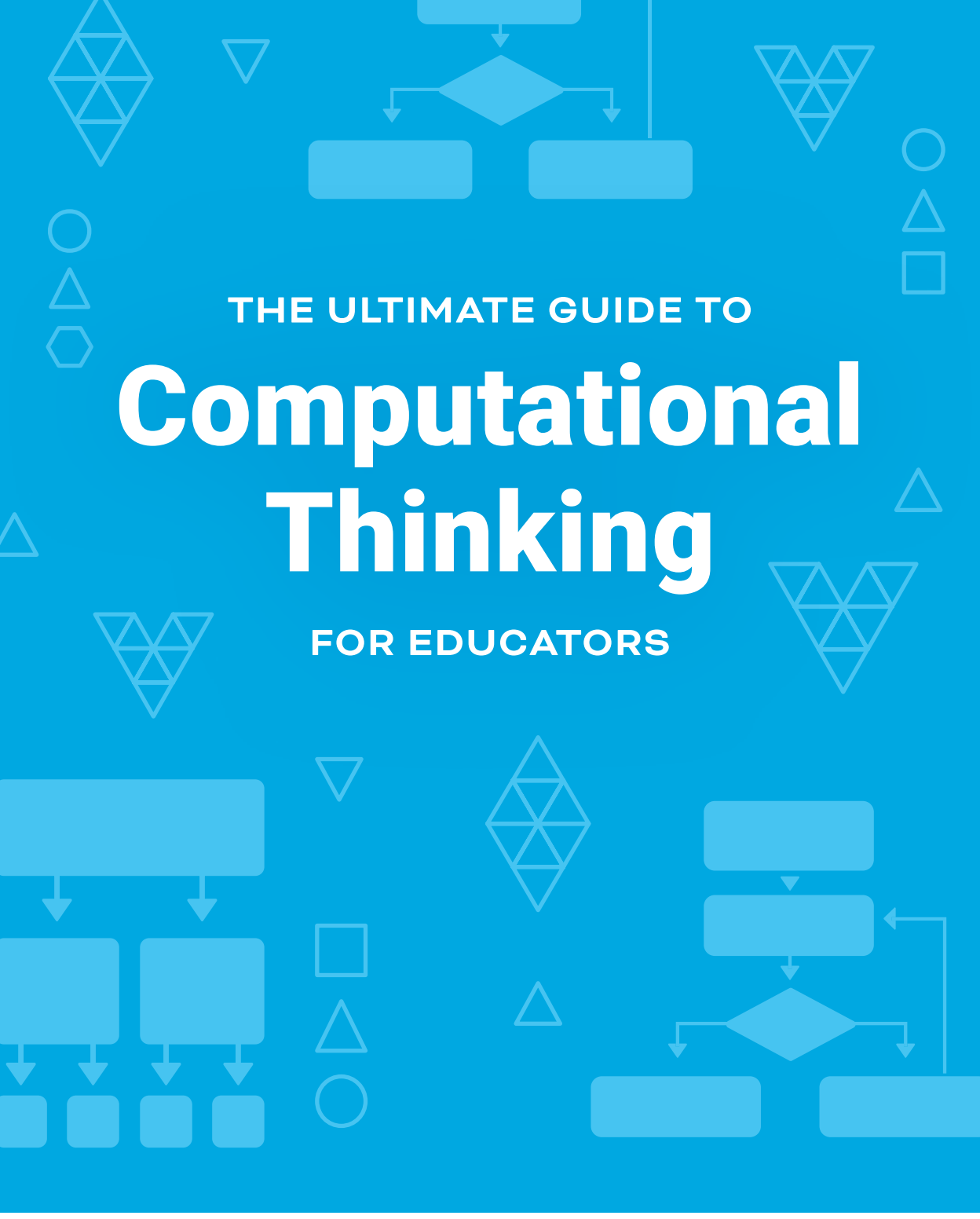


Learning.com

What is computational thinking?

Problem Solving.

More specifically, computational thinking is a set of skills and processes that guide problem solving.

What makes this especially different from other problem-solving processes is that it, in the end, results in an algorithm, which is a series of steps a person or computer uses to perform a task or solve a problem. Computational thinking is derived from the process computer scientists use to develop code and communicate with computers through algorithms.

So, computational thinking is coding?

Not quite. While computational thinking is the problem-solving process that can lead to code, coding is the process of programming different digital tools with algorithms. Coding is a means to apply solutions developed through the processes of computational thinking. Algorithms, in the case of coding, are a series of logic-based steps that communicate with digital tools and help them execute different actions.

However, computational thinking results in algorithms for both computers and people, making it much more broadly applied with and without technology. At its core, the steps of the computational thinking process enable people to tackle large and small problems.

The Computational Thinking Process

Computational thinking is a map from curiosity to understanding that ensures the problem-solving process can be replicated or automated in the future.

The computational thinking process includes four key concepts:

Decomposition — Break the problem down into smaller, more manageable parts.

Pattern Recognition — Analyze data and identify similarities and connections among its different parts.

Abstraction — Identify the most relevant information needed to solve the problem and eliminate the extraneous details.

Algorithmic Thinking — Develop a step-by-step process to solve the problem so that the work is replicable by humans or computers.

The process starts with data as the input and through a series of steps, we – like computers (hence the name) – process the information and produce some sort of output to the problem. In this way, computational thinking results in an answer to whatever question was asked at the outset and a systematic process for how students arrived at the answer. Moreover, computational thinking is about the process itself just as much as it is about solving the problem.

Both ‘plugged’ and ‘unplugged,’ computational thinking underscores the course of student learning in an era in which education is moving from content acquisition to higher-order thinking skills. Beyond this, computational thinking requires students to be mindful and intentional throughout the problem-solving process, which helps them develop persistence and a growth mindset.

As Marcos Navas, a technology facilitator with the Union City School District in New Jersey, explained:

> **“We are all computational thinkers and computer scientists; our brains naturally recognize patterns, create algorithms, and debug solutions. When a problem arises, I tell my students to figure it out and work with their peers. We do a lot of hand holding, but we need to challenge students to solve it. That’s not just a tip for coding; it’s a lifelong skill they need.**

Indeed, computational thinking is a lifelong skill that can and should be built throughout the student learning experience. In the remainder of this eBook, we will go into more depth about each component of computational thinking and offer real-world examples for integrating them in any subject area. We will also explore why computational thinking is essential for future readiness by cultivating digital skills and problem-solving prowess.

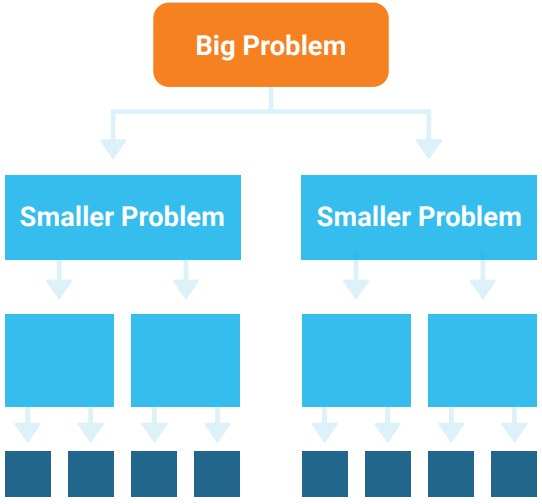
Let’s dive in!

Decomposition

The power of computational thinking starts with decomposition, which is the process of breaking down complex problems into smaller, more manageable parts. With decomposition, problems that seem overwhelming at first become much more manageable.

> **If you can't solve a problem, then there is an easier problem you can solve: find it.**
>
> GEORGE PÓLYA

Problems we encounter both in the course of student learning and throughout our daily lives are ultimately comprised of smaller problems we can more easily address. This process of breaking down problems enables us to analyze the different aspects of them, ground our thinking, and guide ourselves to an end point.



Examples of Decomposition in Everyday Life

Decomposition is something we inherently do in our daily lives, even if we don't realize it.

If you hosted a holiday dinner, you used decomposition to select the menu, enlist support from others in the kitchen, task people with what to bring, determine the process by which to cook the different elements, and set the time for the event.

If you went to the grocery store for said holiday dinner you used decomposition to build your grocery list, guide the direction you took as you meandered the aisles, the route you followed to and from the store, and the vehicle in which you drove.

If you've implemented a new program or initiative at your school, you used decomposition to build your strategic plan, which included the program's vision, strategy for gaining buy-in, annual goals, and everything else involved.

Examples of Decomposition in Curriculum

Indeed, decomposition is a powerful tool that guides how we approach projects and tasks regularly. And it is also something employed in student learning. Here are some examples for accentuating these in curriculum.

English Language Arts	Students analyze themes in a text by first answering: Who is the protagonist and antagonist? Where is the setting? What is the conflict? What is the resolution?
Mathematics	Students find the area of different shapes by decomposing them into triangles.
Science	Students research the different organs in order to understand how the human body digests food.
Social Studies	Students explore a different culture by studying the traditions, history, and norms that comprise it.
Languages	Students learn about sentence structure in a foreign language by breaking it down into different parts like subject, verb, and object.
Arts	Students work to build the set for a play by reviewing the scenes to determine their setting and prop needs.

Examples of Decomposition in Computer Science

Then, from a computer science and coding perspective, decomposition can come into play when students are programming a new game. For example, students need to consider the characters, setting, and plot, as well as consider how different actions will take place, how it will be deployed, and so much more.

Character Actions

Standard	Movement	Attack	Special
Left	Walk	Spell	Mix Potion
Right	Jump	Confuse	Regenerate
Up	Crouch	Throw	
Down		Punch	

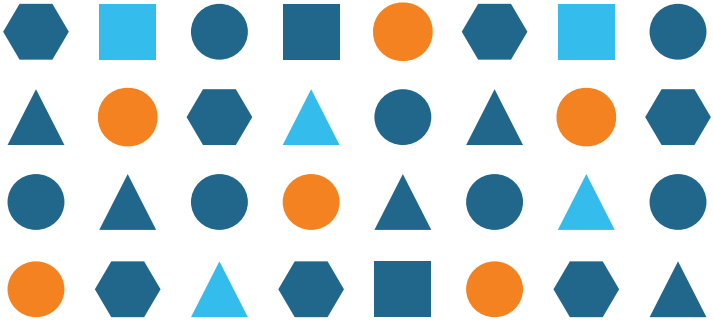
It's hopefully clear that decomposition is deeply ingrained in how we function daily and address problems both big and small. The concept already exists with students, but students need to learn how to recognize this process as it happens and leverage it when they feel overwhelmed in the case of a problem, task, or project. Decomposition teaches students to embrace ambiguity and equips them with the confidence to learn new things.

Pattern Recognition

As it sounds, pattern recognition is all about recognizing patterns. Specifically, with computational thinking, pattern recognition occurs as people study the different decomposed problems.

> **There are common ways we see patterns. Patterns are the laws of nature and life that present themselves in all disciplines of life — from the smallest microorganism to macrocosm…While patterns aren't always apparent, they are continuous and autonomous.**
>
> AMY OESTREICHER

Through analysis, students recognize patterns or connections among the different pieces of the larger problem. These patterns can be both shared similarities and shared differences. This concept is essential to building understanding amid dense information and goes well beyond recognizing patterns amongst sequences of numbers, characters, or symbols.



Examples of Pattern Recognition in Everyday Life

Pattern recognition is the foundation of our knowledge. As infants, we used patterns to make sense of the world around us, to begin to respond verbally and grow our language skills, to develop behavioral responses, and to cultivate connections in this world.

Beyond this, pattern recognition also occurs when scientists try to identify the cause of a disease outbreak by looking for similarities in the different cases to determine the source of the outbreak.

Additionally, when Netflix recommends shows based on your interests or a chat bot pesters you on a website, the technology (Artificial Intelligence and Machine Learning) relies on pattern recognition.

Personally, I used pattern recognition recently when I created a food diary for my dog to identify the source of his newest allergic reaction. The culprit? Fish. And this is to add to a long list including red meat, chicken, bison, and grains. But I digress.

Examples of Pattern Recognition in Curriculum

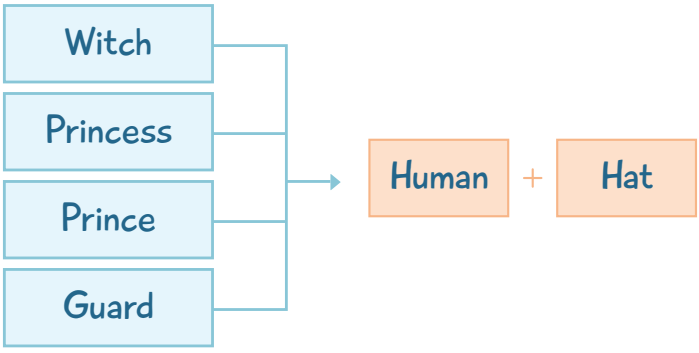
Pattern recognition applies in the classroom as well.

English Language Arts	Students begin to define sonnets based on similarities in separate examples.
Mathematics	Students recognize the specific formulas used to calculate slopes and intercepts.
Science	Students classify animals based on their characteristics and articulate common characteristics for the groupings.
Social Studies	Students identify the potential impact different economic trends reap by looking at data.
Languages	Students group different words in a foreign language by looking at their roots to build a better understanding of vocabulary.
Arts	Students categorize paintings based on commonalities between artists' aesthetics and detail key characteristics that each grouping presents.

Examples of Pattern Recognition in Computer Science

And in computer science, pattern recognition helps students identify similarities between decomposed problems. If they are coding a game, they may recognize similar objects, patterns, and actions. Finding these allows them to apply the same, or slightly modified, string of code to each, which makes their programming more efficient.

Character Design



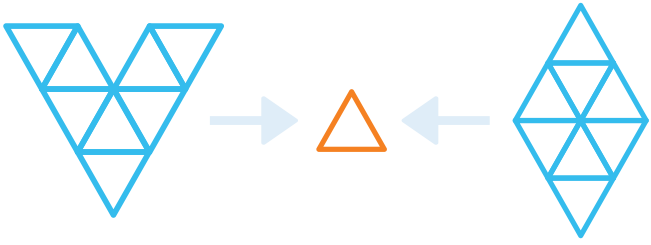
Through the quest to build understanding in unfamiliar scenarios or in the face of uncertainty, students learn to persist through iteration and experimentation and accept that failure and struggle are a part of the learning process.

Abstraction

Also called, pattern generalization, abstraction enables us to navigate complexity and find relevance and clarity at scale. Decomposition and pattern recognition broke down the complex, and abstraction figures out how to work with the different parts efficiently and accurately. This process occurs through filtering out the extraneous and irrelevant in order to identify what's most important and connect each decomposed problem.

> **"But it is a pipe."**
>
> **No, it's not. It's a drawing of a pipe. Get it? All representations of a thing are inherently abstract. It's very clever.**
>
> JOHN GREEN

Abstraction is similar to the selective filtering function in our brains that gates the neural signals with which we are constantly bombarded so we can make sense of our world and focus on what's essential to us.



Examples of Abstraction in Everyday Life

Another way to think about abstraction is in the context of those big concepts that inform how we think about the world like Newton's Laws of Motion, the Law of Supply and Demand, or the Pythagorean Theorem. All of these required the people behind them to think about big, broad, and complex concepts; to break down the problem and to experiment; to find patterns amongst the experimentations; and to eventually abstract this concrete knowledge to package it into these sterile statements that shelter us from the complexity and difficulty waded through to arrive at this law.

Educators use abstraction when looking at vast sets of student data to focus on the most relevant numbers and trends. And educators also use it when helping a student complete an assignment. There may be kids running around the classroom or making loud noises, but they can tune that out to focus on what the kid in need is asking – until of course it reaches an apex level of rambunctiousness and an intervention must be had.

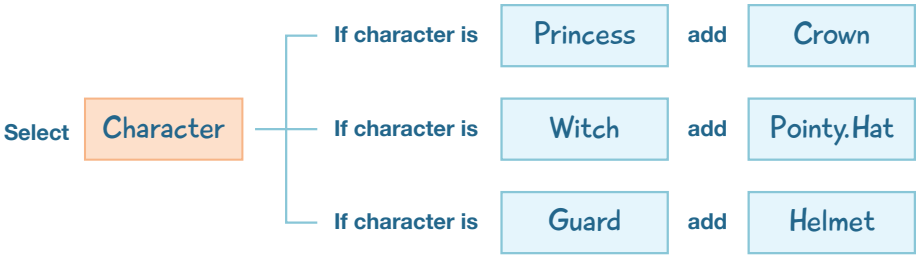
Examples of Abstraction in Curriculum

Like the other elements of computational thinking, abstraction occurs inherently and can be addressed throughout curriculum with students. Here are some ideas.

English Language Arts	Students summarize a novel into a book review.
Mathematics	Students conduct a survey of peers and analyze the data to note the key findings, create visualizations, and present the results.
Science	Students develop laws and theorems by looking at similar formulas and equations.
Social Studies	Students coalesce the most important details shared in articles about a current event and write a brief about the event.
Languages	Students create a personal guide that dictates when to use the formal and informal 'you' in Spanish class or the two 'to know' verbs in French, which, mind you, always confounded me.
Arts	Students generalize chord progressions for common musical genres into a set of general principles they can communicate.

Examples of Abstractions in Computer Science

Abstraction in coding is used to simplify strings of code into different functions. It hides the underlying complexity in a programming language, which makes it simpler to implement algorithms and communicate with digital tools.



Abstraction helps students return to the larger problem that prompted this whole computational thinking adventure and identify the most important details from the earlier phases.

Understanding abstraction enables students to make sense of problems they encounter, helping them to not be overwhelmed in the face of something complex and to persist, compute, iterate, and ideate.

Algorithmic Thinking

An algorithm is a process or formula for calculating answers, sorting data, and automating tasks; and algorithmic thinking is the process for developing an algorithm.

> **Effective algorithms make assumptions, show a bias toward simple solutions, trade off the costs of error against the cost of delay, and take chances.**
>
> **BRIAN CHRISTIAN & TOM GRIFFITHS**

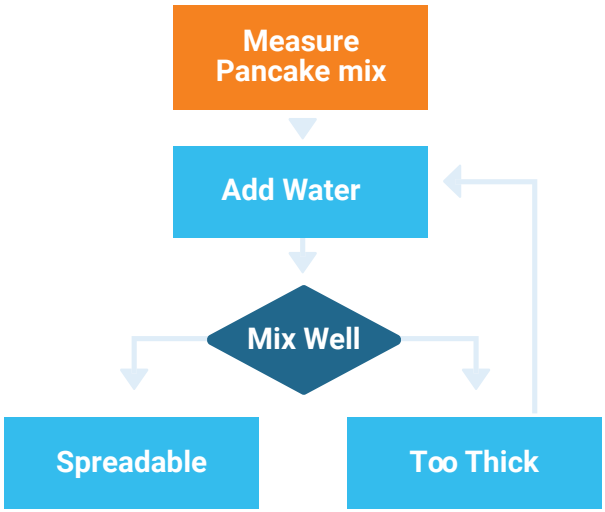
With algorithmic thinking, students endeavor to construct a step-by-step process for solving a problem so that the work is replicable by humans or computers. Algorithmic thinking is a derivative of computer science and the process to develop code and program applications. This approach automates the problem-solving process by creating a series of systematic, logical steps that intake a defined set of inputs and produce a defined set of outputs based on these.

In other words, algorithmic thinking is not solving for a specific answer; instead, it solves how to build a sequential, complete, and replicable process that has an end point – an algorithm. Designing an algorithm helps students to both communicate and interpret clear instructions for a predictable, reliable output. As was said earlier, this is the crux of computational thinking.

Examples of Algorithms in Everyday Life

And like computational thinking and its other elements we've discussed, algorithms are something we experience quite regularly in our lives.

If you're an amateur chef or a frozen meal aficionado, you follow recipes and directions for preparing food, and that's an algorithm.



When you're feeling groovy and bust out in a dance routine – maybe the Cha Cha Slide, the Macarena, or Flossing – you are also following a routine that emulates an algorithm while simultaneously being really cool.

Outlining a process for checking out books in a school library or instructions for cleaning up at the end of the day is developing an algorithm and letting your inner computer scientist shine.

Examples of Algorithms in Curriculum

Beginning to develop students' algorithmic prowess, however, does not require formal practice with coding or even access to technology. To get started, here are ideas for incorporating algorithmic thinking in different subjects.

English Language Arts	Students map a flow chart that details directions for determining whether to use a colon or dash in a sentence.
Mathematics	In a word problem, students develop a step-by-step process for how they answered the question that can then be applied to similar problems.
Science	Students articulate how to classify elements in the periodic table.
Social Studies	Students describe a sequence of smaller events in history that precipitated a much larger event.
Languages	Students apply new vocabulary and practice speaking skills to direct another student to perform a task, whether it's ordering coffee at a café or navigating from one point in a classroom to another.
Arts	Students create instructions for drawing a picture that another student then has to use to recreate the image.

Examples of Algorithms in Computer Science

These are obviously more elementary examples; algorithms – especially those used in coding – are often far more intricate and complex. To contextualize algorithms in computer science and programming, below are two examples.

Standardized Testing and Algorithms: Coding enables the adaptive technology often leveraged in classrooms today. For example, the shift to computer-based standardized tests has led to the advent of adaptive assessments that pick questions based on student ability as determined by correct and incorrect answers.

If students select the correct answer to a question, then the next question is moderately more difficult. But if they answer wrong, then the assessment offers a moderately easier question. This occurs through an iterative algorithm that starts with a pool of questions. After an answer, the pool is adjusted accordingly. This repeats continuously.

The Omnipotent Google and Algorithms: Google's search results are determined (in part) by the PageRank algorithm, which assigns a webpage's importance based on the number of sites linking to it.

So, if we google 'what is an algorithm,' we can bet that the chosen pages have some of the most links to them for the topic 'what is an algorithm.' It's still more complicated than this, of course; if you are interested, this article goes into the intricacies of the PageRank algorithm.

There are over 1.5 billion websites with billions more pages to count, but thanks to algorithmic thinking we can type just about anything into Google and expect to be delivered a curated list of resources in under a second. This right here is the power of algorithmic thinking.

"The Google algorithm was a significant development. I've had thank-you emails from people whose lives have been saved by information on a medical website or who have found the love of their life on a dating website."

TIM BERNERS-LEE

In whatever way it's approached in the classroom, algorithmic thinking encourages students to communicate clearly and logically. Students learn to persevere throughout its multiple iterations, challenges, and solutions. To arrive at an algorithm (especially as algorithms advance in complexity), they must apply computational thinking and practice metacognition as they do so. In this process, students become more adept critical thinkers, eloquent communicators, and curious problem solvers that ask bold questions and flourish in ambiguity and uncertainty.

The Whole Shebang: Four Computational Thinking Projects for Students

Now that we have explored the nuances of decomposition, pattern recognition, abstraction, and algorithmic thinking, this section will offer four project examples in math, English language arts, science, and social studies. These can all be easily modified to fit different grade levels, too.

Data Analysis in Math Class

In a middle school math class, students embarked on an inquiry-driven project in which they curated and collected data and analyzed it algebraically. They mapped the quantitative variables in scatter plots to identify trends and then used r-value representations to show their findings.

The teacher emphasized throughout this project that "*data is a tool to get people to hear you.*" With concise analysis and engaging visuals, students were able to create compelling projects on topics about which they were passionate.

One student analyzed instances of breast cancer by using an online database to curate data dating back to 1995. Having had her mother diagnosed with the disease as well as two family friends, she wanted to study the rates of mortality and measure the rate of increases in diagnoses. Looking at the data, *she applied computational thinking skills to find patterns and abstract the most important information*.

Understanding Character Connections in English Language Arts

Language arts classes are also opportunities to leverage computational thinking in the classroom. In this example, students used computational thinking skills to perform literary analysis on books like Hamlet and Harry Potter. *Students developed network diagrams and interaction graphs to abstract the different connections between characters.*

This technique helped *contextualize the literature* for students so they can better create understanding about the work, like power dynamics or important relationships that drive the narrative. This helped students to *build a more complete understanding of the readings* and track the flow of narratives on anything from the Cat in the Hat to Beowulf. As explained by the writer, this sort of analysis enabled students to understand the questions that data can answer and what data analysis can be automated.

Using Design Thinking to Build Models in Science

In this Science class, students applied computational thinking, physics, and engineering design to build earthquake resistant bridges. The unit started with understanding the function of bridges and the different types. Students then moved to studying earthquakes and the impact of their forces.

When it came time to design the earthquake resistant bridge, students used their computational thinking skills. Computational thinking enabled them to *analyze a variety of bridge models to find patterns in their structure and abstract from this the important elements needed in a functional design*.

As they tested the different prototypes, computational thinking allowed them to collect data and find opportunities to improve the structure.

This can be a great unplugged project that hones student *collaboration* and *critical thinking* through working to design functional models and also enhance their engineering design skills.

Decoding Cryptography in Social Studies

In studying the importance of cryptography for sending coded messages in World War II, specifically focusing on the German Enigma machine, students learned *how secret codes can be both developed and cracked using algorithms* and other aspects of computational thinking. In this example of computational thinking, students designed their own cipher wheel to send coded messages and learned how algorithms are integral to developing coded languages.

While this example is great to engage students in a unit about World War II with collaborative, hands-on practice, it can also be translated into similar projects for other coded languages used in combat, Morse code, or language in general. *Language, in essence, is a series of patterns from which we can abstract different rules*, making it an excellent way to engage students in computational thinking in real-world contexts. Beyond growing their understanding of code in World War II in this unit, students also deepen their understanding of language and ability to recognize patterns that exist all around us.

More Than Just Problem Solving

Computational thinking is a shift in how we approach problem solving. With a formulaic process, we can navigate complexity and stay focused on what is important without losing site of the solution amongst all the noise. With it, we can solve problems with mass amounts of data and lead unknown journeys through these data-filled landscapes. This ability to navigate complex information and to think in a way that complements technological processes is essential to student readiness.

Beyond critical thinking and other problem-solving skills, computational thinking builds essential attitudes like:

- Embracing ambiguity with confidence.
- Persisting through iteration and experimentation.
- Practicing teamwork.
- Leading learning with inquiry.
- Situating oneself as a lifelong learner.

Through this process, students learn to ask bold questions and persist through complexities toward yet-to-be imagined solutions. In applying computational thinking, students collect and analyze resources, think critically and creatively in collaborative environments, and develop a growth mindset by learning to embrace ambiguity and reframe challenges as opportunities, both with and without technology.

These abilities empower students to be intentional and mindful in their thoughts, their actions, and their connections they build.

What's the takeaway?

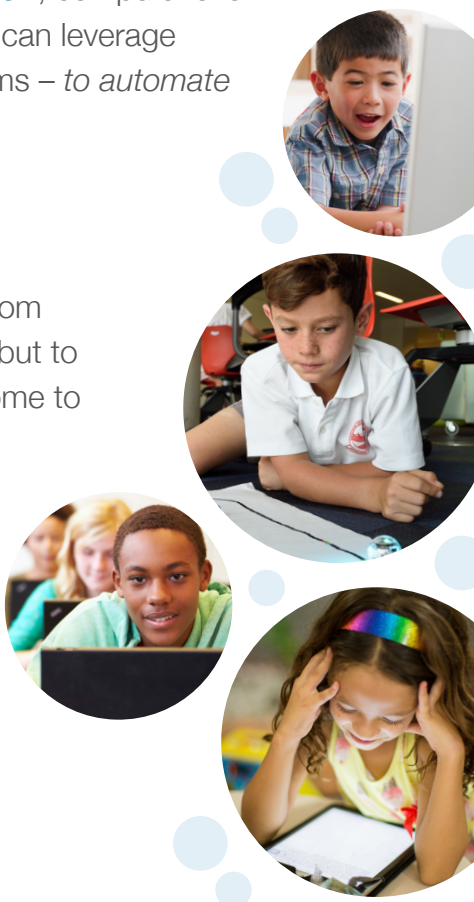
Computational thinking is more than just problem solving.

As a foundation for coding, computational thinking encourages us to reflect clearly on a problem we're solving and intentionally program solutions for it.

As a foundation for technology integration, computational thinking encourages us to consider how we can leverage technology to aid us in solving these problems – *to automate certain tasks.*

As a foundation for problem solving, computational thinking encourages us to be diligent and organized in our work, to plan from the outset how we want to solve a problem but to embrace the fluidity of the process as we come to more and more understanding of the data and information we're navigating.

As a foundation for thought, computational thinking encourages us to push the bounds of our creativity, to imagine impossible solutions and strive to make these possible, and to think according to potential and not limits.



To learn more visit **Learning.com** or
contact us at **hello@learning.com**.