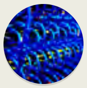
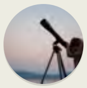
What is Computer Science?

Computer Science (CS) is not programming or coding



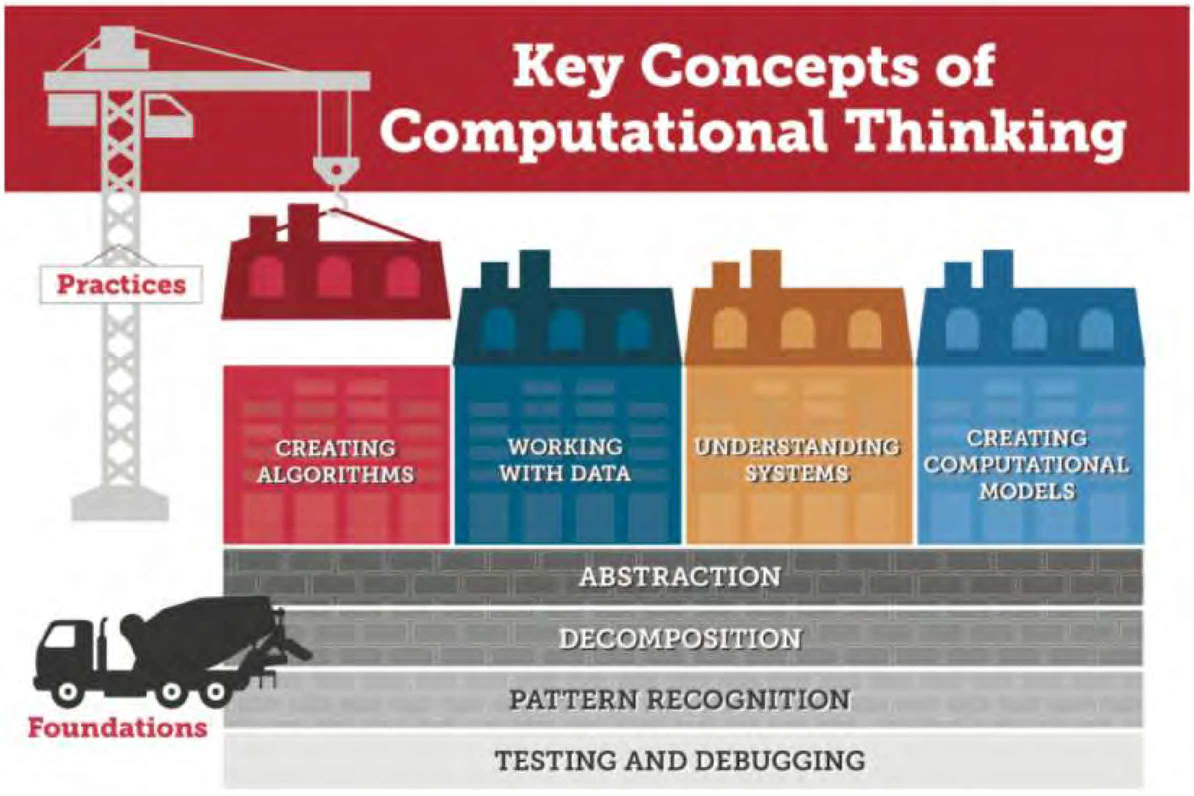
“Computer science, at its core, is about storing, accessing, transforming, transmitting, and interacting with information. ”

- Thomas Corman



“Computer Science is no more about computers than astronomy is about telescopes. ”

- Edsger Dijkstra



Digital Promise

Computational Thinking

Computational thinking refers to the thought processes involved in expressing solutions as computational steps or algorithms that can be carried out by a computer. (Cuny, Snyder, & Wing, 2010; Aho, 2011; Lee, 2016).

Operational Definition of Computational Thinking

Computational thinking (CT) is a problem-solving process that includes (but is not limited to) the following characteristics:

- Formulating problems in a way that enables us to use a computer and other tools to help solve them.
- Logically organizing and analyzing data
- Representing data through abstractions such as models and simulations
- Automating solutions through algorithmic thinking (a series of ordered steps)
- Identifying, analyzing, and implementing possible solutions with the goal of achieving the most efficient and effective combination of steps and resources
- Generalizing and transferring this problem solving process to a wide variety of problems

Skills are supported and enhanced by a number of dispositions or attitudes that are essential dimensions of CT.

- Confidence in dealing with complexity
- Persistence in working with difficult problems
- Tolerance for ambiguity
- The ability to deal with open ended problems
- The ability to communicate and work with others to achieve a common goal or solution
    - Refer to the attached pdf below for more information.

computational-thinking-operational-definition-flyer.pdf
174.4 KB

Differences Between Computational Thinking & Computer Science

It is important to note that computational thinking overlaps, and yet is distinct from, computer science (the study of computers and algorithmic processes, including their principles, hardware and software designs, and impact on society) and coding (the practice of developing a set of instructions that a computer can understand and execute).

Digital Promise

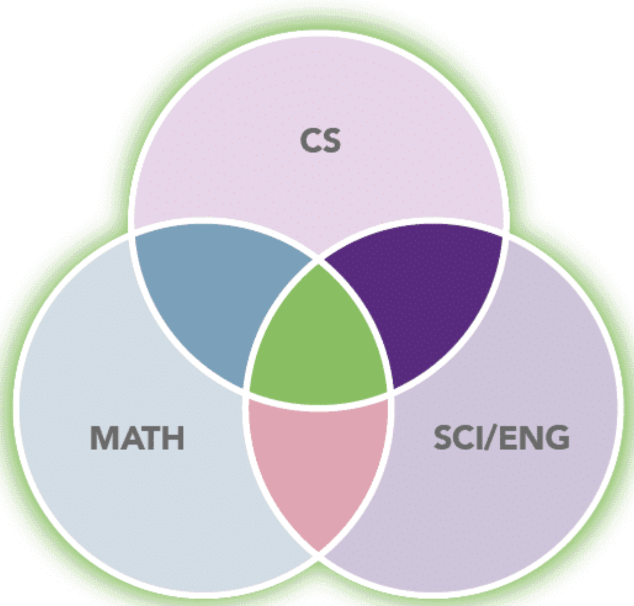
What is Computational Thinking? **external link**

Concepts at the Intersection of Computer Science and Computational Thinking

- Decomposition
- Pattern Recognition
- Abstraction
- Algorithm Design
- Testing and Debugging

- Source: [Early Learning Strategies for Developing Computational Thinking Skills]

RELATIONSHIPS BETWEEN COMPUTER SCIENCE, SCIENCE AND ENGINEERING, AND MATH PRACTICES



CS + Math

- **Develop and use abstractions**
  M2. Reason abstractly and quantitatively
  M7. Look for and make use of structure
  M8. Look for and express regularity in repeated reasoning
  CS4. Developing and Using Abstractions
- **Use tools when collaborating**
  M5. Use appropriate tools strategically
  CS2. Collaborating Around Computing
- **Communicate precisely**
  M6. Attend to precision
  CS7. Communicating About Computing

CS + Sci/Eng

- **Communicate with data**
  S4. Analyze and interpret data
  CS7. Communicating About Computing
- **Create artifacts**
  S3. Plan and carry out investigations
  S6. Construct explanations and design solutions
  CS4. Developing and Using Abstractions
  CS5. Creating Computational Artifacts
  CS6. Testing and Refining Computational Artifacts

CS + Math + Sci/Eng

- **Model**
  S2. Develop and use models
  M4. Model with mathematics
  CS4. Developing and Using Abstractions
  CS6. Testing and Refining Computational Artifacts
- **Use computational thinking**
  S5. Use mathematics and computational thinking
  CS3. Recognizing and Defining Computational Problems
  CS4. Developing and Using Abstractions
  CS5. Creating Computational Artifacts
- **Define problems**
  S1. Ask questions and define problems
  M1. Make sense of problems and persevere in solving them
  CS3. Recognizing and Defining Computational Problems
- **Communicate rationale**
  S7. Engage in argument from evidence
  S8. Obtain, evaluate, and communicate information
  M3. Construct viable arguments and critique the reasoning of others
  CS7. Communicating About Computing

* Computer science practices also overlap with practices in other domains, including English language arts. For example, *CS1. Fostering an Inclusive Computing Culture* and *CS2. Collaborating Around Computing* overlap with *E7. Come to understand other perspectives and cultures through reading, listening, and collaborations.*

TESTING AND DEBUGGING ADDED BY K12 COMPUTER SCIENCE

Decomposition

Breaking a problem down into its constituent parts. 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. 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.

Source:

McVeigh-Murphy, A. (2019, September 25). The one about decomposition in computational thinking. equip. Retrieved July 19, 2022, from https://equip.learning.com/decomposition-computational-thinking/

Examples of Decomposition 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.

Pattern Recognition in CS and CT

equip.learning.com

What is patter recognition? In computer science, pattern recognition is the imposition of identity on input data, such as speech, images, or a stream of text, by the recognition and delineation of patterns it contains and their relationships. Stages in pattern recognition may involve measurement of the object to identify distinguishing attributes, extraction of features for the defining attributes, and comparison with known patterns to determine a match or mismatch. ~britannica.com

Specifically, with computational thinking, pattern recognition occurs as people study the different decomposed problems. 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.

The One About Pattern Recognition in Computational Thinking-external link

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, and to develop behavioral responses and cultivate connections in this world.

Beyond this, pattern recognition also occurs when scientists are trying 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 chatbot pesters you on a website, the technology (Artificial Intelligence and Machine Learning) rely on pattern recognition.

Examples of Pattern Recognition in Curriculum

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.

Abstraction

Abstraction is the process of filtering out – ignoring - the characteristics of patterns that we don't need in order to concentrate on those that we do. It is also the filtering out of specific details. From this we create a representation (idea) of what we are trying to solve.

Abstraction in computational thinking enables us to navigate complexity and find relevance and clarity at scale. This process occurs through filtering out the extraneous and irrelevant in order to identify what's most important and connects each decomposed problem. Abstraction is actually 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.

The One About Abstraction in Computational Thinking external link

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; and 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

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, present the findings.

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 specific 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.

"Spherical Chickens

A farmers has some chickens who don't lay any eggs. The farmer calls a physicist to help. The physicist does some calculation and says "I have a solution, but it only works for spherical chickens in a vacuum!

Why spherical chickens? This is an ABSTRACTION."

Big Bang Theory

Algorithmic Thinking

Algorithmic thinking is the use of algorithms, or step-by-step sets of instructions, to complete a task. Knowing what an algorithm is is very different from thinking algorithmically. It’s the difference between memorizing a formula and constructing your own, it’s understanding conditional statements and loops and using logic to problem-solve and create procedural writing. ~medium.com

Project Lead the Way-Algorithmic Thinking external link-YouTube

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.

With algorithmic thinking, students endeavor to construct a step-by-step process for solving a problem and like problems 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. **This is the crux of computational thinking.**

The One About Algorithmic Thinking in Computational Thinking

An algorithm is a precisely defined process for solving a problem, usually in a way that can be easily automated. Algorithmic thinking, or computational thinking, refers to thinking about these processes for solving problems. In schools, the study of algorithms includes learning about the ways that information and processes can be represented systematically, the common building blocks of algorithms, such as loops and conditional statements, and patterns in the design of algorithms. Many of the problem solving strategies that students learn in mathematics classes, such as 'Guess, Check and Improve' or 'Try a simpler problem' are similar to algorithm design strategies.

Greg Breese

Examples of Algorithms in Everyday Life

Like computational thinking and its other elements we've discussed, algorithms are something we experience 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 and 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. ~Anna McVeigh-Murphy

Examples of Algorithms in Curriculum

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 a 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.

Testing & Debugging

Testing, debugging, and evaluation is about testing and refining a potential solution, and ensuring it is the best fit for the problem. Testing and refinement is the deliberate and iterative process of improving a computational artifact. This process includes debugging (identifying and fixing errors) and comparing actual outcomes to intended outcomes. Students also respond to the changing needs and expectations of end users and improve the performance, reliability, usability, and accessibility of artifacts. ~Computer Science Framework

PDF

K–12-Computer-Science-Framework.pdf

9.1 MB

The Art of Debugging & How t...

Share

[The Art Of Debugging & How To Think Like A Computer Scientist]

Learning Outcomes for Testing and Refining Computational Artifacts

- Systematically test computational artifacts by considering all scenarios and using test cases.
- Identify and fix errors using a systematic process.
- Evaluate and refine a computational artifact multiple times to enhance its performance, reliability, usability, and accessibility.

Examples of Review and Testing from Everyday Life

- The task of a book editor involves very detailed review of manuscripts.
- Engineering plans for a new building are reviewed extensively throughout the process of design and construction.

- For a faculty member, publishing in a research journal often requires a long review process before publication.

Examples of Review and Testing in Curriculum

- **English:** for an assignment of writing an essay, students submit a paper draft. These drafts are reviewed by classmates.
- **Mathematics:** in an assignment to learn long division, students are permitted to check their work with a calculator.
- **Science:** in a Chemistry class, students perform a wet lab experiment and verify their results analytically.