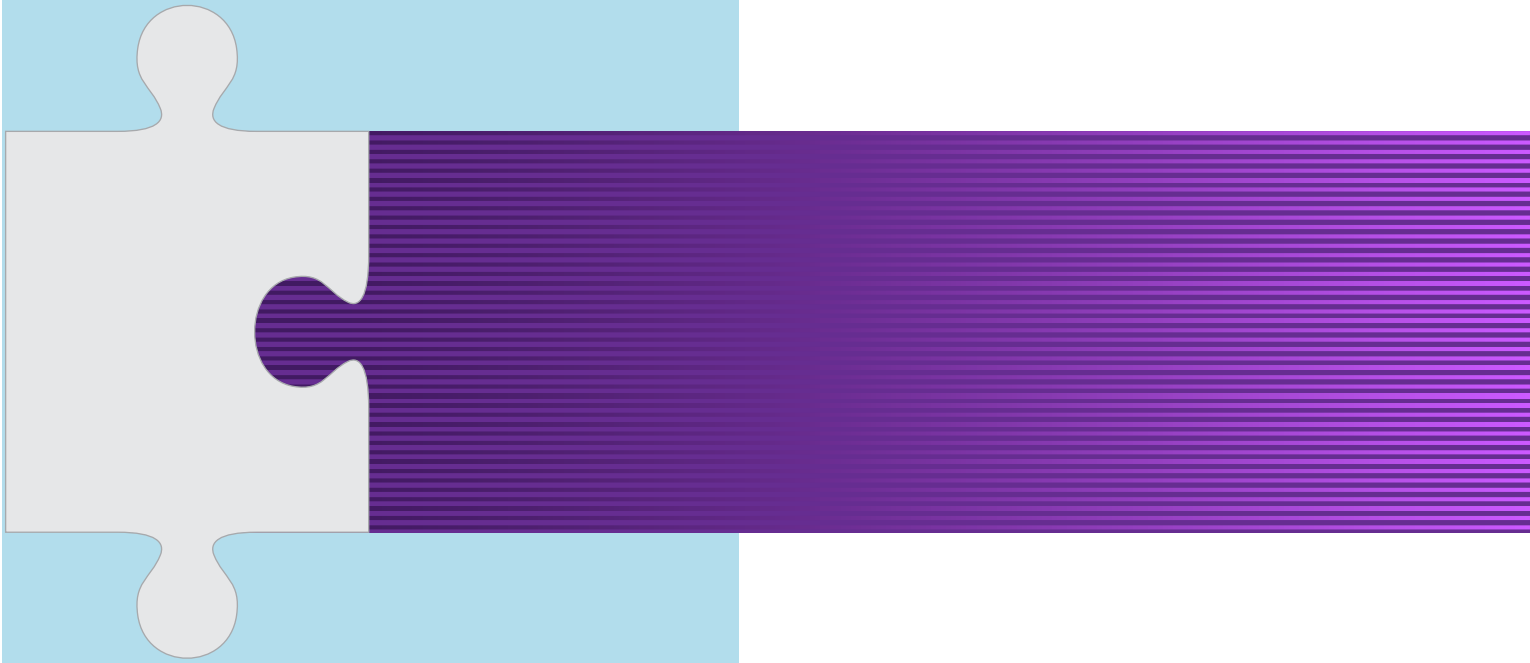
Utah K-5 Computer Science Standards Spring 2019



DRAFT

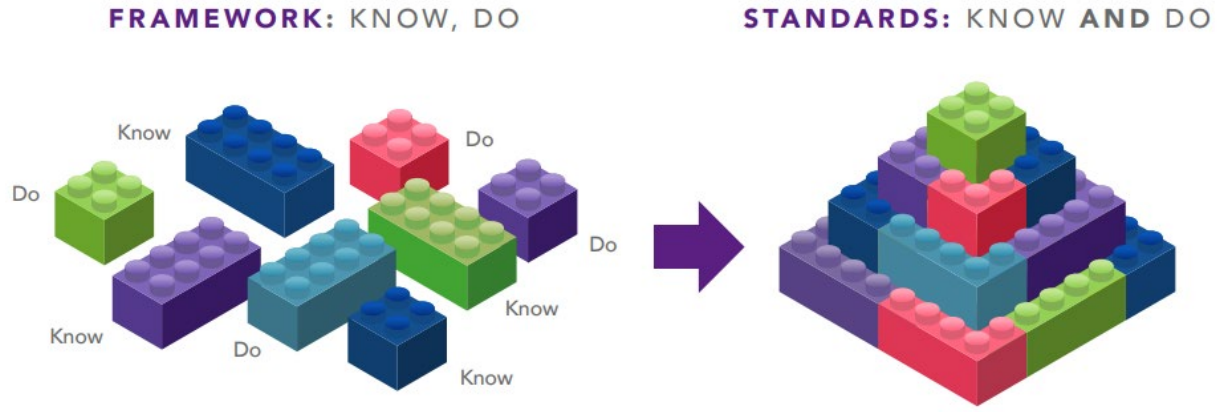
Utah State Board of Education

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 meet multiple times to identify a strategic planning of recommendations to successfully carry out computer science education within the K-12 schools. In June 2018, the Task Force's strategic priorities (steps) to accomplish the vision was presented to the Board and subsequently accepted by the Board. 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.
- Ensure students can access a majority (5 most popular courses include Computer Programming 1, Computer Science Principles, Exploring Computer Science, Web Development 1, and Game Development) of the 19 computer science courses currently offered, (30+ including IT courses); regardless of geography.

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 Utah State Board of Education in March 2019 to develop K-5 Computer Science Standards. The writers used the *Utah Computer Science K-12 Framework*[2] Statements 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/wp-content/uploads/2016/09/K%E2%80%9312-Computer-Science-Framework.pdf

Utah Computer Science Vision Statement:

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.

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.

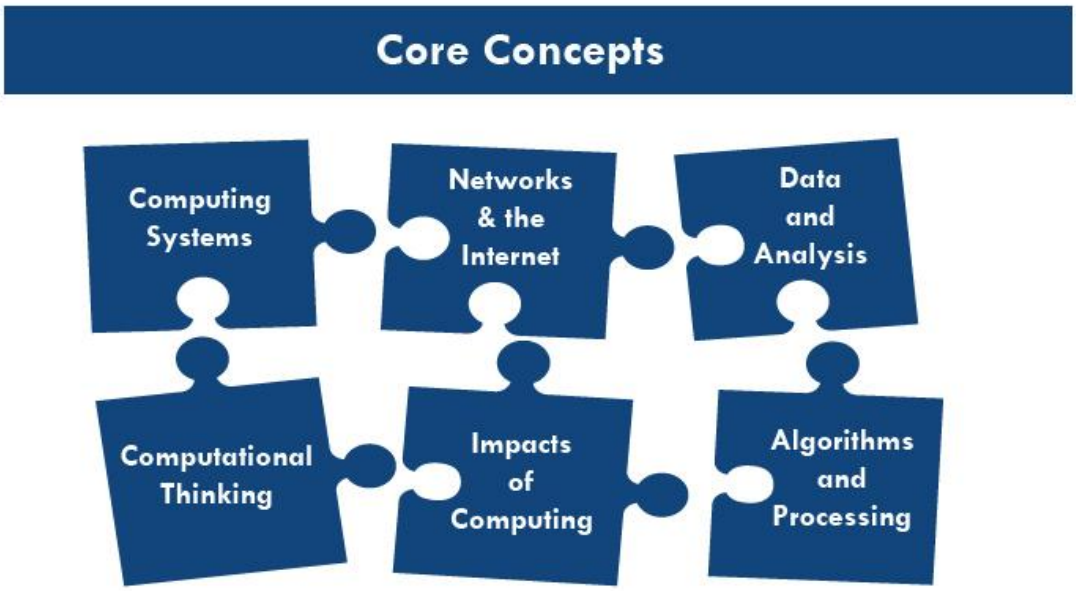
Within the standards there are words that are bold, underlined, and in green text. For example, look at Standard 2.DA.1.

Standard 2.DA.1 - Demonstrate how to **store**, copy, search, retrieve, **modify** and delete information using a **computing device**, and define the information stored as data. *(Practice 4: Developing and Using Abstractions)*

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]:



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/wp-content/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, so the need to process data effectively is increasingly important. Data is collected and stored so that 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):

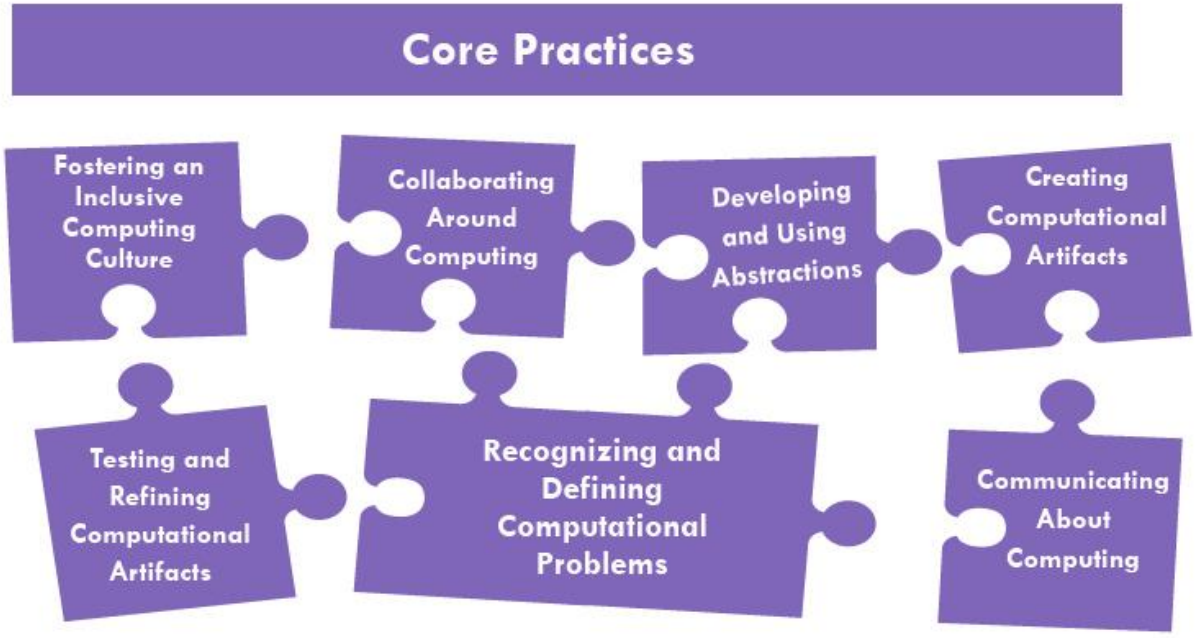
Computing affects many aspects of the world in both positive and negative ways at local, national, and global levels. Individuals and communities influence computing through their behaviors and cultural and social interactions, and in turn, computing influences new cultural practices. An informed and 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]:



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.

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.

[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

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.

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.

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.

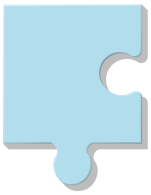
Practice 6: Testing and Refining Computational Artifacts

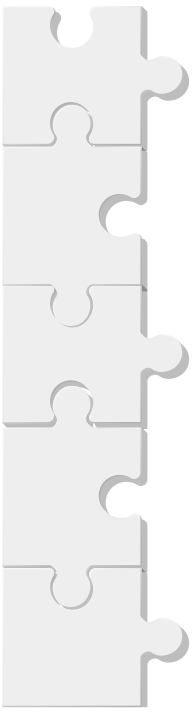
Testing and refinement is the deliberate and iterative process of improving a computational artifact. This process includes debugging (identifying and fixing errors) and comparing actual outcomes to intended outcomes. Students also respond to the changing needs and expectations of end users and improve the performance, reliability, usability, and accessibility of artifacts.

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.

1. 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 identify computing devices (phone, tablet, computer, etc.) and understand that they can be used to aid in a task (email, text message, voice calls, 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 how information is sent and retrieved, such as demonstrating through the action of “telephone” (asking students to pass a message from one person to another). This can include sending words, images, etc.

Standard K.NI.2 - Create patterns to communicate a message. *(Practice 4: Developing and Using Abstractions)*

Students will arrange pictures, objects or words in a sequence/pattern 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, so the need to process data effectively is increasingly important. Data is collected and stored so that 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 be able to use charts, graphs, and other visual data to be able to make predictions based on patterns. Examples include looking at the season's weather patterns to predict future weather or predicting next color in a sequence.

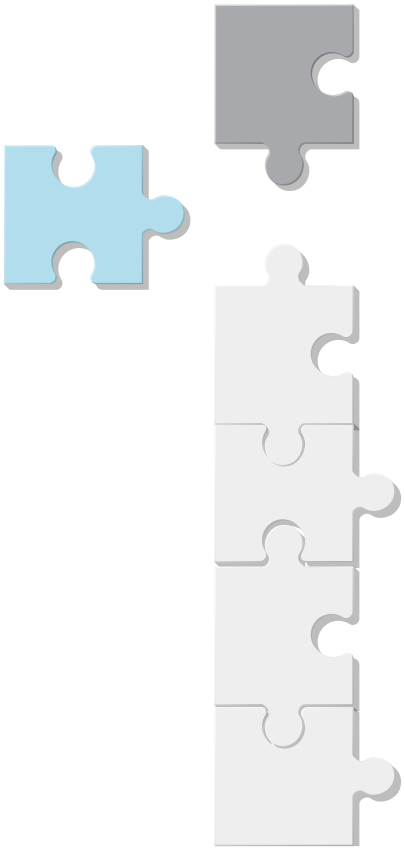
Algorithms and Programming (AP):

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 understand that algorithms are specific instructions in order to complete a familiar process or activity. (Emphasize real life examples, ordering, and sequencing.) Example: brushing teeth, lining up to go to lunch, school safety drills, etc.

2. 1st Grade



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 deciding what to wear to school or what to play at recess.

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

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 on digital devices (laptop, tablet, desktop, etc.) based on availability and the task they are seeking to accomplish.

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 identify common hardware and software components (hardware, software, mouse, keyboard, storage, trackpad, 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, so the need to process data effectively is increasingly important. Data is collected and stored so that 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 collect information and decide how to present the data in various visual formats (tally marks, color blocks stacked, sticky notes, etc.).

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 patterns and create hypothesizes based on data visualizations (charts, graphs, etc.). Examples include drawing a picture graph and a bar graph with single-unit scale 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.

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 understand how computers store data (input) in a variety of ways (variable) 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. Examples may include creating a Mad Lib except use numbers, letters, words, etc.

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. (Emphasize real life examples, ordering, and sequencing.) Example: putting on your shoes, writing a story with a beginning, middle, and end, checking out library books, etc.

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 (such as unplugged or ScratchJr) 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 the ability to** work respectfully and responsibly with others whether communicating face to face or electronically. *(Practice 2: Collaborating Around Computing)*

Students will develop proper etiquette when collaborating with others, physically or electronically.

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]

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)*

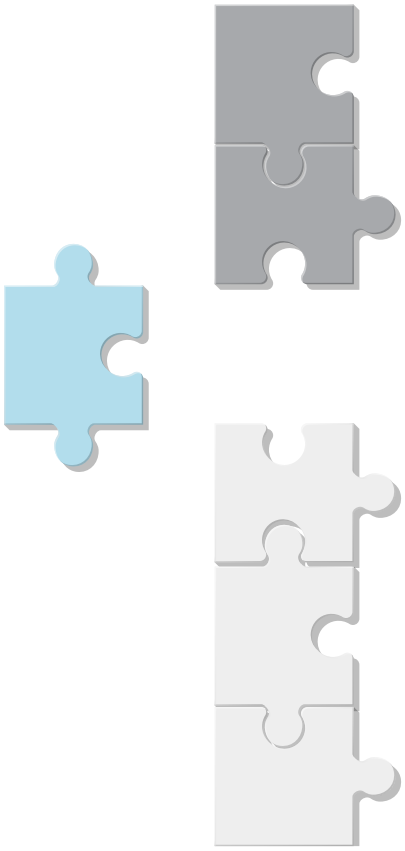
Students will analyze a real-world problem, such as creating a peanut butter and jelly sandwich, and develop instructions that define the steps necessary to achieve the intended outcome. The connection with computer science is the need to be clear and detailed with action steps so the outcome is what is desired.

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

Standard 1.CT.2 - Recognize similarities between new problems and problems they have 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?

3. 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 perform basic troubleshooting tasks. Examples may include checking the device for battery charge and/or power connection, checking cord connections, power down and restart, 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.N1.1 - Explain what a password is, why it is used, and be able to create a secure password. *(Practice 7: Communicating About Computing)*

Students will be able to develop an effective password with at least two of the following: uppercase letters, lowercase letters, numbers, and/or special characters and explain the reasoning behind having certain digital resources password protected.

Data and Analysis (DA):

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

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 navigate through applications/software to create, modify, and save projects using the devices and platforms available. 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 present collected data in various visual formats, for example drawing a picture graph and a bar graph (with single-unit scale) to represent a data set with up to four categories. Solve simple put-together, take-apart, and comparison problems using information presented in a bar graph.

Standard 2.DA.3 - Identify and describe patterns in **data** visualizations (i.e. charts or graphs) to make predictions. *(Practice 4: Developing and Using Abstractions)*

Students will solve put-together (addition) and take-apart (subtraction) problems using information in a bar graph and solve comparison problems using information in a bar graph.

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 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. (Emphasize real life examples, ordering, and sequencing.) Example: 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 with others to solve a problem and develop expected outcomes. The focus is on team or pair-programming and leveraging roles to develop a solution in partnership with others.

Standard 2.AP.3 - Properly credit others when using their ideas and creations while developing **programs**. *(Practice 7: Communicating About Computing)*

Students will properly cite work inspired by others when developing 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 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.

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 articulate how a project was created. 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.

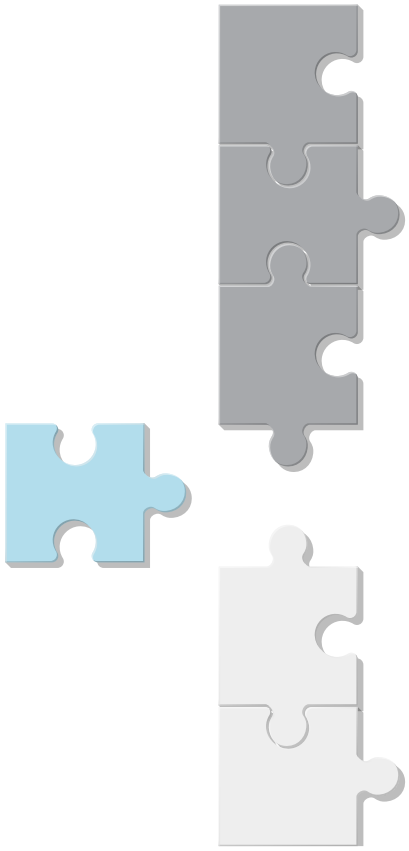
Standard 2.IC.1 - Describe how technology has impacted society over time. *(Practice 3: Recognizing and Defining Computational Problems)*

Students will compare the advances in technology and how it has impacted/impaired 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 explain why people keep passwords private and secure and demonstrate how to log on and off digital devices appropriately.

4. 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 understand computing devices connect to other devices or components (physical or wireless) to create a system. For example, the relationship among the respiratory and circulatory system during physical activity, 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 personal information. For example, virus protection software, creating data backups, creating strong passwords, biometric scanners (e.g., fingerprint, facial recognition), etc.

Standard 3.NI.2 - Develop personal patterns of behavior to protect information from **unauthorized access**. *(Practice 4: Developing and Using Abstractions)*

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

Data and Analysis (DA):

Computing systems exist to process data. The amount of digital data generated in the world is rapidly expanding, so the need to process data effectively is increasingly important. Data is collected and stored so that 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 communicate the meaning of 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, solve one and two-step problems using data from the scaled bar graph.

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 events, sequences, loops, and simple conditionals to complete a task. The new components are events and simple conditionals from the programming in second grade.

Standard 3.AP.2 - Create **programs** that use **variables** to **store** and **modify** **data**. *(Practice 5: Creating Computational Artifacts)*

Students will save and modify data that use variables. The focus is on modifying the data, such as creating a new file or program and saving it under a new file name.

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 to verify programs run properly. The focus is on the role of testing all aspects of a program before beginning the debugging process.

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.

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 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.

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/impaired 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 headphones impact communication between strangers.

Standard 3.IC.2 - Describe reasons creators might limit the use of their work. *(Practice 7: Communicating About Computing)*

Students will understand piracy and copyrights 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]

Standard 3.CT.1 - **Decompose** problems into smaller manageable tasks which may themselves be **decomposed**. *(Practice 3: Recognizing and Defining Computational Problems)*

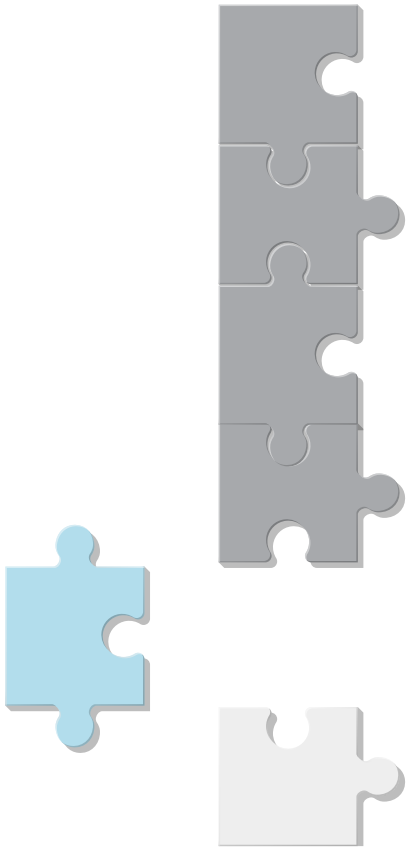
Students will consider a broader challenge, such as improving the classroom recycling program, and identify and decompose the problem into smaller tasks such as signage, education, using different receptacles for paper vs. plastic, etc.

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 analyze 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.

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

5. 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. (Prac*tice 4: Developing and Using Abstractions)*

Students will understand computing devices connect to other devices or components (physical or wireless) to accomplish a task. For example, the relationship among the respiratory and circulatory system during physical activity, the parts of a computer connect to allow input, processing, and output, 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 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 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, so the need to process data effectively is increasingly important. Data is collected and stored so that 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 communicate the meaning of data collected using visualizations. For example, when working with a data set of popular songs, for example, data could be shown by genre or artist. Graphs, charts, infographics, ratios 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). Additionally, use a variety of strategies to solve addition and subtraction problems related to data on a line plot.

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. Additionally, students may make a line plot to display a data set of measurements in fractions of a unit (halves, quarters, and eighths) and then solve problems involving addition and subtraction with like denominators of fractions by using information presented in line plots. For example, use a line plot to find and interpret the difference in length between the longest and shortest pencils in a classroom.

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 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 develop an algorithm while determining which language (computer, non-computer) is best to use to accomplish a task. For example, students can create blueprints that can include natural language, flowcharts, and pseudocode. Different algorithms can be used to tie shoes or decide which path to take on the way home from school. While the end results may be similar, the solution may not be the same: in the example of going home, some paths could be faster, slower, or more direct, depending on varying factors, such as available time or the presence of obstacles (for example, a barking dog).

Standard 4.AP.2 - Create **programs** that include **events**, **loops**, and **conditionals**. *(Practice 5: Creating Computational Artifacts)*

Students will develop a set of instructions (a program) that include events, loops, and conditionals to facilitate and manage tasks. 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. Other terms that can be used to specify the appropriate groups of instructions to execute under various conditions include AND, OR, and NOT.

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 dissect 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 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/impaired society and analyze 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 use current technology and diverse user needs to brainstorm or collaborate innovative (new) technologically-accessible ideas.

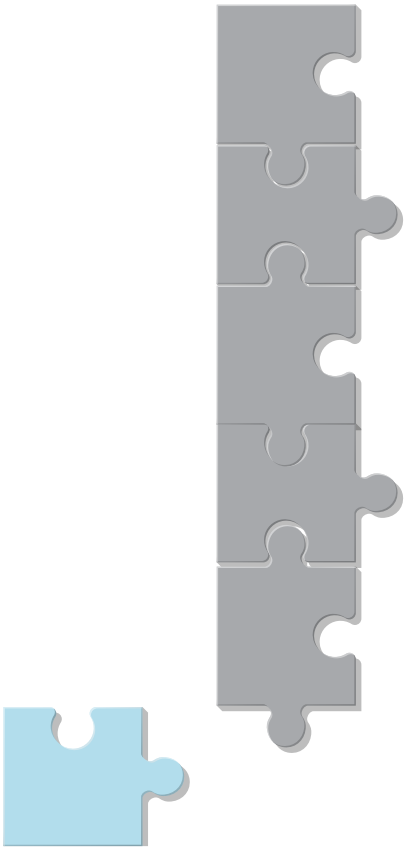
Computational Thinking (CT):

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]

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 look for patterns within problems to identify core elements. Upon identification of core elements, students will seek to identify key strategies to address the core elements, and then build a solution to address the comprehensive problem.

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



6. 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)*

Computing systems share similarities, such as the use of power, data, and memory. Starting with simple solutions like checking that power is available, checking that physical and wireless connections are working, and clearing out the working memory by restarting programs or devices. Just as a computer is like the human body: CPU is like the brain (does the thinking), Motherboard is like the Central Nervous System (helps one part communicate to another) and the Power Supply is like the Heart (supplying power/blood). Students can identify issues that may be hidden if they understand the functions of the parts.

Network and the Internet (NI):

Students will learn that 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 learn and model that Information needs a physical or wireless path to travel to be sent and received, and some paths are better than others. Information is broken into smaller pieces, called packets, that are sent independently and reassembled at the destination. Routers and switches are used to properly send packets across paths to their destinations.

Data and Analysis (DA):

Computing systems exist to process data. The amount of digital data generated in the world is rapidly expanding, so the need to process data effectively is increasingly important. Data is collected and stored so that 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. alphanumeric data in your analysis.

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 data when communicating an idea. Students will be able to synthesize data and present basic data using visual representations, such as storyboards, flowcharts, and graphs. For example, students may make 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 5.DA.3 - Prioritize, analyze and use **data** to communicate ideas, highlight relationships and predict outcomes. *(Practice 7: Communicating About Computing)*

Students will be able to refer to data when communicating an idea. Students will organize the data in larger sets to make interpreting and communicating it to others easier, such as through the creation of basic data representations. Students should will be able to select relevant data from large or complex data sets in support of a claim or to communicate the information in a more sophisticated manner.

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 analyze different algorithms that can achieve the same result, recognizing that some algorithms are more appropriate for a specific context than others. For example, different algorithms can be used to tie shoes or decide which path to take on the way home from school.

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 deconstruct programs, recognizing that they can be broken down into smaller parts to facilitate their design, implementation, and review. Young students may think of an animation as multiple scenes and thus create each scene independently.

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 learn that programs can also be created by incorporating smaller portions of programs that have already been created. For example, students will be able to choose from a set of given commands to create simple animated stories or solve pre-existing problems.

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 learn and model that design often involves reusing existing code or remixing other programs within a community. For example, students will participate in project planning and the creation of brainstorming documents to inform the design and iteration of a program.

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. They will also recognize the contributions of collaborators. 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 provide documentation for end users that explains their artifacts and how they function, and they will both give and receive feedback. For example, students could provide a project overview and ask for input from users. Students will incorporate clear comments in their product and document their process using text, graphics, presentations, and demonstrations.

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 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 learn how the development and modification of computing technology is driven by people's needs and wants and can affect groups differently. For example, new computing technology is created, and existing technologies are modified to increase their benefits (for example, Internet search recommendations), decrease their risks (for example, autonomous cars), and meet societal demands (for example, smartphone apps).

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 explain examples of how computing technologies influence, and are influenced by, cultural practices. For example, increased Internet access and speed have allowed people to share cultural information but have also affected the practice of traditional cultural customs.

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 in their community and develop a computer program to solve the program and/or gather data to help inform the final solution. 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).

Term	Definitions
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]

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

	(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]
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]
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.