Utah K-5 Computer Science Standards

2019

Utah State Board of Education
ADA Compliant: October 2019

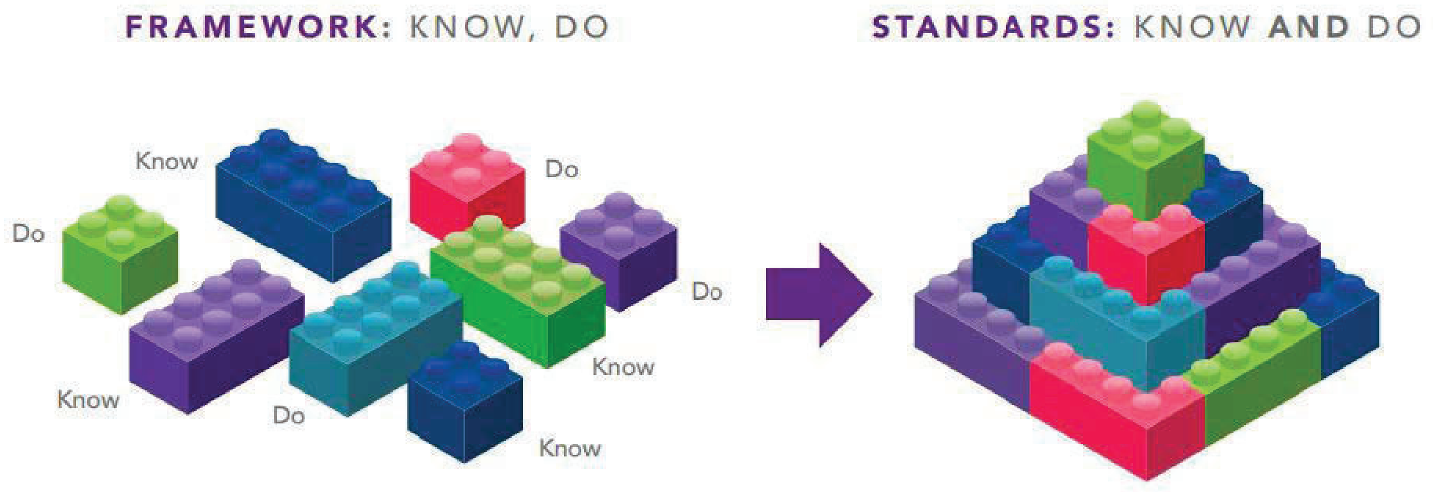
Introduction

The Utah State Board of Education (USBE) formed a Computer Science Taskforce (Taskforce) to establish a Vision of Computer Science in the Public Education System. The Taskforce met multiple times to identify a strategic plan of recommendations to successfully carry out computer science education within the K-12 schools.

In June 2018, the Taskforce's strategic priorities (steps) to accomplish the vision were presented to and subsequently accepted by the USBE. The priorities are as follows:

- Develop and implement statewide K-12 framework for computer science.
- Start early by engaging students at the elementary level.
- Develop a statewide strategy to communicate the value of computer science.
- Build capacity among educators at pre-service and in-service levels.
- Improve upon current course requirements to scaffold computer science learning K-12.
- Regardless of location, ensure students can access a majority of the 30+ computer science and IT courses currently offered (popular courses include Computer Programming 1, Computer Science Principles, Exploring Computer Science, Web Development 1, and Game Development).

The first strategic priority, *Utah Computer Science K-12 Framework*[1], has been developed and approved by the Board. Implementation of the framework is the next step within the strategic priority.



Utah teachers, principals, district leaders, and university professors worked with the USBE in March 2019 to develop K-5 Computer Science Standards. The writers used the Utah Computer Science K-12 Framework[2] and the K-12 Computer Science Framework[3] to identify important concepts and practices to inform the creation of standards for each grade level.

[1] *Utah Computer Science K-12 Framework. (October 2018). Retrieved from:* https://schools.utah.gov/file/46d4ca37-9d23-414e-91fd-6640b6be9df6

2 Utah Computer Science K-12 Framework. (October 2018). Retrieved from: https://schools.utah.gov/file/46d4ca37-9d23-414e-91fd-6640b6be9df6

[3] *K-12 Computer Science Framework. (October 2016) Retrieved from:*

https://k12cs.org/wpcontent/uploads/2016/09/K%E2%80%9312-Computer-Science-Framework.pdf

Each student in secondary public schools will have access to robust and varied computer science courses by 2022. All students will enter secondary schools with exposure to computational thinking and competencies in digital literacy. This begins in our elementary schools with competencies in keyboarding, appropriate and responsible use of technology, and basic coding principles

Utah Computer Science Vision Statement

Organization of Standards:

The Utah K-5 Computer Science standards are organized into strands, which represent significant areas of learning within content areas. Within each strand are standards. A standard is an articulation of the demonstrated proficiency to be obtained. A standard represents an essential element of the learning that is expected. While some standards within a strand may be more comprehensive than others, all standards are essential for mastery

Within the standards there are words that are bold, underlined, and in green text. For example, look at Standard 2.DA.1.

Standard 2.DA.1 - Demonstrate how to **store**, copy, search, retrieve, modify and delete information using a **computing device**, and define theinformation stored as data.

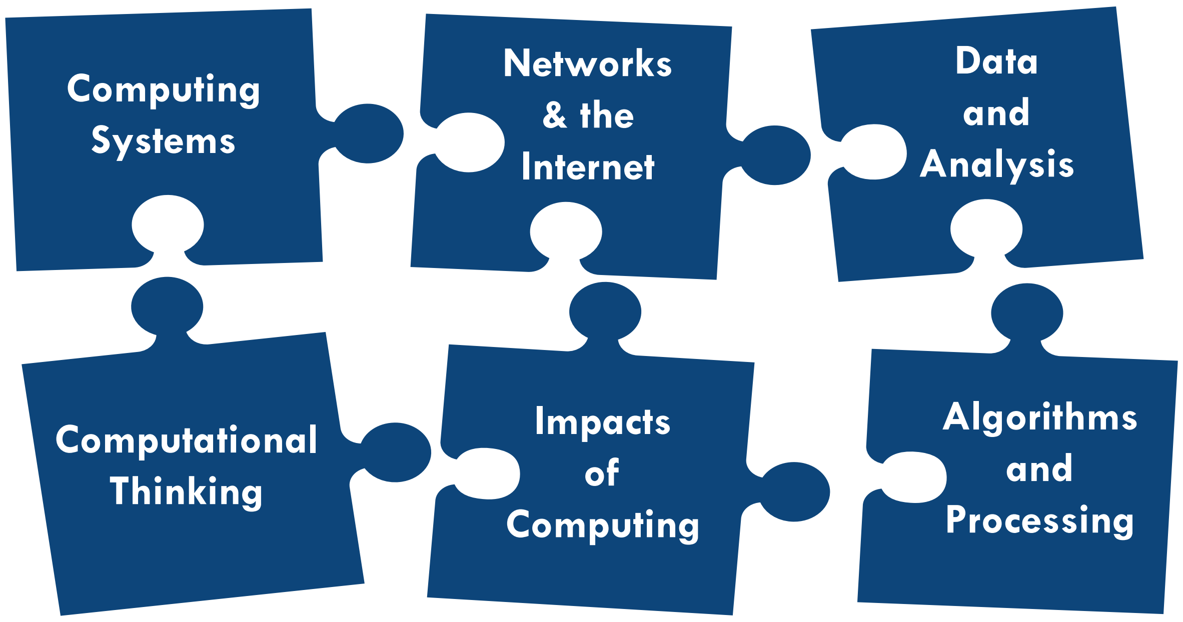
(Practice 4: Developing and UsingAbstractions)

Green text demonstrates an alignment to the practice language that is highlighted at the conclusion of each standard.

The **bold and underlined text**, such as **computing device**, are included in the standards glossary at the conclusion of the document.

Strand Language[4]:

Core Concepts



Computing Systems (CS):

People interact with a wide variety of computing devices that collect, store, analyze, and act upon information in ways that can affect human capabilities, both positively and negatively. The physical components (hardware) and instructions (software) that make up a computing system communicate and process information in digital form. An understanding of hardware and software is useful when troubleshooting a computing system that does not work as intended.

Network and the Internet (NI):

Computing devices typically do not operate in isolation. Networks connect computing devices to share information and resources and are an increasingly integral part of computing. Networks and communication systems provide greater connectivity in the computing world by providing fast, secure communication and facilitating innovation.

[4] *K-12 Computer Science Framework. (October 2016) Retrieved from: https://k12cs.org/wpcontent/uploads/2016/09/K%E2%80%9312-Computer-Science-Framework.pdf*

Data and Analysis (DA):

Computing systems exist to process data. The amount of digital data generated in the world is rapidly expanding, and the need to process data effectively is increasingly important. Data is collected and stored so it can be analyzed to better understand the world and make more accurate predictions.

Algorithms and Programming (AP):

An algorithm is a sequence of steps designed to accomplish a specific task. Algorithms are translated into programs, or code, to provide instructions for computing devices. Algorithms and programming control all computing systems, empowering people to communicate with the world in new ways and solve compelling problems. The development process to create meaningful and efficient programs involves choosing which information to use and how to process and store it, breaking apart large problems into smaller ones, recombining existing solutions, and analyzing different solutions.

Impacts of Computing (IC):

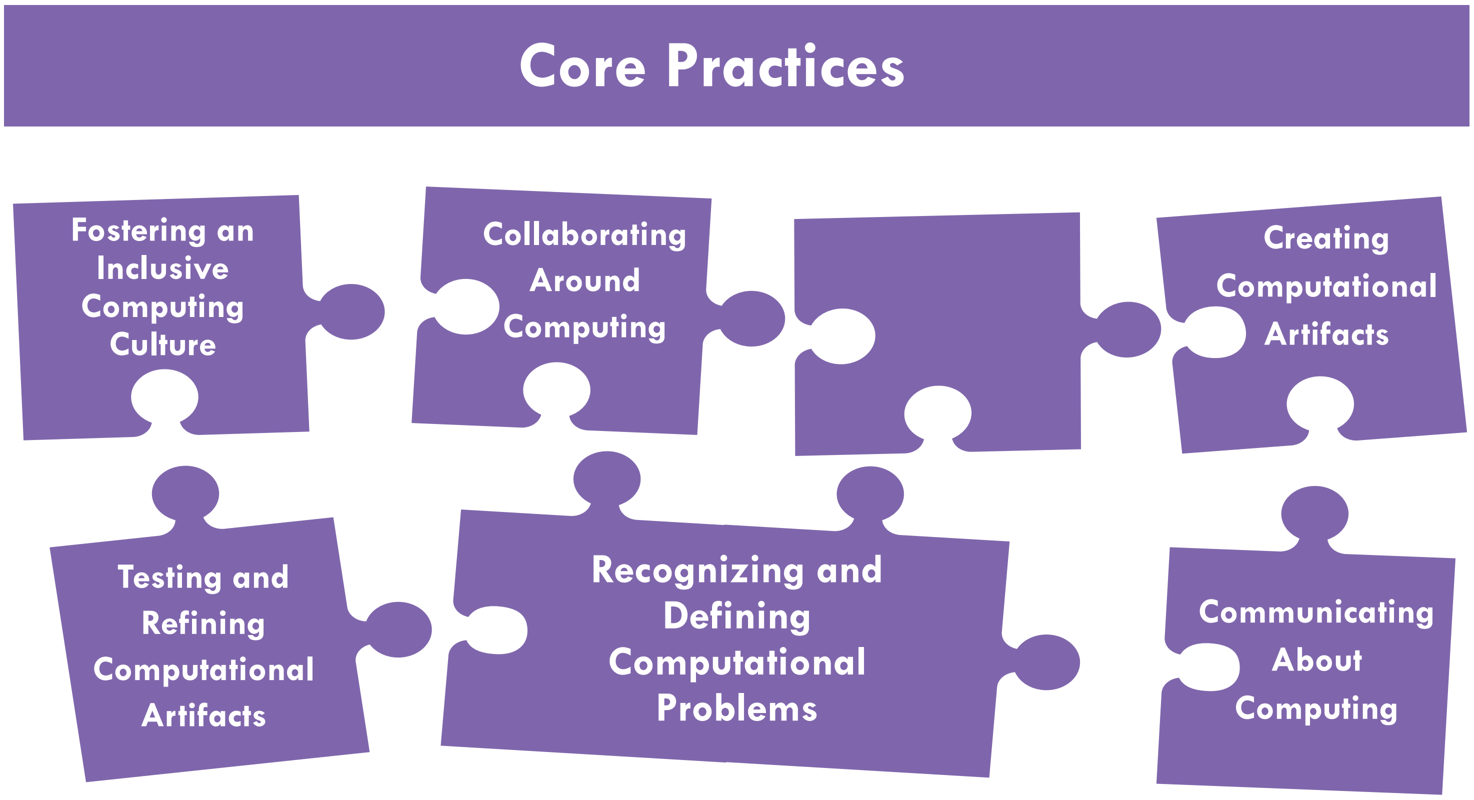
Computing affects many aspects of the world in both positive and negative ways at local, national, and global levels. Individuals and communities influence computing through their behaviors and cultural and social interactions, and in turn, computing influences new cultural practices. An informed and responsible person should understand the social implications of the digital world, including equity and access to computing.

Computational Thinking (CT):

Computational thinking (CT) is a problem-solving process that includes several characteristics, such as logically ordering and analyzing data and creating solutions using a
series of ordered steps (or algorithms), and dispositions, such as the ability to confidently deal with complexity and open-ended problems. CT is essential to the development of computer applications, but it can also be used to support problem-solving across all disciplines, including math, science, and the humanities. Students who learn CT across the curriculum can begin to see a relationship between subjects as well as between school and life outside of the classroom[5].

[5] *Exploring Computer Science - Google Resources. Retrieved from: https://edu.google.com/resources/programs/exploring-computational-thinking/*

Practice Language[6]:


Core Practices
Fostering an Inclusive Computing Culture
Collaborating Around Computing
Creating Computational Artifacts
Testing and Refining Computational Artifacts
Recognizing and Defining Computational Problems
Communicating About Computing

Practice 1: Fostering an Inclusive Computing Culture

Building an inclusive and diverse computing culture requires strategies for incorporating perspectives from people of different genders, ethnicities, and abilities. Incorporating these perspectives involves understanding the personal, ethical, social, economic, and cultural contexts in which people operate. Considering the needs of diverse users during the design process is essential to producing inclusive computational products.

By the end of Grade 5, students should be able to:

1. Include the unique perspectives of others and reflect on one's own perspectives when designing and developing computational products.

At all grade levels, students should recognize that the choices people make when they create artifacts are based on personal interests, experiences, and needs. Young learners should begin to differentiate their technology preferences from the technology preferences of others. Initially, students should be presented with perspectives from people with different backgrounds, ability levels, and points of view. As students progress, they should independently seek diverse perspectives throughout the design process for the purpose of improving their computational artifacts.

[6] *K-12 Computer Science Framework. (October 2016) Retrieved from: https://k12cs.org/wp-content/uploads/2016/09/K%E2%80%9312-Computer-Science-Framework.pdf*

2. Address the needs of diverse end users during the design process to produce artifacts with broad accessibility and usability.

At any level, students should recognize that users of technology have different needs and preferenc¬es and that not everyone chooses to use, or is able to use, the same technology products. For example, young learners, with teacher guidance, might compare a touchpad and a mouse to exam¬ine differences in usability. As students progress, they should consider the preferences of people who might use their products. Students should be able to evaluate the accessibility of a product to a broad group of end users, such as people with various disabilities. For example, they may notice that allowing an end user to change font sizes and colors will make an interface usable for people with low vision.

3. Employ self- and peer-advocacy to address bias in interactions, product design, and development methods.

After students have experience identifying diverse perspectives and including unique perspectives (P1.1), they should begin to employ self-advocacy strategies, such as speaking for themselves if their needs are not met.

Practice 2: Collaborating Around Computing

Collaborative computing is the process of performing a computational task by working in pairs and on teams. Because it involves asking for the contributions and feedback of others, effective collaboration can lead to better outcomes than working independently. Collaboration requires individuals to navigate and incorporate diverse perspectives, conflicting ideas, disparate skills, and distinct personalities. Students should use collaborative tools to effectively work together and to create complex artifacts.

By the end of Grade 5, students should be able to:

1. Cultivate working relationships with individuals possessing diverse perspectives, skills, and personalities.

At any grade level, students should work collaboratively with others. Early on, they should learn strategies for working with team members who possess varying individual strengths. For example, with teacher support, students should begin to give each team member opportunities to contribute and to work with each other as co-learners. Eventually, students should become more sophisti¬cated at applying strategies for mutual encouragement and support. They should express their ideas with logical reasoning and find ways to reconcile differences cooperatively. For example, when they disagree, they should ask others to explain their reasoning and work together to make respectful, mutual decisions.

2. Create team norms, expectations, and equitable workloads to increase efficiency and effectiveness.

After students have had experience cultivating working relationships within teams (P2.1), they should gain experi-ence working in particular team roles. Early on, teachers may help guide this process by providing collaborative structures. For example, students may take turns in different roles on the project, such as note taker, facilitator, or "driver" of the computer. As students progress, they should become less dependent on the teacher assigning roles and become more adept at assigning roles within their teams. For example, they should decide together how to take turns in different roles.

3. Solicit and incorporate feedback from, and provide constructive feedback to, team members and other stakeholders.

At any level, students should ask questions of others and listen to their opinions. Early on, with teacher scaffolding, students should seek help and share ideas to achieve a particular purpose. As they progress in school, students should provide and receive feedback related to computing in constructive ways. For example, pair programming is a collaborative process that promotes giving and receiving feedback.

4. Evaluate and select technological tools that can be used to collaborate on a project.

At any level, students should be able to use tools and methods for collaboration on a project. For example, in the early grades, students could collaboratively brainstorm by writing on a whiteboard. As students progress, they should use technological collaboration tools to manage team¬work, such as knowledge-sharing tools and online project spaces. They should also begin to make decisions about which tools would be best to use and when to use them.

Practice 3: Recognizing and Defining Computational Problems

The ability to recognize appropriate and worthwhile opportunities to apply computation is a skill that develops over time and is central to computing. Solving a problem with a computational approach requires defining the problem, breaking it down into parts, and evaluating each part to determine whether a computational solution is appropriate.

By the end of Grade 5, students should be able to:

1. Identify complex, interdisciplinary, real-world problems that can be solved computationally.

At any level, students should be able to identify problems that have been solved computationally. For example, young students can discuss a technology that has changed the world, such as email or mobile phones. As they progress, they should ask clarifying questions to understand whether a problem or part of a problem can be solved using a computational approach. For example, before attempting to write an algorithm to sort a large list of names, students may ask questions about how the names are entered and what type of sorting is desired.

2. Decompose complex real-world problems into manageable subproblems that could integrate existing solutions or procedures.

At any grade level, students should be able to break problems down into their component parts. In the early grade levels, students should focus on breaking down simple problems. For example, in a visual programming environment, students could break down (or decompose) the steps needed to draw a shape. As students progress, they should decompose larger problems into manageable smaller problems. For example, young students may think of an animation as multiple scenes and thus create each scene independently. Students can also break down a program into sub goals: getting input from the user, processing the data, and displaying the result to the user.

3. Evaluate whether it is appropriate and feasible to solve a problem computationally.

After students have had some experience breaking problems down (P3.2) and identifying sub-problems that can be solved computationally (P3.1), they should begin to evaluate whether a computational solution is the most appropriate solution for a problem.

Practice 4: Developing and Using Abstractions

Abstractions are formed by identifying patterns and extracting common features from specific examples to create generalizations. Using generalized solutions and parts of solutions designed for broad reuse simplifies the development process by managing complexity.

By the end of Grade 5, students should be able to:

1. Extract common features from a set of interrelated processes or complex phenomena.

Students at all grade levels should be able to recognize patterns. Young learners should be able to identify and describe repeated sequences in data or code through analogy to visual patterns or physical sequences of objects. As they progress, students should identify patterns as opportunities for abstraction, such as recognizing repeated patterns of code that could be more efficiently implemented as a loop.

2. Model phenomena and processes and simulate systems to understand and evaluate potential outcomes.

Students at all grade levels should be able to represent patterns, processes, or phenomena. With guidance, young students can draw pictures to describe a simple pattern, such as sunrise and sunset, or show the stages in a process, such as brushing your teeth. They can also create an animation to model a phenomenon, such as evaporation. As they progress, students should understand that computers can model real-world phenomena, and they should use existing computer simulations to learn about real-world systems. For example, they may use a preprogrammed model to explore how parameters affect a system, such as how rapidly a disease spreads.

Practice 5: Creating Computational Artifacts

The process of developing computational artifacts embraces both creative expression and the exploration of ideas to create prototypes and solve computational problems. Students create artifacts that are personally relevant or beneficial to their community and beyond. Computational artifacts can be created by combining and modifying existing artifacts or by developing new artifacts. Examples of computational artifacts include programs, simulations, visualizations, digital animations, robotic systems, and apps.

By the end of Grade 5, students should be able to:

1. Plan the development of a computational artifact using an iterative process that includes reflection on and modification of the plan, considering key features, time and resource constraints, and user expectations.

At any grade level, students should participate in project planning and the creation of brainstorming documents. The youngest students may do so with the help of teachers. With scaffolding, students should gain greater independence and sophistication in the planning, design, and evaluation of artifacts.

2. Create a computational artifact for practical intent, personal expression, or to address a societal issue.

Students at all grade levels should develop artifacts in response to a task or a computational problem. At the earliest grade levels, students should be able to choose from a set of given commands to create simple animated stories or solve pre-existing problems. Younger students should focus on artifacts of personal importance. As they progress, student expressions should become more complex and of increasingly broader significance.

3. Modify an existing artifact to improve or customize it.

At all grade levels, students should be able to examine existing artifacts to understand what they do. As they progress, students should attempt to use existing solutions to accomplish a desired goal. For example, students could attach a programmable light sensor to a physical artifact they have created to make it respond to light.

Practice 6: Testing and Refining Computational Artifacts

Testing and refinement are 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.

By the end of Grade 5, students should be able to:

1. Systematically test computational artifacts by considering all scenarios and using test cases.

At any grade level, students should be able to compare results to intended outcomes. Young students should verify whether given criteria and constraints have been met. As students progress, they should test computational artifacts by considering potential errors, such as what will happen if a user enters invalid input.

2. Identify and fix errors using a systematic process.

At any grade level, students should be able to identify and fix errors in programs (debugging) and use strategies to solve problems with computing systems (troubleshooting). Young students could use trial and error to fix simple errors. For example, a student may try reordering the sequence of commands in a program. In a hardware context, students could try to fix a device by resetting it or checking whether it is connected to a network. As students progress, they should become more adept at debugging programs and begin to consider logic errors: cases in which a program works, but not as desired. In this way, students will examine and correct their own thinking. For example, they might step through their program, line by line, to identify a loop that does not terminate as expected.

3. Evaluate and refine a computational artifact multiple times to enhance its performance, reliability, usability, and accessibility.

After students have gained experience testing (P6.2), debugging, and revising (P6.1), they should begin to evaluate and refine their computational artifacts. As students progress, the process of evaluation and refinement should focus on improving performance and reliability. For example, students could observe a robot in a variety of lighting conditions to determine that a light sensor should be less sensitive.

Practice 7: Communicating About Computing

Communication involves personal expression and exchanging ideas with others. In computer science, students communicate with diverse audiences about the use and effects of computation and the appropriateness of computational choices. Students write clear comments, document their work, and communicate their ideas through multiple forms of media. Clear communication includes using precise language and carefully considering possible audiences.

By the end of Grade 5, students should be able to:

1. *Select, organize, and interpret large data sets from multiple sources to support a claim.*

At any grade level, students should be able to refer to data when communicating an idea. Early on, students should, with guidance, present basic data using visual representations, such as storyboards, flowcharts, and graphs. As students progress, they should work with larger data sets and organize the data in those larger sets to make interpreting and communicating it to others easier, such as through the creation of basic data representations.

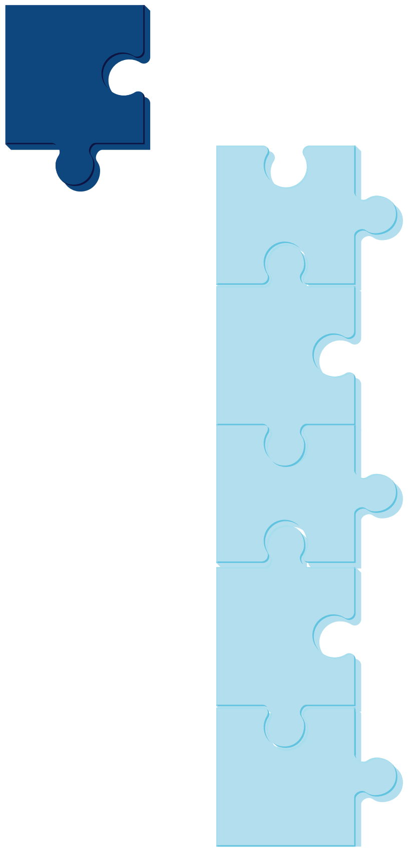
2. *Describe, justify, and document computational processes and solutions using appropriate terminology consistent with the intended audience and purpose.*

At any grade level, students should be able to talk about choices they make while designing a computational artifact. Early on, they should use language that articulates what they are doing and identifies devices and concepts they are using with correct terminology (e.g., program, input, and debug). Younger students should identify the goals and expected outcomes of their solutions. Along the way, students should provide documentation for end users that explains their artifacts and how they function, and they should both give and receive feedback. For example, students could provide a project overview and ask for input from users. As students progress, they should incorporate clear comments in their product and document their process using text, graphics, presentations, and demonstrations.

3. *Articulate ideas responsibly by observing intellectual property rights and giving appropriate attribution.*

All students should be able to explain the concepts of ownership and sharing. Early on, students should apply these concepts to computational ideas and creations. They should identify instances of remixing, when ideas are borrowed and iterated upon, and give proper attribution. They should also recognize the contributions of collaborators.

Kindergarten



Kindergarten

Computing Systems (CS):

People interact with a wide variety of computing devices that collect, store, analyze, and act upon information in ways that can affect human capabilities, both positively and negatively. The physical components (hardware) and instructions (software) that make up a computing system communicate and process information in digital form. An understanding of hardware and software is useful when troubleshooting a computing system that does not work as intended.

Standard K.CS.1 - Select **computing devices** that perform a variety of tasks accurately and quickly based on user needs and preferences.

(Practice 1: Fostering an Inclusive Computing Culture)

Students will select computing devices (phone, tablet, computer, cameras, software, 3D printers, etc.) and understand that they can be used to aid in a task (email, text message, voice calls, videos, digital imaging, modeling, etc.).

Network and the Internet (NI):

Computing devices typically do not operate in isolation. Networks connect computing devices to share information and resources and are an increasingly integral part of computing. Networks and communication systems provide greater connectivity in the computing world by providing fast, secure communication and facilitating innovation.

Standard K.NI.1 - Model and describe how people connect to other people and information through a **network**.

(Practice 4: Developing and Using Abstractions)

Students will be able to model and describe how information is sent and retrieved using a network to share information as a class. For example, demonstrating through the game of "telephone" (asking students to pass a message from one person to another). This can include sending words, images, etc. Students should demonstrate their understanding of this flow of information by drawing a model of the way packets are transmitted, programming an animation to show how packets are transmitted, or demonstrating this through an unplugged activity which has them act it out.

Standard K.NI.2 - Create patterns to communicate a message.

(Practice 4: Developing and Using Abstractions)

Students will use digital devices to create patterns with pictures, objects or words to communicate. Examples may include basic coding for simple directions with arrows, manipulatives, simple directions, etc. These basic coding examples can be classroom routines, such as what to do when you enter the classroom each day.

Data and Analysis (DA):

Computing systems exist to process data. The amount of digital data generated in the world is rapidly expanding, and the need to process data effectively is increasingly important. Data is collected and stored so it can be analyzed to better understand the world and make more accurate predictions.

Standard K.DA.1 - Identify and describe patterns in **data** visualizations, such as charts or graphs, to make predictions

(Practice 4: Developing and Using Abstractions)

Students will show data in a pattern, which could include images of favorite fruit, sport, or cookie. Students will show what would be next in a pattern, or what might be missing from a pattern. This could be color, number, animal, or letter pattern. Teachers can use digital tools to model data visualizations for students, this could be done with an interactive board, tablets, or computers.

Algorithms and Programming (AP):

An algorithm is a sequence of steps designed to accomplish a specific task. Algorithms are translated into programs, or code, to provide instructions for computing devices. Algorithms and programming control all computing systems, empowering people to communicate with the world in new ways and solve compelling problems. The development process to create meaningful and efficient programs involves choosing which information to use and how to process and store it, breaking apart large problems into smaller ones, recombining existing solutions, and analyzing different solutions.

Standard K.AP.1 - Model processes by creating and following **algorithms** to complete tasks.

(Practice 3: Recognizing and Defining Computational Problems and Practice 4: Developing and Using Abstractions)

Students will complete a familiar process or activity by creating algorithms to follow specific instructions. Emphasize real life examples, ordering, and sequencing such as brushing teeth, lining up to go to lunch, school safety drills. Expose students to algorithms as sequencing events.

Computational Thinking (CT):

Computational thinking (CT) is a problem-solving process that includes several characteristics, such as logically ordering and analyzing data and creating solutions using a series of ordered steps (or algorithms), and dispositions, such as the ability to confidently deal with complexity and open-ended problems. CT is essential to the development of computer applications, but it can also be used to support problem-solving across all disciplines, including math, science, and the humanities. Students who learn CT across the curriculum can begin to see a relationship between subjects as well as between school and life outside of the classroom.[7]

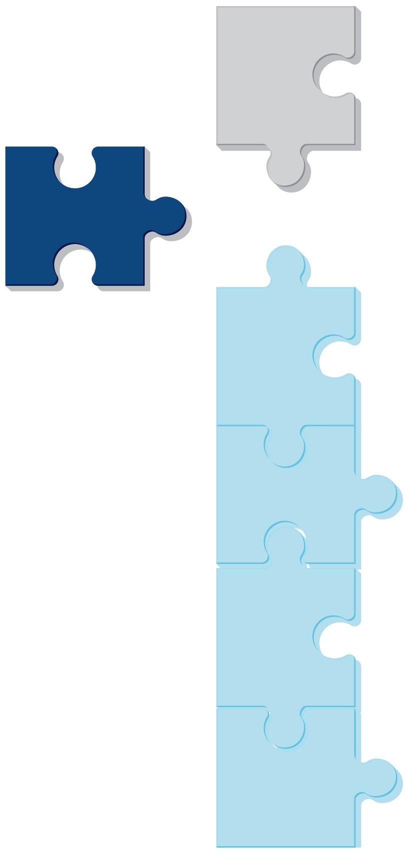
Standard K.CT.1 - Decompose problems into smaller manageable parts to better understand them.

(Practice 3: Recognizing and Defining Computational Problems)

Students will be able to take a complex problem and break it down into smaller components. Examples may include breaking down the steps needed to make breakfast, get ready for school, or to code a character across the screen. Teachers may use digital tools to diagram components of a task such as drawing a shape.

[7] *Exploring Computer Science - Google Resources. Retrieved from: https://edu.google.com/resources/programs/exploring-computational-thinking/*

1st Grade



First Grade

Computing Systems (CS):

People interact with a wide variety of computing devices that collect, store, analyze, and act upon information in ways that can affect human capabilities, both positively and negatively. The physical components (hardware) and instructions (software) that make up a computing system communicate and process information in digital form. An understanding of hardware and software is useful when troubleshooting a computing system that does not work as intended.

Standard 1.CS.1 - Operate a variety of **computing devices** that perform tasks accurately and quickly based on user needs and preferences.

(Practice 1: Fostering an Inclusive Computing Culture)

Students will perform a variety of tasks by operating digital devices (laptop, tablet, desktop, etc.) based on availability and the task they are seeking to accomplish. For example, if students are asked to draw a picture, they should be able to open and use a drawing app/program to complete this task.

Standard 1.CS.2 - Explore the functions of common **hardware** and **software** components of **computing systems**.

(Practice 6: Testing and Refining Computational Artifacts and Practice 7: Communicating About Computing)

Students will explore and identify common hardware and software components (mouse, keyboard, storage, trackpad, tablet devices, laptop, monitor, application (app), input, output, etc.) as well as their function.

Data and Analysis (DA):

Computing systems exist to process data. The amount of digital data generated in the world is rapidly expanding, and the need to process data effectively is increasingly important. Data is collected and stored so it can be analyzed to better understand the world and make more accurate predictions.

Standard 1.DA.1 - Collect and present **data** in various visual formats.

(Practice 4: Developing and Using Abstractions and Practice 7: Communicating About Computing)

Students will create surveys of things that interest them (such as favorite foods, colors, books, etc.), collect answers, and decide how to present the data in various visual formats (tally marks, color blocks stacked, sticky notes, etc.). Teachers can use digital tools to model data visualizations for students.

Data and Analysis (DA): *continued*

Standard 1.DA.2 - Identify and describe patterns in **data** visualizations (unplugged or digital), such as charts or graphs, to make predictions.

(Practice 4: Developing and Using Abstractions)

Students will identify and describe patterns to create hypotheses based on data visualizations (charts, graphs, etc.). Examples include drawing a picture graph and a bar graph with single-unit scale using a drawing app/program to represent a data set with up to four categories or using a chart to sort and predict colors of M&M's in a bag. This focus is making predictions based on data.

Algorithms and Programming (AP):

An algorithm is a sequence of steps designed to accomplish a specific task. Algorithms are translated into programs, or code, to provide instructions for computing devices. Algorithms and programming control all computing systems, empowering people to communicate with the world in new ways and solve compelling problems. The development process to create meaningful and efficient programs involves choosing which information to use and how to process and store it, breaking apart large problems into smaller ones, recombining existing solutions, and analyzing different solutions.

Standard 1.AP.1 - Demonstrate understanding of the way **programs store** and manipulate **data** as **variables**, such as numbers, words, colors, and images.

(Practice 4: Developing and Using Abstractions)

Students will demonstrate understanding of how computers store data (input) in a variety of ways (variables) that a program can use. For example, a number variable can only store a number and not letters. An alphanumeric variable can store numbers or letters but cannot be added or subtracted because it is text. Students could use thumbs up/down as representations of yes/no, use arrows when writing algorithms to represent direction, or in code and decode words using numbers, pictographs, or other symbols to represent letters or words.

Standard 1.AP.2 - Break down (deconstruct) **algorithms** and list the steps needed to solve a problem into a sequence of tasks and sub-tasks.

(Practice 3: Recognizing and Defining Computational Problems)

Students will be able to identify the steps needed to solve a problem. Students will understand that algorithms are specific instructions in order to complete a familiar process or activity. Students may use digital tools to demonstrate ordering and sequencing of a task. Emphasize real life examples, ordering, and sequencing such as putting on your shoes, writing a story with a beginning, middle, and end, checking out library books, etc. For example, student may break down the steps needed to draw a shape or coding steps to move a character across the screen.

Algorithms and Programming (AP): *continued*

Standard 1.AP.3 - Create **programs** with **sequences** (steps) of commands and simple **loops** (repeated patterns), to express ideas or address a problem.

(Practice 5: Creating Computational Artifacts)

Students will create programs using elementary block programing (such as unplugged or ScratchJr on a device) that contain simple loops. These loops are repeating a pattern to create an image (such as a square), or to address a problem (such as hopscotch or designing a code that allows an avatar to avoid a barrier).

Impacts of Computing (IC):

Computing affects many aspects of the world in both positive and negative ways at local, national, and global levels. Individuals and communities influence computing through their behaviors and cultural and social interactions, and in turn, computing influences new cultural practices. An informed and responsible person should understand the social implications of the digital world, including equity and access to computing.

Standard 1.IC.1 – Develop and demonstrate the ability to work respectfully and responsibly with others whether communicating face-to-face or digitally.

(Practice 2: Collaborating Around Computing)

Students will develop and demonstrate proper etiquette when collaborating with others, physically or digitally.

Computational Thinking (CT):

Computational thinking (CT) is a problem-solving process that includes several characteristics, such as logically ordering and analyzing data and creating solutions using a series of ordered steps (or algorithms), and dispositions, such as the ability to confidently deal with complexity and open-ended problems. CT is essential to the development of computer applications, but it can also be used to support problem-solving across all disciplines, including math, science, and the humanities. Students who learn CT across the curriculum can begin to see a relationship between subjects as well as between school and life outside of the classroom.[8]

[8] *Exploring Computer Science - Google Resources. Retrieved from: https://edu.google.com/resources/programs/exploring-computational-thinking/*

Computational Thinking (CT): *continued*

Standard 1.CT.1 - Determine the steps needed to solve a problem and develop a **sequence** of instructions.

(Practice 3: Recognizing and Defining Computational Problems)

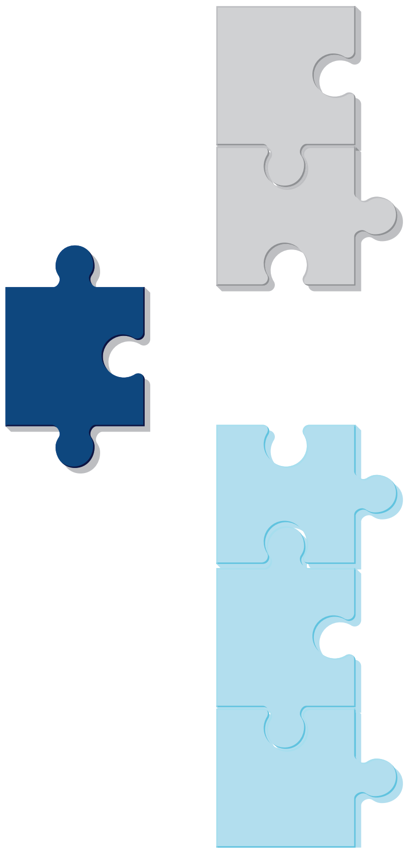
Students will analyze a real-world problem and develop instructions that determine the steps necessary to achieve the intended outcome such as creating a peanut butter and jelly sandwich. The connection with computer science is the need to be clear and detailed with action steps so the outcome is what is desired. Teachers may use digital tools to diagram components of a task such as making simple foods.

Standard 1.CT.2 - Recognize similarities between new problems and problems that have been solved in the past.

(Practice 3: Recognizing and Defining Computational Problems)

Students will have the opportunity to consider how previous problem-solving and code development have similarities. The use of previous solutions and strategies is useful in crafting a new solution. For example, how do the lessons learned from determining steps to create a peanut butter and jelly sandwich inform the development of steps to create a school lunch?

2nd Grade



Second Grade

Computing Systems (CS):

People interact with a wide variety of computing devices that collect, store, analyze, and act upon information in ways that can affect human capabilities, both positively and negatively. The physical components (hardware) and instructions (software) that make up a computing system communicate and process information in digital form. An understanding of hardware and software is useful when troubleshooting a computing system that does not work as intended.

Standard 2.CS.1 - Describe and solve basic hardware and software problems.

(Practice 7: Communicating About Computing)

Students will describe, solve and perform basic troubleshooting tasks such as checking the device for battery charge and/or power connection, checking cord connections, turning a device off and on to reboot it, closing and reopening an app/program, plugging in headphones, etc.

Network and the Internet (NI):

Computing devices typically do not operate in isolation. Networks connect computing devices to share information and resources and are an increasingly integral part of computing. Networks and communication systems provide greater connectivity in the computing world by providing fast, secure communication and facilitating innovation.

Standard 2.NI.1 - Explain what a password or pass phrase is, why it is used, and be able to create a secure password.

(Practice 7: Communicating About Computing)

Learning to protect one's device or information from unwanted use by others is an essential first step in learning about cybersecurity. Students will be able to explain the reasoning behind having certain digital resources password protected and create an effective password and/or pass phrase.

Data and Analysis (DA):

Computing systems exist to process data. The amount of digital data generated in the world is rapidly expanding, and the need to process data effectively is increasingly important. Data is collected and stored so it can be analyzed to better understand the world and make more accurate predictions.

Data and Analysis (DA): *continued*

Standard 2.DA.1 - Demonstrate how to **store**, copy, search, retrieve, modify and delete information using a **computing device**, and define the information stored as **data**.

(Practice 4: Developing and Using Abstractions)

Students will demonstrate how to create, modify, and save projects using the devices, platforms, applications, and software available. Data can be images, text documents, audio files, software programs or apps, video files, etc. The information is specific to data, not text within a single text document.

Standard 2.DA.2 - Collect and present **data** in various visual formats.

(Practice 4: Developing and Using Abstractions and Practice 7: Communicating About Computing)

Students will collect and present data in various visual formats, such as drawing a picture graph and a bar graph (with single-unit scale) with up to four categories. Students will create surveys of things that interest them (such as favorite foods, colors, books, etc.), collect answers, and decide how to present the data in various visual formats (picture graphs and bar graphs). Teachers can use digital tools to model data visualizations for students and students may collaborate to create and present digital data.

Standard 2.DA.3 - Identify and describe patterns in **data** visualizations to make predictions.

(Practice 4: Developing and Using Abstractions)

Students will make predictions by identifying and describing information in picture graphs and bar graphs. For example, the class may create and analyze a digital pictograph of favorite animals in the class. Students will make a prediction of favorite animals for the grade level based on patterns observed.

Algorithms and Programming (AP):

An algorithm is a sequence of steps designed to accomplish a specific task. Algorithms are translated into programs, or code, to provide instructions for computing devices. Algorithms and programming control all computing systems, empowering people to communicate with the world in new ways and solve compelling problems. The development process to create meaningful and efficient programs involves choosing which information to use and how to process and store it, breaking apart large problems into smaller ones, recombining existing solutions, and analyzing different solutions.

Algorithms and Programming (AP): *continued*

Standard 2.AP.1 - **Deconstruct** the steps needed to solve a task into a **sequence** of instructions.

(Practice 3: Recognizing and Defining Computational Problems)

Students will be able to identify the steps needed to solve a problem. Students will understand that algorithms are specific instructions in order to complete a familiar process or activity. Students may use digital tools to demonstrate ordering and sequencing of a task, such as code a character to move across the screen or solve a coding puzzle. Emphasize real life examples, ordering, and sequencing such as putting on your shoes, writing a story with a beginning, middle, and end, checking out library books, etc.

Standard 2.AP.2 - Collaboratively develop plans that describe a **program's sequence** of **events**, goals, and expected outcomes.

(Practice 5: Creating Computational Artifacts and Practice 7: Communicating About Computing)

Students will collaborate to develop a digital program using a sequence of events that includes goals and expected outcomes. The focus is on team or pair-programming on device and leveraging roles to develop a shared solution. For example, students may work in teams to navigate an avatar to reach a goal.

Standard 2.AP.3 - Properly credit others when using their ideas and creations while developing **programs**.

(Practice 7: Communicating About Computing)

Using computers comes with a level of responsibility. Students will properly credit work by citing work inspired by others when developing digital programs.

Standard 2.AP.4 - **Debug** and solve simple problems within an **algorithm** or **program** that includes **sequences** and simple **loops**.

(Practice 6: Testing and Refining Computational Artifacts)

Students will test algorithms to find problems and resolve errors (debug) within the program. Students will start with an existing code that includes sequences and/or simple loops and determine the errors in the code to make improvements to achieve the task. This can be done both on device and with unplugged activities. For example, students create an alternative emergency exit plan with alternative routes for evacuation drill.

Algorithms and Programming (AP): *continued*

Standard 2.AP.5 - Summarize the steps taken and choices made during the **iterative** process of **program** development.

(Practice 7: Communicating About Computing)

Students will summarize how a digital project was created and revised. For example, students will be able to answer who, what, where, when, why, how, etc. about the final program solution.

Impacts of Computing (IC):

Computing affects many aspects of the world in both positive and negative ways at local, national, and global levels. Individuals and communities influence computing through their behaviors and cultural and social interactions, and in turn, computing influences new cultural practices. An informed and responsible person should understand the social implications of the digital world, including equity and access to computing.

Standard 2.IC.1 - Describe how technology has impacted society over time.

(Practice 3: Recognizing and Defining Computational Problems)

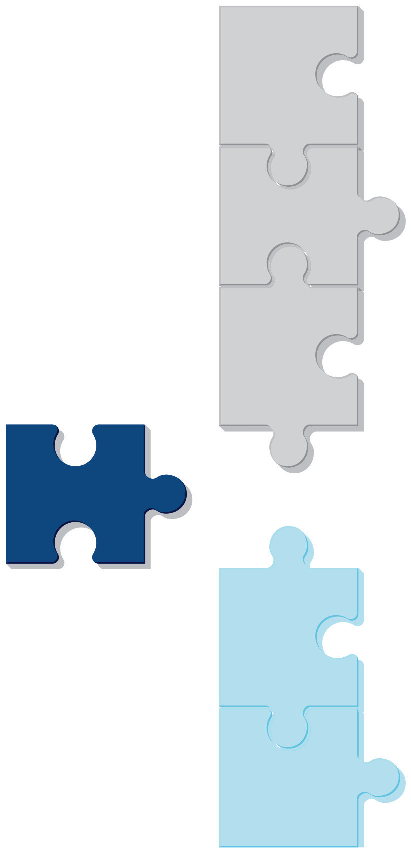
Students will compare the advances in technology and describe how it has impacted society. For example, students can consider how a cell phone saves phone numbers in contacts and consider how that impacts whether people know important phone numbers and how that impacts society.

Standard 2.IC.2 - Describe rationales for keeping login information private, and for logging off devices appropriately.

(Practice 3: Recognizing and Defining Computational Problems)

Students will describe why people keep passwords private and secure and demonstrate how to log on and off digital devices appropriately.

3rd Grade



Third Grade

Computing Systems (CS):

People interact with a wide variety of computing devices that collect, store, analyze, and act upon information in ways that can affect human capabilities, both positively and negatively. The physical components (hardware) and instructions (software) that make up a computing system communicate and process information in digital form. An understanding of hardware and software is useful when troubleshooting a computing system that does not work as intended.

Standard 3.CS.1 - Describe and model how **computing devices** connect to other components to extend their capabilities and form a **system**.

(Practice 7: Communicating About Computing)

Students will describe and model how computing devices connect to other devices or components (physical or wireless) to create a system. For example, the relationship between the respiratory and circulatory system during physical activity serves as a metaphor for how the parts of a computer connect to allow input, processing, and output.

Network and the Internet (NI):

Computing devices typically do not operate in isolation. Networks connect computing devices to share information and resources and are an increasingly integral part of computing. Networks and communication systems provide greater connectivity in the computing world by providing fast, secure communication and facilitating innovation.

Standard 3.NI.1 - Describe physical and digital **security** measures for protecting personal information.

(Practice 3: Recognizing and Defining Computational Problems)

Students will identify personal information and describe physical and digital measures for protecting it. Examples of physical security may include a lock on the door, a safe, covering the camera on your device. Examples of digital security may include virus protection software, strong passwords, biometric scanners (e.g., fingerprint, facial recognition), etc.

Network and the Internet (NI): *continued*

Standard 3.NI.2 - Develop personal patterns of behavior to protect information from u**nauthorized access**.

(Practice 4: Developing and Using Abstractions)

Students will begin to develop habits that protect their personal information. For example, using strong passwords, changing passwords often, logging off devices, etc.

Data and Analysis (DA):

Computing systems exist to process data. The amount of digital data generated in the world is rapidly expanding, and the need to process data effectively is increasingly important. Data is collected and stored so it can be analyzed to better understand the world and make more accurate predictions.

Standard 3.DA.1 - Organize and present collected **data** visually to highlight relationships and support a claim.

(Practice 7: Communicating About Computing)

Students will organize and present data collected using visualizations. For example, draw a scaled picture and scaled bar graph to represent data, with several categories. Gathering data may be used as an instructional strategy, but it is not required of students.

Standard 3.DA.2 - Use data to communicate ideas, highlight relationships, and predict outcomes.

(P7.1)

Students will use data to communicate ideas to emphasize relationships and predict outcomes. For example, using a scaled bar graph, students will predict what flavors of ice cream will be most popular among the third grade population.

Algorithms and Programming (AP):

An algorithm is a sequence of steps designed to accomplish a specific task. Algorithms are translated into programs, or code, to provide instructions for computing devices. Algorithms and programming control all computing systems, empowering people to communicate with the world in new ways and solve compelling problems. The development process to create meaningful and efficient programs involves choosing which information to use and how to process and store it, breaking apart large problems into smaller ones, recombining existing solutions, and analyzing different solutions.

Standard 3.AP.1 - Create **programs** that include **events**, **sequences**, **loops**, and simple **conditionals** to express ideas or address a problem.

(Practice 5: Creating Computational Artifacts)

Students will create programs using an elementary block coding program (e.g. ScratchJr.) that include events, sequences, loops, and simple conditionals to complete a task. The new components for third grade are events (starting your computer and having applications automatically start) and simple conditionals (if you click on the character then the character jumps 3 times).

Standard 3.AP.2 – **Modify a previously** created **program** that uses **variables** to **store** and **modify data**.

(Practice 5: Creating Computational Artifacts)

Students will save and modify data of previously created programs that use variables. For example, students can take an existing elementary block coding program (e.g. ScratchJr.) that collects what time you get up for school in the morning and modify it to collect what time you get home from school in the afternoon.

Standard 3.AP.3 - **Test** and **debug** a program or **algorithm** to ensure it accomplishes the intended task.

(Practice 6: Testing and Refining Computational Artifacts)

Students will test and make corrections (debug) to verify programs (an existing elementary block coding program or a recipe) run properly, similar to proofreading writing. The focus is on testing all aspects of a program before beginning the debugging process.

Algorithms and Programming (AP): *continued*

Standard 3.AP.4 - Perform different roles when collaborating with peers during the design, implementation, and review stages of **program** development.

(Practice 2: Collaborating Around Computing)

Students will collaborate, in a variety of roles, in the program development process (design, implementation, and review). This builds on the team or peer programming from the previous year. The students will take steps to define and select roles, as well as trading roles during the project to learn different aspects of collaboration in computer science. For example, roles may include being the navigator, developer, time manager, quality control, etc.

Standard 3.AP.5 - Use an **iterative** design process to plan and develop a **program** by considering the perspectives and preferences of others.

(Practice 1: Fostering an Inclusive Computing Culture and Practice 5: Creating Computational Artifacts)

Students will understand the process of planning (key features, time and resource constraints, and user expectations) before developing a program. Once the program is created, they will review the program with another team for feedback before revising (iterating) and creating an improved program.

Standard 3.AP.6 - Create **programs** by incorporating smaller portions of existing **programs** to develop something new or add more advanced features.

(Practice 4: Developing and Using Abstractions and Practice 5: Creating Computational Artifacts)

Students will incorporate pre-established programs into their original draft. The existing program will only address part of the necessary solution, requiring students to develop and add new code to achieve the desired outcome.
For example, using an existing program that collects data on student lunch preferences, and adding a feature that directs the program to display the class data at the end of the program.

Impacts of Computing (IC):

Computing affects many aspects of the world in both positive and negative ways at local, national, and global levels. Individuals and communities influence computing through their behaviors and cultural and social interactions, and in turn, computing influences new cultural practices. An informed and responsible person should understand the social implications of the digital world, including equity and access to computing.

Standard 3.IC.1 - Evaluate how **computing** technologies have changed the world, and express how those technologies influence, and are influenced by, cultural practices.

(Practice 3: Recognizing and Defining Computational Problems)

Students will evaluate how the advances in technology have impacted society and analyze how those technologies have influenced culture. For example, students may consider how the use of headphones has changed the world and consider societal changes such as how people wearing headphones may not engage in conversations while waiting for public transportation, but also have access to voice translation when speaking with people in different languages.

Standard 3.IC.2 - Describe reasons creators might limit the use of their work.

(Practice 7: Communicating About Computing)

Students will describe piracy and copyright and why owners limit the use of their work.

Computational Thinking (CT):

Computational thinking (CT) is a problem-solving process that includes several characteristics, such as logically ordering and analyzing data and creating solutions using a series of ordered steps (or algorithms), and dispositions, such as the ability to confidently deal with complexity and open-ended problems. CT is essential to the development of computer applications, but it can also be used to support problem-solving across all disciplines, including math, science, and the humanities. Students who learn CT across the curriculum can begin to see a relationship between subjects as well as between school and life outside of the classroom.[9]

[9] *Exploring Computer Science - Google Resources. Retrieved from: https://edu.google.com/resources/programs/exploring-computational-thinking/*

Computational Thinking (CT): *continued*

Standard 3.CT.1 - **Decompose** problems into smaller manageable tasks which may themselves be **decomposed**.

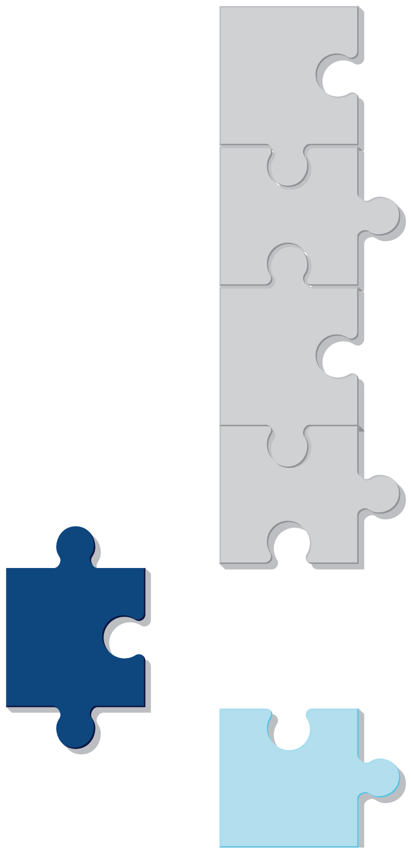
(Practice 3: Recognizing and Defining Computational Problems)

Students will consider a broader challenge, such as improving the classroom recycling program, and identify and decompose the problem into smaller tasks such as signage, education, using different receptacles for paper vs. plastic, etc.

Standard 3.CT.2 - Recognize common patterns between problems and recurring patterns within problems.

(Practice 3: Recognizing and Defining Computational Problems)

Students will be able to recognize common patterns within problems, such as the challenges of accommodating all students and parents at school drop off and similarities with challenges of the entire school eating lunch at the same time. After identifying the similarities in challenges, students can brainstorm other problem scenarios that share in these patterns.



4th Grade

Fourth Grade

Computing Systems (CS):

People interact with a wide variety of computing devices that collect, store, analyze, and act upon information in ways that can affect human capabilities, both positively and negatively. The physical components (hardware) and instructions (software) that make up a computing system communicate and process information in digital form. An understanding of hardware and software is useful when troubleshooting a computing system that does not work as intended.

Standard 4.CS.1 - Demonstrate how computer **hardware** and **software** work together as a **system** to accomplish tasks.

(Practice 4: Developing and Using Abstractions)

Students will describe and model how computing devices connect to other devices or components (physical or wireless) to create a system. For example, the relationship between the respiratory and circulatory system during physical activity serves as a metaphor for how the parts of a computer connect to allow input, processing, and output.

Network and the Internet (NI):

Computing devices typically do not operate in isolation. Networks connect computing devices to share information and resources and are an increasingly integral part of computing. Networks and communication systems provide greater connectivity in the computing world by providing fast, secure communication and facilitating innovation.

Standard 4.NI.1 - Model how information is broken down into smaller pieces called **packets** and transmitted through multiple **devices** over physical or wireless paths and reassembled at the destination.

(Practice 4: Developing and Using Abstractions)

Students will learn and model different pathways information travels to and from devices. For example, students will have a set of tennis balls (packets) that have information on each one. Students will hit tennis balls one at a time over the net. If the ball does not clear net, that represents packet loss and represents message not being sent. Tennis balls that clear the net will then be reassembled to deliver message.

Data and Analysis (DA):

Computing systems exist to process data. The amount of digital data generated in the world is rapidly expanding, and the need to process data effectively is increasingly important. Data is collected and stored so it can be analyzed to better understand the world and make more accurate predictions.

Standard 4.DA.1 - Select, organize, and categorize **data** and represent that **data** visually to provide clarity or support a claim.

(Practice 7: Communicating About Computing)

Students will organize and present data collected using visualizations. For example, when working with a data set of popular songs, data could be shown by genre or artist. Graphs, charts, and infographics can all represent the statistical characteristics of the data. An additional visualization may include making a line plot using provided data sets; include a horizontal scale, title, labels, and straight columns of symbols to represent the data points (• or X).

Standard 4.DA.2 - Use **data** to highlight and propose relationships, predict outcomes, or communicate ideas.

(Practice 7: Communicating About Computing)

Students will use data to communicate ideas to emphasize relationships and predict outcomes.
For example, demonstrating irrelevant data connections such as predicting age by eye color or predicting the outcome of an election by polling only a few people.

Algorithms and Programming (AP):

An algorithm is a sequence of steps designed to accomplish a specific task. Algorithms are translated into programs, or code, to provide instructions for computing devices. Algorithms and programming control all computing systems, empowering people to communicate with the world in new ways and solve compelling problems. The development process to create meaningful and efficient programs involves choosing which information to use and how to process and store it, breaking apart large problems into smaller ones, recombining existing solutions, and analyzing different solutions.

Algorithms and Programming (AP): *continued*

Standard 4.AP.1 - Compare and refine multiple **algorithms** for the same task, using computer and non-computer languages, and determine which is the most appropriate.

(Practice 3: Recognizing and Defining Computational Problems and Practice 6: Testing and Refining Computational Artifacts)

Students will collaborate, in a variety of roles, in the program development process (design, implementation, and review). This builds on the team or peer programming from the previous year. The students will take steps to define and select roles, as well as trading roles during the project to learn different aspects of collaboration in computer science.
For example, roles may include being the navigator, developer, time manager, quality control, etc.

Standard 4.AP.2 - Create **programs** that include **events**, **loops**, and **conditionals**.

(Practice 5: Creating Computational Artifacts)

Students will create a set of instructions (a program) that include events, loops, and conditionals to facilitate and manage tasks. Students will create programs using an elementary block coding program (e.g. ScratchJr.) that include events, sequences, loops, and simple conditionals to complete a task. Event examples include mouse clicks, typing on the keyboard, and collisions between objects. Conditional statements are sets of commands that are tied to specific actions based on whether the condition evaluates to TRUE or FALSE.

Standard 4.AP.3 - **Decompose** problems into smaller, manageable tasks which may be then be broken down further.

(Practice 3: Recognizing and Defining Computational Problems)

Students will decompose a program into smaller, more manageable parts. For example, decomposition at this level is creating an animation by separating a story into different scenes. For each scene, a background needs to be selected, characters placed, and actions programmed. The instructions required to program each scene may be like instructions in other programs.

Standard 4.AP.4 - Test and **debug** a **program** or **algorithm** to ensure it accomplishes the intended task.

(Practice 6: Testing and Refining Computational Artifacts)

Students will test and make corrections (debug) to verify programs run properly.

Impacts of Computing (IC):

Computing affects many aspects of the world in both positive and negative ways at local, national, and global levels. Individuals and communities influence computing through their behaviors and cultural and social interactions, and in turn, computing influences new cultural practices. An informed and responsible person should understand the social implications of the digital world, including equity and access to computing.

Standard 4.IC.1 - Evaluate **computing** technologies that have changed the world and express how those technologies influence and are influenced by cultural practices.

(Practice 3: Recognizing and Defining Computational Problems)

Students will evaluate how the advances in technology have impacted society and explain how those technologies have influenced culture. For examples, students can investigate the evolution of a technology (such as cameras, phones, or audio devices) and discuss the impact of those changes.

Standard 4.IC.2 - Propose ways to improve the accessibility and usability of technology products for the diverse needs and wants of users.

(Practice 1: Fostering an Inclusive Computing Culture)

Students will reference current technology and diverse user needs to brainstorm, collaborate, and propose innovative (new) technologically accessible ideas. For example, students may investigate voice-to-text, translation to other languages, and adaptive devices.

Computational Thinking (CT):

Computational thinking (CT) is a problem-solving process that includes several characteristics, such as logically ordering and analyzing data and creating solutions using a series of ordered steps (or algorithms), and dispositions, such as the ability to confidently deal with complexity and open-ended problems. CT is essential to the development of computer applications, but it can also be used to support problem-solving across all disciplines, including math, science, and the humanities. Students who learn CT across the curriculum can begin to see a relationship between subjects as well as between school and life outside of the classroom.[10]

[10] *Exploring Computer Science - Google Resources. Retrieved from: https://edu.google.com/resources/programs/exploring-computational-thinking/*

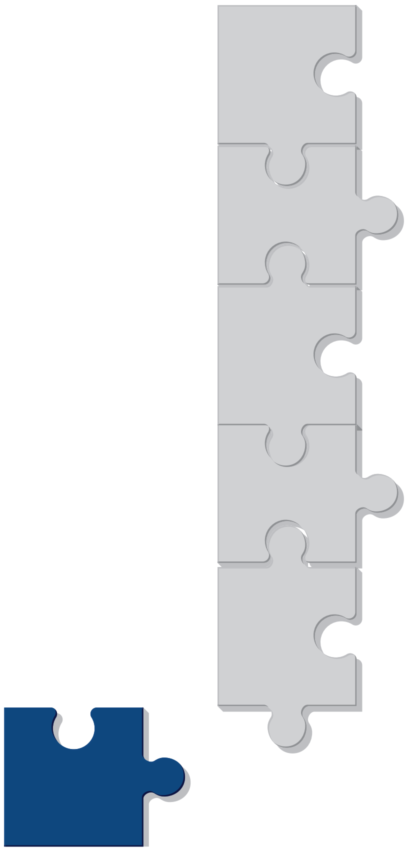
Computational Thinking (CT): *continued*

Standard 4.CT.1 - Determine specific aspects of patterns between or within problems that can be abstracted out to leave only the common or important elements.

(Practice 3: Recognizing and Defining Computational Problems and Practice 4: Developing and Using Abstractions)

Students will determine patterns within problems to identify core elements. Students will seek to identify key strategies to address the core elements, and then build a solution to address the comprehensive problem. For example, when the school is purchasing recess equipment, the students can identify possible challenges and problems that may exist for their community. Students can identify how to address those problems individually, then create a comprehensive solution to make sure recess is a success.

5th Grade



Fifth Grade

Computing Systems (CS):

People interact with a wide variety of computing devices that collect, store, analyze, and act upon information in ways that can affect human capabilities, both positively and negatively. The physical components (hardware) and instructions (software) that make up a computing system communicate and process information in digital form. An understanding of hardware and software is useful when troubleshooting a computing system that does not work as intended.

Standard 5.CS.1 - Create potential solutions to solve **hardware** and **software** problems using common **troubleshooting** strategies.

(Practice 4: Developing and Using Abstractions and Practice 6: Testing and Refining Computational Artifacts)

Students will find common hardware and software troubleshooting solutions. For example, checking power source, restarting programs and/or device, checking physical and wireless connections, etc.

Network and the Internet (NI):

Computing devices typically do not operate in isolation. Networks connect computing devices to share information and resources and are an increasingly integral part of computing. Networks and communication systems provide greater connectivity in the computing world by providing fast, secure communication and facilitating innovation.

Standard 5.NI.1 - Model how information is broken down into smaller pieces, transmitted as **packets** (**data** groups) through multiple devices over **networks** and the Internet, and reassembled at the destination.

(Practice 4: Developing and Using Abstractions)

Students will understand the functional use of routers and switches to send packets across multiple paths for communicating information to its destinations (such as wired connections, Wi-Fi, light (fiber optics), etc.). For example, students may create diagrams, models, written explanations, presentations etc. to demonstrate their understanding of the concept of transmitting packets.

Data and Analysis (DA):

Computing systems exist to process data. The amount of digital data generated in the world is rapidly expanding, and the need to process data effectively is increasingly important. Data is collected and stored so it can be analyzed to better understand the world and make more accurate predictions.

Standard 5.DA.1 - Explain how the amount of space required to **store data** differs based on the type of **data** and level of detail and that the utility of that data varies.

(Practice 7: Communicating About Computing)

Students will be able to explain that text files are smaller than picture files which are smaller than movie files. Additionally, students will be able to identify the utility of different data types, such as how you can use numerical data vs. alpha-numeric data in your analysis. For example, a number variable can only store a number and not letters. An alphanumeric variable can store numbers or letters but cannot be added or subtracted because it is text.

Standard 5.DA.2 - Organize and share collected **data** visually to highlight relationships and support a claim.

(Practice 7: Communicating About Computing)

Students will be able to refer to organized data when communicating an idea. For example, students may make a line plot about how many students share a length of their shoe size in inches using provided data sets; include a horizontal scale, title, labels, and straight columns of symbols to represent the data points (• or X).

Standard 5.DA.3 - Prioritize, analyze and use **data** to communicate ideas, highlight relationships and predict outcomes.

(Practice 7: Communicating About Computing)

Students should will be able to select and prioritize relevant data from large or complex data sets in support of a claim or to communicate the information in a more sophisticated manner. For example, looking at a data set of earthquakes over the past 20 years to determine what data is relevant to predict a future earthquake.

Algorithms and Programming (AP):

An algorithm is a sequence of steps designed to accomplish a specific task. Algorithms are translated into programs, or code, to provide instructions for computing devices. Algorithms and programming control all computing systems, empowering people to communicate with the world in new ways and solve compelling problems. The development process to create meaningful and efficient programs involves choosing which information to use and how to process and store it, breaking apart large problems into smaller ones, recombining existing solutions, and analyzing different solutions.

Standard 5.AP.1 - Compare and refine multiple **algorithms** for the same task and determine which is the most appropriate.

(Practice 3: Recognizing and Defining Computational Problems and Practice 6: Testing and Refining Computational Artifacts)

Students will compare different algorithms that achieve the same result, and determine which algorithm is more appropriate.
For example, students will compare different ways to get ready in the morning before school or which is the best route to get to the lunchroom.

Standard 5.AP.2 - **Decompose** problems into smaller, manageable tasks which may themselves be **deconstructed** and analyzed.

(Practice 3: Recognizing and Defining Computational Problems)

Students will decompose problems into smaller tasks to complete said task. For example, students creating maps of counties to compose a full state map or students working in teams to create acts to put on a three-scene play.

Standard 5.AP.3 - Create **programs** by incorporating smaller portions of existing **programs**, to develop something new or add more advanced features.

(Practice 4: Developing and Using Abstractions and Practice 5: Creating Computational Artifacts)

Students will create a new program, based on portions of existing programs.
For example, teacher gives a writing prompt where students create an animation and design alternative endings.

Algorithms and Programming (AP): *continued*

Standard 5.AP.4 - Use an **iterative** process to plan and develop a **program** by considering the perspectives and preferences of others.

(Practice 1: Fostering an Inclusive Computing Culture and Practice 5: Creating Computational Artifacts)

Students will plan and develop a solution for another person's problem. For example, a student has a hard time completing homework. The team designs a solution for how to manage time in order to complete homework, gathers data on the new solution, and revises the solution.

Standard 5.AP.5 - Recognize and observe **intellectual property rights** and give appropriate attribution when creating, **remixing**, or combining **programs**.

(Practice 5: Creating Computational Artifacts and Practice 7: Communicating About Computing)

Students will explain the concepts of ownership and sharing and be able to cite the owner. For example, a downloaded image may have restrictions that prohibit modification of an image or using it for commercial purposes.

Standard 5.AP.6 - Describe choices made during **program** development using code comments, presentations, and demonstrations.

(Practice 7: Communicating About Computing)

Students will describe the process used to develop a program using comments, presentations, and demonstrations. For example, students will be able to explain their selection of controlled variables in a science investigation.

Impacts of Computing (IC):

Computing affects many aspects of the world in both positive and negative ways at local, national, and global levels. Individuals and communities influence computing through their behaviors and cultural and social interactions, and in turn, computing influences new cultural practices. An informed and responsible person should understand the social implications of the digital world, including equity and access to computing.

Impacts of Computing (IC): *continued*

Standard 5.IC.1 - Propose ways to improve the accessibility and usability of technology products for the diverse needs and wants of users.

(Practice 1: Fostering an Inclusive Computing Culture)

Students will propose improvements of current technology based on needs and wants of a user. For example, having programs read in multiple languages, modifying hardware to meet the needs of a user, etc.

Standard 5.IC.2 - Seek and explain the impact of diverse perspectives for the purpose of improving **computational artifacts**.

(Practice 1: Fostering an Inclusive Computing Culture)

Students will research and explain how computing technologies influence, and are influenced by, cultural practices. For example, looking at school website and diverse student and parent needs to improve upon the website design and layout.

Computational Thinking (CT):

Computational thinking (CT) is a problem-solving process that includes several characteristics, such as logically ordering and analyzing data and creating solutions using a series of ordered steps (or algorithms), and dispositions, such as the ability to confidently deal with complexity and open-ended problems. CT is essential to the development of computer applications, but it can also be used to support problem-solving across all disciplines, including math, science, and the humanities. Students who learn CT across the curriculum can begin to see a relationship between subjects as well as between school and life outside of the classroom.[11]

Standard 5.CT.1 - Develop **algorithms** in computer **programs** to solve problems, including unique and repeated sub-tasks within a larger **program**.

(Practice 3: Recognizing and Defining Computational Problems and Practice 5: Creating Computational Artifacts)

Students will identify a problem and develop a computer program to find a solution to the problem. For example, students may investigate safety related to school routes and develop a program to gather data from students who use different modes of transportation.

[11] *Exploring Computer Science - Google Resources. Retrieved from: https://edu.google.com/resources/programs/exploring-computational-thinking/*

Glossary[12]

All Glossary definitions are attributed to the K-12 Computer Science Framework (2016).

Abstraction
Process: The process of reducing complexity by focusing on the main idea. By hiding details irrelevant to the question at hand and bringing together related and useful details, abstraction reduces complexity and allows one to focus on the problem.

Product: A new representation of a thing, a system, or a problem that helpfully reframes a problem by hiding details irrelevant to the question at hand. [MDESE, 2016]

Algorithm
A step-by-step process to complete a task.

Artifact
Anything created by a human. See computational artifact for the definition used in computer science.

Component
An element of a larger group. Usually, a component provides a particular service or group of related services. [Tech Terms, TechTarget]

Computational Artifacts
Anything created by a human using a computational thinking process and a computing device. A computational artifact can be, but is not limited to, a program, image, audio, video, presentation, or web page file. [College Board, 2016]

Computing
Any goal-oriented activity requiring, benefiting from, or creating algorithmic processes. [MDESE, 2016]

Computing Devices
A physical device that uses hardware and software to receive, process, and output information. Computers, mobile phones, and computer chips inside appliances are all examples of computing devices.

Computing System
A collection of one or more computers or computing devices, together with their hardware and software, integrated for the purpose of accomplishing shared tasks. Although a computing system can be limited to a single computer or computing device, it more commonly refers to a collection of multiple connected computers, computing devices, and hardware.

Conditionals
A feature of a programming language that performs different computations or actions depending on whether a programmer-specified Boolean condition evaluates to true or false. [MDESE, 2016] (A conditional could refer to a conditional statement, conditional expression, or conditional construct.)

Cybersecurity
The protection against access to, or alteration of, computing resources using technology, processes, and training. [TechTarget]

Data
Information that is collected and used for reference or analysis. Data can be digital or nondigital and can be in many forms, including numbers, text, show of hands, images, sounds, or video. [CAS, 2013; Tech Terms]

[12] *K-12 Computer Science Framework. (October 2016) Retrieved from: www.k12cs.org*

Debug
The process of finding and correcting errors (bugs) in programs. [MDESE, 2016]

Decompose; Decomposition
Decompose: To break down into components.
Decomposition: Breaking down a problem or system into components. [MDESE, 2016]

Deconstruct
Reduce (something) to its constituent parts in order to reinterpret it.

Device
A unit of physical hardware that provides one or more computing functions within a computing system. It can provide input to the computer, accept output, or both. [Techopedia]

Event
Any identifiable occurrence that has significance for system hardware or software. User-generated events include keystrokes and mouse clicks; system-generated events include program loading and errors. [TechTarget]

Hardware
The physical components that make up a computing system, computer, or computing device. [MDESE, 2016]

Intellectual Property Rights
Intellectual property rights are the rights given to persons over the creations of their minds. They usually give the creator an exclusive right over the use of his/her creation for a certain period of time

Iterative
Involving the repeating of a process with the aim of approaching a desired goal, target, or result. [MDESE, 2016]

Loop
A programming structure that repeats a sequence of instructions as long as a specific condition is true. [Tech Terms]

Modify
Make partial or minor changes to (something), typically to improve it or to make it less extreme.

Network
A group of computing devices (personal computers, phones, servers, switches, routers, etc.) connected by cables or wireless media for the exchange of information and resources.

Packet
The unit of data sent over a network. [Tech Terms]

Password; Pass Phrase:
A secret word or phrase that allows access to a computer system or service.

Program; Programming
Program (n): A set of instructions that the computer executes to achieve a particular objective. [MDESE, 2016]
Program (v): To produce a program by programming.
Programming: The craft of analyzing problems and designing, writing, testing, and maintaining programs to solve them. [MDESE, 2016]

Remixing
To create a new version of (a recording) by recombining and re-editing the elements of the existing recording and often adding material such as new vocals or instrumental tracks.

Security
See the definition for cybersecurity.

Sequence
One of the three basic logic structures in computer programming. The other two logic structures are selection and loop. In a sequence structure, an action, or event, leads to the next ordered action in a predetermined order.

Software
Programs that run on a computing system, computer, or other computing device.

Storage / Store
Place: A place, usually a device, into which data can be entered, in which the data can be held, and from which the data can be retrieved at a later time. [FOLDOC]
Process: A process through which digital data is saved within a data storage device by means of computing technology. Storage is a mechanism that enables a computer to retain data, either temporarily or permanently. [Techopedia]

System
A collection of elements or components that work together for a common purpose. [TechTarget]
See also the definition for computing system.

Test Case
A set of conditions or variables under which a tester will determine whether the system being tested satisfies requirements or works correctly. [STF]

Troubleshooting
A systematic approach to problem-solving that is often used to find and resolve a problem, error, or fault within software or a computing system. [Techopedia, TechTarget]

Unauthorized Access
Unauthorized access is when someone gains access to a website, program, server, service, or other system using someone else's account or other methods. For example, if someone kept guessing a password or username for an account that was not theirs until they gained access it is considered unauthorized access.

Unplugged
Computer Science without a computer.

Variables
A symbolic name that is used to keep track of a value that can change while a program is running. Variables are not just used for numbers; they can also hold text, including whole sentences (strings) or logical values (true or false). A variable has a data type and is associated with a data storage location; its value is normally changed during the course of program execution. [CAS, 2013; Techopedia]
Note: This definition differs from that used in math.

Bibliography

California K-12 Computer Science Standards. (January 2018)
Retrieved from: https://www.cde.ca.gov/be/st/ss/computerscicontentstds.asp

Exploring Computer Science - Google Resources.
Retrieved from: https://edu.google.com/resources/programs/exploring-computational-thinking/

K-8 Computer Science Georgia Standards of Excellence. (2019)
Retrieved from: https://www.gadoe.org/SiteAssets/Pages/Computer-Sci-
ence/K-8%20Computer%20Science%20Georgia%20Standards%20of%20Excellence.pdf

K-12 Computer Science Framework. (October 2016)
Retrieved from: https://k12cs.org/wp-content/uploads/2016/09/K%E2%80%9312-Com-puter-Science-
Framework.pdf

Michigan K-12 Computer Science Standards - Descriptive Statements. (May 2019)
Retrieved from: https://www.michigan.gov/documents/mde/CompSci_Standards_De-
scriptions_Final_655569_7.pdf

Utah Computer Science K-12 Framework. (October 2018).
Retrieved from: https://schools.utah.gov/file/46d4ca37-9d23-414e-91fd-6640b6be9df6