K12 COMPUTER SCIENCE FRAMEWORK

VIEW BY CONCEPT

The Concepts and Practices of the K–12 Computer Science Framework

Core Concepts

1. Computing Systems
2. Networks and the Internet
3. Data and Analysis
4. Algorithms and Programming
5. Impacts of Computing

Core Practices

1. Fostering an Inclusive Computing Culture
2. Collaborating Around Computing
3. Recognizing and Defining Computational Problems
4. Developing and Using Abstractions
5. Creating Computational Artifacts
6. Testing and Refining Computational Artifacts
7. Communicating About Computing

Suggested citation: K–12 Computer Science Framework. (2016). *Framework view by concept*. Retrieved from http://www.k12cs.org

Suggested attribution: "The K–12 Computer Science Framework, led by the Association for Computing Machinery, Code.org, Computer Science Teachers Association, Cyber Innovation Center, and National Math and Science Initiative in partnership with states and districts, informed the development of this work."

How to refer to the concepts: [Grade Band].[Core Concept].[Subconcept]
Example: K–2.Algorithms and Programming.Program Development

How to refer to the practices: P[Practice Number].[Core Practice].[Practice Statement Number]
Example: P4.Developing and Using Abstractions.1

Practices

Practice 1. Fostering an Inclusive Computing Culture

Overview: 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 12, 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. Students who are well-versed in fostering an inclusive computing culture should be able to differentiate backgrounds and skillsets and know when to call upon others, such as to seek out knowledge about potential end users or intentionally seek input from people with diverse backgrounds.

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 preferences 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 examine 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. At the higher grades, students should become aware of professionally accepted accessibility standards and should be able to evaluate computational artifacts for accessibility. Students should also begin to identify potential bias during the design process to maximize accessibility in product design. For example, they can test an app and recommend to its designers that it respond to verbal commands to accommodate users who are blind or have physical disabilities.

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. As students progress, they should advocate for their peers when accommodations, such as an assistive-technology peripheral device, are needed for someone to use a computational artifact. Eventually, students should regularly advocate for both themselves and others.

Practice 2. Collaborating Around Computing

Overview: 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 12, 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 sophisticated 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. As they progress, students should use methods for giving all group members a chance to participate. Older students should strive to improve team efficiency and effectiveness by regularly evaluating group dynamics. They should use multiple strategies to make team dynamics more productive. For example, they can ask for the opinions of quieter team members, minimize interruptions by more talkative members, and give individuals credit for their ideas and their work.

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 experience 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. Eventually, students should independently organize their own teams and create common goals, expectations, and equitable workloads. They should also manage project workflow using agendas and timelines and should evaluate workflow to identify areas for improvement.

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. Older students should engage in active listening by using questioning skills and should respond empathetically to others. As they progress, students should be able to receive feedback from multiple peers and should be able to differentiate opinions. Eventually, students should seek contributors from various environments. These contributors may include end users, experts, or general audiences from online forums.

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 teamwork, 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. Eventually, students should use different collaborative tools and methods to solicit input from not only team members and classmates but also others, such as participants in online forums or local communities.

Practice 3. Recognizing and Defining Computational Problems

Overview: 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 12, 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. Older students should identify more complex problems that involve multiple criteria and constraints. Eventually, students should be able to identify real-world problems that span multiple disciplines, such as increasing bike safety with new helmet technology, and can be solved computationally.

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 subgoals: getting input from the user, processing the data, and displaying the result to the user. Eventually, as students encounter complex real-world problems that span multiple disciplines or social systems, they should decompose complex problems into manageable subproblems that could potentially be solved with programs or procedures that already exist. For example, students could create an app to solve a community problem that connects to an online database through an application programming interface (API).

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 subproblems that can be solved computationally (P3.1), they should begin to evaluate whether a computational solution is the most appropriate solution for a particular problem. For example, students might question whether using a computer to determine whether someone is telling the truth would be advantageous. As students progress, they should systematically evaluate the feasibility of using computational tools to solve given problems or subproblems, such as through a cost-benefit analysis. Eventually, students should include more factors in their evaluations, such as how efficiency affects feasibility or whether a proposed approach raises ethical concerns.

Practice 4. Developing and Using Abstractions

Overview: 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 12, 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. Eventually, students should extract common features from more complex phenomena or processes. For example, students should be able to identify common features in multiple segments of code and substitute a single segment that uses variables to account for the differences. In a procedure, the variables would take the form of parameters. When working with data, students should be able to identify important aspects and find patterns in related data sets such as crop output, fertilization methods, and climate conditions.

2. **Evaluate existing technological functionalities** and **incorporate** them into new designs.

   At all levels, students should be able to use well-defined abstractions that hide complexity. Just as a car hides operating details, such as the mechanics of the engine, a computer program's "move" command relies on hidden details that cause an object to change location on the screen. As they progress, students should incorporate predefined functions into their designs, understanding that they do not need to know the underlying implementation details of the abstractions that they use. Eventually, students should understand the advantages of, and be comfortable using, existing functionalities (abstractions) including technological resources created by other people, such as libraries and application programming interfaces (APIs). Students should be able to evaluate existing abstractions to determine which should be incorporated into designs and how they should be incorporated. For example, students could build powerful apps by incorporating existing services, such as online databases that return geolocation coordinates of street names or food nutrition information.

3. **Create modules** and **develop points of interaction** that can apply to multiple situations and reduce complexity.

   After students have had some experience identifying patterns (P4.1), decomposing problems (P3.2), using abstractions (P4.2), and taking advantage of existing resources (P4.2), they should begin to develop their own abstractions. As they progress, students should take advantage of opportunities to develop generalizable modules. For example, students could write more efficient programs by designing procedures that are used multiple times in the program. These procedures can be generalized by defining parameters that create different outputs for a wide range of inputs. Later on, students should be able to design systems of interacting modules, each with a well-defined role, that coordinate to accomplish a common goal. Within an object-oriented programming context, module design may include defining the interactions among objects. At this stage, these modules, which combine both data and procedures, can be designed and documented for reuse in other programs. Additionally, students can design points of interaction, such as a simple user interface, either text or graphical, that reduces the complexity of a solution and hides lower-level implementation details.

4. **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. Older students should model phenomena as systems, with rules governing the interactions within the system. Students should analyze and evaluate these models against real-world observations. For example, students might create a simple producer–consumer ecosystem model using a programming tool. Eventually, they could progress to creating more complex and realistic interactions between species, such as predation, competition, or symbiosis, and evaluate the model based on data gathered from nature.

Practice 5. Creating Computational Artifacts

Overview: 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 12, 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, taking into account 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. As learning progresses, students should systematically plan the development of a program or artifact and intentionally apply computational techniques, such as decomposition and abstraction, along with knowledge about existing approaches to artifact design. Students should be capable of reflecting on and, if necessary, modifying the plan to accommodate end goals.

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. Eventually, students should engage in independent, systematic use of design processes to create artifacts that solve problems with social significance by seeking input from broad audiences.

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. Later on, they should modify or remix parts of existing programs to develop something new or to add more advanced features and complexity. For example, students could modify prewritten code from a single-player game to create a two-player game with slightly different rules.

Practice 6. Testing and Refining Computational Artifacts

Overview: 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.

By the end of Grade 12, 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. Eventually, testing should become a deliberate process that is more iterative, systematic, and proactive. Older students should be able to anticipate errors and use that knowledge to drive development. For example, students can test their program with inputs associated with all potential scenarios.

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. Eventually, older students should progress to using more complex strategies for identifying and fixing errors, such as printing the value of a counter variable while a loop is running to determine how many times the loop runs.

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. Later on, evaluation and refinement should become an iterative process that also encompasses making artifacts more usable and accessible (P1.2). For example, students can incorporate feedback from a variety of end users to help guide the size and placement of menus and buttons in a user interface.

Practice 7. Communicating About Computing

Overview: 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 12, 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 through the use of 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. Eventually, students should 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.

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. Eventually, students should consider common licenses that place limitations or restrictions on the use of computational artifacts. For example, a downloaded image may have restrictions that prohibit modification of an image or using it for commercial purposes.

Concepts

Computing Systems

Overview: 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.

By the end of Grade 2:	By the end of Grade 5:	By the end of Grade 8:	By the end of Grade 12:
DEVICES **Overview:** Many everyday objects contain computational components that sense and act on the world. In early grades, students learn features and applications of common computing devices. As they progress, students learn about connected systems and how the interaction between humans and devices influences design decisions.			
People use computing devices to perform a variety of tasks accurately and quickly. Computing devices interpret and follow the instructions they are given literally. *Computing devices can be used to do a number of things, such as play music, create documents, and send pictures. Computing devices are also very precise. For example, computers can perform multiple complex calculations much faster and with greater accuracy than people. While people may diverge from instructions given to them, computers will follow instructions exactly as they are given, even if they do not achieve the intended result.* ***Crosscutting Concept:*** *Human–Computer Interaction* ***Connections Within Framework:*** *K–2.Algorithms and Programming.Control; K–2.Algorithms and Programming.Modularity; 3–5.Algorithms and Programming.Control*	Computing devices may be connected to other devices or components to extend their capabilities, such as sensing and sending information. Connections can take many forms, such as physical or wireless. Together, devices and components form a system of interdependent parts that interact for a common purpose. *Computing devices often depend on other devices or components. For example, a robot depends on a physically attached light sensor to detect changes in brightness, whereas the light sensor depends on the robot for power. A smartphone can use wirelessly connected headphones to send audio information, and the headphones are useless without a music source.* ***Crosscutting Concepts:*** *Communication and Coordination; System Relationships*	The interaction between humans and computing devices presents advantages, disadvantages, and unintended consequences. The study of human–computer interaction can improve the design of devices and extend the abilities of humans. *Accessibility is an important consideration in the design of any computing system. For example, assistive devices provide capabilities such as scanning written information and converting it to speech. The use of computing devices also has potential consequences, such as in the areas of privacy and security. For example, GPS-enabled smartphones can provide directions to a destination yet unintentionally allow a person to be tracked for malicious purposes. Also, the attention required to follow GPS directions can lead to accidents due to distracted driving.* ***Crosscutting Concepts:*** *Human–Computer Interaction; Privacy and Security*	Computing devices are often integrated with other systems, including biological, mechanical, and social systems. These devices can share data with one another. The usability, dependability, security, and accessibility of these devices, and the systems they are integrated with, are important considerations in their design as they evolve. *A medical device can be embedded inside a person to monitor and regulate his or her health, a hearing aid (a type of assistive device) can filter out certain frequencies and magnify others, a monitoring device installed in a motor vehicle can track a person's driving patterns and habits, and a facial recognition device can be integrated into a security system to identify a person. The devices embedded in everyday objects, vehicles, and buildings allow them to collect and exchange data, creating a network (e.g., Internet of Things). The creation of integrated or embedded systems is not an expectation at this level.*

Table continued on next page

Table continued from previous page

	Connection Within Framework: 3–5. Networks and the Internet.Network Communication and Organization	*Connection Within Framework: 3–5.Impacts of Computing.Culture*	*Crosscutting Concepts: System Relationships; Human–Computer Interaction; Privacy and Security* *Connections Within Framework: 9–12. Networks and the Internet.Network Communication and Organization; 9–12. Data and Analysis.Collection; 9–12. Impacts of Computing.Culture*

HARDWARE AND SOFTWARE

Overview: Computing systems use hardware and software to communicate and process information in digital form. In early grades, students learn how systems use both hardware and software to represent and process information. As they progress, students gain a deeper understanding of the interaction between hardware and software at multiple levels within computing systems.

A computing system is composed of hardware and software. Hardware consists of physical components, while software provides instructions for the system. These instructions are represented in a form that a computer can understand.

Examples of hardware include screens to display information and buttons, keys, or dials to enter information. Software applications are programs with specific purposes, such as a web browser or game. A person may use a mouse (hardware) to click on a button displayed in a web browser (software) to navigate to a new web page. Computing systems convert instructions, such as "print," "save," or "crop," into a special language that the computer can understand. At this level, understanding that computer information is encoded is appropriate, but the explicit understanding of "bits" is reserved for later grade levels.

Crosscutting Concept: Communication and Coordination

Connections Within Framework: K–2. Algorithms and Programming. Algorithms; K–2.Algorithms and Programming.Control

Hardware and software work together as a system to accomplish tasks, such as sending, receiving, processing, and storing units of information as bits. Bits serve as the basic unit of data in computing systems and can represent a variety of information.

For example, a photo filter application (software) works with a camera (hardware) to produce a variety of effects that change the appearance of an image. This image is transmitted and stored as bits, or binary digits, which are commonly represented as 0s and 1s. All information, including instructions, is encoded as bits. Knowledge of the inner workings of hardware and software, number systems such as binary or hexadecimal, and how bits are represented in physical media are not priorities at this level.

Crosscutting Concepts: Communication and Coordination; Abstraction

Connection Within Framework: 3–5. Data and Analysis.Storage

Hardware and software determine a computing system's capability to store and process information. The design or selection of a computing system involves multiple considerations and potential tradeoffs, such as functionality, cost, size, speed, accessibility, and aesthetics.

The capability of a computing system is determined by the processor speed, storage capacity, and data transmission speed, as well as other factors. Selecting one computing system over another involves balancing a number of tradeoffs. For example, selecting a faster computer with more memory involves the tradeoffs of speed and cost. Choosing one operating system over another involves the tradeoff of capability and compatibility, such as which apps can be installed or which devices can be connected. Designing a robot requires choosing both hardware and software and may involve a tradeoff between the potential for customization and ease of use. The use of a device that connects wirelessly through a Bluetooth connection versus a device that connects physically through a USB connection involves a tradeoff between mobility and the need for an additional power source for the wireless device.

Crosscutting Concepts: System Relationships; Communication and Coordination

Connection Within Framework: 6–8. Data and Analysis.Collection

Levels of interaction exist between the hardware, software, and user of a computing system. The most common levels of software that a user interacts with include system software and applications. System software controls the flow of information between hardware components used for input, output, storage, and processing.

At its most basic level, a computer is composed of physical hardware and electrical impulses. Multiple layers of software are built upon the hardware and interact with the layers above and below them to reduce complexity. System software manages a computing device's resources so that software can interact with hardware. For example, text editing software interacts with the operating system to receive input from the keyboard, convert the input to bits for storage, and interpret the bits as readable text to display on the monitor. System software is used on many different types of devices, such as smart TVs, assistive devices, virtual components, cloud components, and drones. Knowledge of specific, advanced terms for computer architecture, such as BIOS, kernel, or bus, is not expected at this level.

Crosscutting Concepts: Abstraction; Communication and Coordination; System Relationships

Connections Within Framework: 9–12. Networks and the Internet.Network Communication and Organization; 9–12. Algorithms and Programming. Variables; 9–12.Algorithms and Programming. Modularity

TROUBLESHOOTING

Overview: When computing systems do not work as intended, troubleshooting strategies help people solve the problem. In early grades, students learn that identifying the problem is the first step to fixing it. As they progress, students learn systematic problem-solving processes and how to develop their own troubleshooting strategies based on a deeper understanding of how computing systems work.

Computing systems might not work as expected because of hardware or software problems. Clearly describing a problem is the first step toward finding a solution. *Problems with computing systems have different causes, such as hardware settings, programming errors, or faulty connections to other devices. Developmentally appropriate ways to solve problems include debugging simple programs and seeking help by clearly describing a problem (for example, "The computer won't turn on," "The pointer on the screen won't move," or "I lost the web page.") Knowing how to diagnose or troubleshoot a problem with a computing system is not expected.* *Crosscutting Concept: System Relationships* *Connection Within Framework: 3–5. Algorithms and Programming.Program Development*	Computing systems share similarities, such as the use of power, data, and memory. Common troubleshooting strategies, such as checking that power is available, checking that physical and wireless connections are working, and clearing out the working memory by restarting programs or devices, are effective for many systems. *Although computing systems may vary, common troubleshooting strategies can be used on them, such as checking connections and power or swapping a working part in place of a potentially defective part. Rebooting a machine is commonly effective because it resets the computer. Because computing devices are composed of an interconnected system of hardware and software, troubleshooting strategies may need to address both.* *Crosscutting Concepts: System Relationships; Abstraction* *Connection Within Framework: 3–5. Networks and the Internet.Network Communication and Organization*	Comprehensive troubleshooting requires knowledge of how computing devices and components work and interact. A systematic process will identify the source of a problem, whether within a device or in a larger system of connected devices. *Just as pilots use checklists to troubleshoot problems with aircraft systems, people can use a similar, structured process to troubleshoot problems with computing systems and ensure that potential solutions are not overlooked. Because a computing device may interact with interconnected devices within a system, problems may not be due to the specific computing device itself but to devices connected to it. Examples of system components that may need troubleshooting are physical and wireless connections, peripheral equipment, and network hardware. Strategies for troubleshooting a computing system and debugging a program include some problem-solving steps that are similar.* *Crosscutting Concepts: System Relationships; Abstraction* *Connection Within Framework: 6–8.Algorithms and Programming. Algorithms*	Troubleshooting complex problems involves the use of multiple sources when researching, evaluating, and implementing potential solutions. Troubleshooting also relies on experience, such as when people recognize that a problem is similar to one they have seen before or adapt solutions that have worked in the past. *Troubleshooting information may come from external sources, such as user manuals, online technical forums, or manufacturer websites. Distinguishing between reliable and unreliable sources is important. Examples of complex troubleshooting strategies include resolving connectivity problems, adjusting system configurations and settings, ensuring hardware and software compatibility, and transferring data from one device to another.* *Crosscutting Concepts: Abstraction; System Relationships* *Connection Within Framework: 9–12. Algorithms and Programming.Program Development*

Networks and the Internet

Overview: 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.

NETWORK COMMUNICATION AND ORGANIZATION

Overview: Computing devices communicate with each other across networks to share information. In early grades, students learn that computers connect them to other people, places, and things around the world. As they progress, students gain a deeper understanding of how information is sent and received across different types of networks.

By the end of Grade 2:	By the end of Grade 5:	By the end of Grade 8:	By the end of Grade 12:
Computer networks can be used to connect people to other people, places, information, and ideas. The Internet enables people to connect with others worldwide through many different points of connection. *Small, wireless devices, such as cell phones, communicate with one another through a series of intermediary connection points, such as cellular towers. This coordination among many computing devices allows a person to voice call a friend or video chat with a family member. Details about the connection points are not expected at this level.* *Crosscutting Concepts: Communication and Coordination; Human–Computer Interaction* *Connections Within Framework: K–2.Impacts of Computing. Social Interactions; K–2.Data and Analysis. Collection; 3–5.Impacts of Computing. Social Interactions*	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. *There are physical paths for communicating information, such as ethernet cables, and wireless paths, such as Wi-Fi. Often, information travels on a combination of physical and wireless paths; for example, wireless paths originate from a physical connection point. The choice of device and type of connection will affect the path information travels and the potential bandwidth (the capacity to transmit data or bits in a given timeframe). Packets and packet switching (the method used to send packets) are the foundation for further understanding of Internet concepts. At this level, the priority is understanding the flow of information, rather than details of how routers and switches work and how to compare paths.*	Computers send and receive information based on a set of rules called protocols. Protocols define how messages between computers are structured and sent. Considerations of security, speed, and reliability are used to determine the best path to send and receive data. *Protocols allow devices with different hardware and software to communicate, in the way that people with different native languages may use a common language for business. Protocols describe established commands and responses between computers on a network, such as requesting data or sending an image. There are many examples of protocols including TCP/IP (Transmission Control Protocol/Internet Protocol) and HTTP (Hypertext Transfer Protocol), which serve as the foundation for formatting and transmitting messages and data, including pages on the World Wide Web. Routers also implement protocols to record the fastest and most reliable paths by sending small packets as tests. The priority at this grade level is understanding the purpose of protocols, while knowing details of how specific protocols work is not expected.*	Network topology is determined, in part, by how many devices can be supported. Each device is assigned an address that uniquely identifies it on the network. The scalability and reliability of the Internet are enabled by the hierarchy and redundancy in networks. *Large-scale coordination occurs among many different machines across multiple paths every time a web page is opened or an image is viewed online. Devices on the Internet are assigned an Internet Protocol (IP) address to allow them to communicate. The design decisions that directed the coordination among systems composing the Internet also allowed for scalability and reliability. Scalability is the capability of a network to handle a growing amount of work or its potential to be enlarged to accommodate that growth.* *Crosscutting Concepts: Communication and Coordination; Abstraction; System Relationships*

Table continued on next page

Table continued from previous page

	Crosscutting Concept: *Communication and Coordination* ***Connections Within Framework:*** *3–5. Computing Systems.Devices; 3–5. Computing Systems.Troubleshooting*	***Crosscutting Concepts:*** *Communication and Coordination; Abstraction; Privacy and Security* ***Connection Within Framework:*** *6–8. Data and Analysis.Storage*	***Connections Within Framework:*** *9–12. Computing Systems.Devices; 9–12. Computing Systems.Hardware and Software; 9–12.Impacts of Computing. Social Interactions*

CYBERSECURITY

Overview: Transmitting information securely across networks requires appropriate protection. In early grades, students learn how to protect their personal information. As they progress, students learn increasingly complex ways to protect information sent across networks.

Connecting devices to a network or the Internet provides great benefit, care must be taken to use authentication measures, such as strong passwords, to protect devices and information from unauthorized access. *Authentication is the ability to verify the identity of a person or entity. Usernames and passwords, such as those on computing devices or Wi-Fi networks, provide a way of authenticating a user's identity. Because computers make guessing weak passwords easy, strong passwords have characteristics that make them more time-intensive to break.* ***Crosscutting Concepts:*** *Privacy and Security; Communication and Coordination* ***Connection Within Framework:*** *K–2. Impacts of Computing.Safety, Law, and Ethics*	Information can be protected using various security measures. These measures can be physical and/or digital. *An offline backup of data is useful in case of an online security breach. A variety of software applications can monitor and address viruses and malware and alert users to their presence. Security measures can be applied to a network or individual devices on a network. Confidentiality refers to the protection of information from disclosure to unauthorized individuals, systems, or entities.* ***Crosscutting Concept:*** *Privacy and Security* ***Connection Within Framework:*** *3–5. Impacts of Computing.Safety, Law, and Ethics*	The information sent and received across networks can be protected from unauthorized access and modification in a variety of ways, such as encryption to maintain its confidentiality and restricted access to maintain its integrity. Security measures to safeguard online information proactively address the threat of breaches to personal and private data. *The integrity of information involves ensuring its consistency, accuracy, and trustworthiness. For example, HTTPS (Hypertext Transfer Protocol Secure) is an example of a security measure to protect data transmissions. It provides a more secure browser connection than HTTP (Hypertext Transfer Protocol) because it encrypts data being sent between websites. At this level, understanding the difference between HTTP and HTTPS, but not how the technologies work, is important.* ***Crosscutting Concept:*** *Privacy and Security* ***Connection Within Framework:*** *6–8. Impacts of Computing.Safety, Law, and Ethics*	Network security depends on a combination of hardware, software, and practices that control access to data and systems. The needs of users and the sensitivity of data determine the level of security implemented. *Security measures may include physical security tokens, two-factor authentication, and biometric verification, but every security measure involves tradeoffs between the accessibility and security of the system. Potential security problems, such as denial-of-service attacks, ransomware, viruses, worms, spyware, and phishing, exemplify why sensitive data should be securely stored and transmitted. The timely and reliable access to data and information services by authorized users, referred to as availability, is ensured through adequate bandwidth, backups, and other measures.* ***Crosscutting Concepts:*** *Privacy and Security; System Relationships; Human–Computer Interaction* ***Connection Within Framework:*** *9–12. Algorithms and Programming. Algorithms*

Data and Analysis

Overview: 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.

By the end of Grade 2:	By the end of Grade 5:	By the end of Grade 8:	By the end of Grade 12:
COLLECTION **Overview:** Data is collected with both computational and noncomputational tools and processes. In early grades, students learn how data about themselves and their world is collected and used. As they progress, students learn the effects of collecting data with computational and automated tools.			
Everyday digital devices collect and display data over time. The collection and use of data about individuals and the world around them is a routine part of life and influences how people live. *Many everyday objects, such as cell phones, digital toys, and cars, can contain tools (such as sensors) and computers to collect and display data from their surroundings.* *Crosscutting Concept: Human–Computer Interaction* *Connection Within Framework: K–2. Networks and the Internet.Network Communication and Organization*	People select digital tools for the collection of data based on what is being observed and how the data will be used. For example, a digital thermometer is used to measure temperature and a GPS sensor is used to track locations. *There is a wide array of digital data collection tools; however, only some are appropriate for certain types of data. Tools are chosen based upon the type of measurement they use as well as the type of data people wish to observe. Data scientists use the term observation to describe data collection, whether or not a human is involved in the collection.* *Crosscutting Concept: Abstraction* *Connections Within Framework: 3–5. Algorithms and Programming.Variables; 3–5.Algorithms and Programming. Algorithms*	People design algorithms and tools to automate the collection of data by computers. When data collection is automated, data is sampled and converted into a form that a computer can process. For example, data from an analog sensor must be converted into a digital form. The method used to automate data collection is influenced by the availability of tools and the intended use of the data. *Data can be collected from either individual devices or systems. The method of data collection (for example, surveys versus sensor data) can affect the accuracy and precision of the data. Some types of data are more difficult to collect than others. For example, emotions must be subjectively evaluated on an individual basis and are thus difficult to measure across a population. Access to tools may be limited by factors including cost, training, and availability.* *Crosscutting Concept: Human–Computer Interaction* *Connection Within Framework: 6–8. Computing Systems.Hardware and Software*	Data is constantly collected or generated through automated processes that are not always evident, raising privacy concerns. The different collection methods and tools that are used influence the amount and quality of the data that is observed and recorded. *Data can be collected and aggregated across millions of people, even when they are not actively engaging with or physically near the data collection devices. This automated and nonevident collection can raise privacy concerns, such as social media sites mining an account even when the user is not online. Other examples include surveillance video used in a store to track customers for security or information about purchase habits or the monitoring of road traffic to change signals in real time to improve road efficiency without drivers being aware. Methods and devices for collecting data can differ by the amount of storage required, level of detail collected, and sampling rates. For example, ultrasonic range finders are good at long distances and are very accurate, as compared to infrared range finders, which are better for short distances. Computer models and simulations produce large amounts of data used in analysis.*

Table continued on next page

Table continued from previous page

			Crosscutting Concept: *Privacy and Security* ***Connections Within Framework:*** *9–12. Computing Systems.Devices; 9–12. Impacts of Computing.Safety, Law, and Ethics*

STORAGE

Overview: Core functions of computers are storing, representing, and retrieving data. In early grades, students learn how data is stored on computers. As they progress, students learn how to evaluate different storage methods, including the tradeoffs associated with those methods.

Computers store data that can be retrieved later. Identical copies of data can be made and stored in multiple locations for a variety of reasons, such as to protect against loss. *For example, pictures can be stored on a cell phone and viewed later, or progress in a game can be saved and continued later. The advantage of recording data digitally, such as in images or a spreadsheet, versus on a physical space, such as a whiteboard, is that old data (states of data collected over time) can be easily retrieved, copied, and stored in multiple places. This is why personal information put online can persist for a long time. Understanding local versus online storage is not expected at this level.* *Crosscutting Concepts: System Relationships; Privacy and Security* *Connections Within Framework: K–2.Impacts of Computing.Social Interactions; K–2.Algorithms and Programming.Variables*	Different software tools used to access data may store the data differently. The type of data being stored and the level of detail represented by that data affect the storage requirements. *Music, images, video, and text require different amounts of storage. Video will often require more storage than music or images alone because video combines both. For example, two pictures of the same object can require different amounts of storage based upon their resolution. Different software tools used to access and store data may add additional data about the data (metadata), which results in different storage requirements. An image file is a designed representation of a real-world image and can be opened by either an image editor or a text editor, but the text editor does not know how to translate the data into the image. Understanding binary or 8-bit versus 16-bit representations is not expected at this level.* *Crosscutting Concept: System Relationships* *Connections Within Framework: 3–5.Computing Systems.Hardware and Software; 3–5.Algorithms and Programming.Variables*	Applications store data as a representation. Representations occur at multiple levels, from the arrangement of information into organized formats (such as tables in software) to the physical storage of bits. The software tools used to access information translate the low-level representation of bits into a form understandable by people. *Computers can represent a variety of data using discrete values at many different levels, such as characters, numbers, and bits. Text is represented using character encoding standards like UNICODE, which represent text as numbers. All numbers and other types of data are encoded and stored as bits on a physical medium. Lossy and lossless data formats are used to store different levels of detail, but whenever digital data is used to represent analog measurements, such as temperature or sound, information is lost. Representations, or file formats, can contain metadata that is not always visible to the average user. There are privacy implications when files contain metadata, such as the location where a photograph was taken.* *Crosscutting Concept: Abstraction* *Connections Within Framework: 6–8. Algorithms and Programming. Variables; 6–8.Networks and the Internet.Network Communication and Organization*	Data can be composed of multiple data elements that relate to one another. For example, population data may contain information about age, gender, and height. People make choices about how data elements are organized and where data is stored. These choices affect cost, speed, reliability, accessibility, privacy, and integrity. *A data model combines data elements and describes the relationships among the elements. Data models represent choices made about which data elements are available and feasible to collect. Storing data locally may increase security but decrease accessibility. Storing data on a cloud-based, redundant storage system may increase accessibility but reduce security, as it can be accessed online easily, even by unauthorized users. Data redundancies and backups are useful for restoring data when integrity is compromised.* *Crosscutting Concepts: System Relationships; Privacy and Security; Communication and Coordination* *Connection Within Framework: 9–12.Algorithms and Programming. Algorithms*

VISUALIZATION AND TRANSFORMATION

Overview: Data is transformed throughout the process of collection, digital representation, and analysis. In early grades, students learn how transformations can be used to simplify data. As they progress, students learn about more complex operations to discover patterns and trends and communicate them to others.

Data can be displayed for communication in many ways. People use computers to transform data into new forms, such as graphs and charts. *Examples of displays include picture graphs, bar charts, or histograms. A data table that records a tally of students' favorite colors can be displayed as a chart on a computer.* ***Crosscutting Concept:*** *Abstraction* ***Connection Within Framework:*** *K–2. Impacts of Computing. Social Interactions*	People select aspects and subsets of data to be transformed, organized, clustered, and categorized to provide different views and communicate insights gained from the data. *Data is often sorted or grouped to provide additional clarity. Data points can be clustered by a number of commonalities without a category label. For example, a series of days might be grouped by temperature, air pressure, and humidity and later categorized as fair, mild, or extreme weather. The same data could be manipulated in different ways to emphasize particular aspects or parts of the data set. For example, when working with a data set of popular songs, data could be shown by genre or artist. Simple data visualizations include graphs and charts, infographics, and ratios that represent statistical characteristics of the data.* ***Crosscutting Concept:*** *Abstraction: Human–Computer Interaction* ***Connection Within Framework:*** *6–8.Impacts of Computing.Social Interactions*	Data can be transformed to remove errors, highlight or expose relationships, and/or make it easier for computers to process. *The cleaning of data is an important transformation for reducing noise and errors. An example of noise would be the first few seconds of a sample in which an audio sensor collects extraneous sound created by the user positioning the sensor. Errors in survey data are cleaned up to remove spurious or inappropriate responses. An example of a transformation that highlights a relationship is representing two groups (such as males and females) as percentages of a whole instead of as individual counts. Computational biologists use compression algorithms to make extremely large data sets of genetic information more manageable and the analysis more efficient.* ***Crosscutting Concept:*** *Abstraction* ***Connection Within Framework:*** *6–8.Algorithms and Programming. Algorithms*	People transform, generalize, simplify, and present large data sets in different ways to influence how other people interpret and understand the underlying information. Examples include visualization, aggregation, rearrangement, and application of mathematical operations. *Visualizations, such as infographics, can obscure data and positively or negatively influence people's views of the data. People use software tools or programming to create powerful, interactive data visualizations and perform a range of mathematical operations to transform and analyze data. Examples of mathematical operations include those related to aggregation, such as summing and averaging. The same data set can be visualized or transformed to support multiple sides of an issue.* ***Crosscutting Concept:*** *Abstraction: Human–Computer Interaction* ***Connection Within Framework:*** *6–8.Impacts of Computing.Social Interactions*

INFERENCE AND MODELS

Overview: Data science is one example where computer science serves many fields. Computer science and science use data to make inferences, theories, or predictions based upon the data collected from users or simulations. In early grades, students learn about the use of data to make simple predictions. As they progress, students learn how models and simulations can be used to examine theories and understand systems and how predictions and inferences are affected by more complex and larger data sets.

Data can be used to make inferences or predictions about the world. Inferences, statements about something that cannot be readily observed, are often based on observed data. Predictions, statements about future events, are based on patterns in data and can be made by looking at data visualizations, such as charts and graphs.

Observations of people's clothing (jackets and coats) can be used to make an inference about the weather (it is cold outside). Patterns in past data can be identified and extrapolated to make predictions. For example, a person's lunch menu selection can be predicted by using data on past lunch selections.

Crosscutting Concept: *Abstraction*

Connection Within Framework: *K–2. Impacts of Computing.Culture*

The accuracy of inferences and predictions is related to how realistically data is represented. Many factors influence the accuracy of inferences and predictions, such as the amount and relevance of data collected.

People use data to highlight or propose cause-and-effect relationships and predict outcomes. Basing inferences or predictions on data does not guarantee their accuracy; the data must be relevant and of sufficient quantity. An example of irrelevance is using eye color data when inferring someone's age. An example of insufficient quantity is predicting the outcome of an election by polling only a few people.

Crosscutting Concept: *System Relationships*

Computer models can be used to simulate events, examine theories and inferences, or make predictions with either few or millions of data points. Computer models are abstractions that represent phenomena and use data and algorithms to emphasize key features and relationships within a system. As more data is automatically collected, models can be refined.

Very large data sets require a model for analysis; they are too large to be analyzed by examining all of the records. While individual users are online, shopping websites and online advertisements use personal data they generate, compared to millions of other users, to predict what they would like and make recommendations. A video-streaming website may recommend videos based on models generated from other users and based upon their personal habits and preferences. The data that is collected about an individual and potential inferences made from that data can have implications for privacy.

Crosscutting Concepts: *Privacy and Security; Abstraction*

Connections Within Framework: *6–8.Algorithms and Programming. Algorithms; 6–8.Impacts of Computing.Culture*

The accuracy of predictions or inferences depends upon the limitations of the computer model and the data the model is built upon. The amount, quality, and diversity of data and the features chosen can affect the quality of a model and ability to understand a system. Predictions or inferences are tested to validate models.

Large data sets are used to make models used for inference or predictions, such as forecasting earthquakes, traffic patterns, or the results of car crashes. Larger quantities and higher quality of collected data will tend to improve the accuracy of models. For example, using data from 1,000 car crashes would generally yield a more accurate model than using data from 100 crashes. Additionally, car crashes provide a wide variety of data points, such as impact speed, car make and model, and passenger type, and this data is useful in the development of injury prevention measures.

Crosscutting Concepts: *Abstraction; Privacy and Security*

Algorithms and Programming

Overview: 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.

By the end of Grade 2:	By the end of Grade 5:	By the end of Grade 8:	By the end of Grade 12:
ALGORITHMS **Overview:** Algorithms are designed to be carried out by both humans and computers. In early grades, students learn about age-appropriate algorithms from the real world. As they progress, students learn about the development, combination, and decomposition of algorithms, as well as the evaluation of competing algorithms.			
People follow and create processes as part of daily life. Many of these processes can be expressed as algorithms that computers can follow. *Routines, such as morning meeting, clean-up time, and dismissal, are examples of algorithms that are common in many early elementary classrooms. Other examples of algorithms include making simple foods, navigating a classroom, and daily routines like brushing teeth. Just as people use algorithms to complete daily routines, they can program computers to use algorithms to complete different tasks. Algorithms are commonly implemented using a precise language that computers can interpret.* *Crosscutting Concept: Abstraction* *Connection Within Framework: K–2. Computing Systems.Hardware and Software*	Different algorithms can achieve the same result. Some algorithms are more appropriate for a specific context than others. *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, they 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). Algorithms can be expressed in noncomputer languages, including natural language, flowcharts, and pseudocode.* *Crosscutting Concept: Abstraction* *Connection Within Framework: 3–5. Data and Analysis.Collection*	Algorithms affect how people interact with computers and the way computers respond. People design algorithms that are generalizable to many situations. Algorithms that are readable are easier to follow, test, and debug. *Algorithms control what recommendations a user may get on a music-streaming website, how a game responds to finger presses on a touchscreen, and how information is sent across the Internet. An algorithm that is generalizable to many situations can produce different outputs, based on a wide range of inputs. For example, an algorithm for a smart thermostat may control the temperature based on the time of day, how many people are at home, and current electricity consumption. The testing of an algorithm requires the use of inputs that reflect all possible conditions to evaluate its accuracy and robustness.* *Crosscutting Concepts: Human–Computer Interaction; Abstraction* *Connections Within Framework: 6–8. Data and Analysis.Inference and Models; 6–8.Computing Systems. Troubleshooting; 6–8.Data and Analysis. Visualization and Transformation*	People evaluate and select algorithms based on performance, reusability, and ease of implementation. Knowledge of common algorithms improves how people develop software, secure data, and store information. *Some algorithms may be easier to implement in a particular programming language, work faster, require less memory to store data, and be applicable in a wider variety of situations than other algorithms. Algorithms used to search and sort data are common in a variety of software applications. Encryption algorithms are used to secure data, and compression algorithms make data storage more efficient. At this level, analysis may involve simple calculations of steps. Analysis using sophisticated mathematical notation to classify algorithm performance, such as Big-O notation, is not expected.* *Crosscutting Concepts: Abstraction; Privacy and Security* *Connections Within Framework: 9–12. Data and Analysis. Storage; 9–12. Networks and the Internet.Cybersecurity*

VARIABLES

Overview: Computer programs store and manipulate data using variables. In early grades, students learn that different types of data, such as words, numbers, or pictures, can be used in different ways. As they progress, students learn about variables and ways to organize large collections of data into data structures of increasing complexity.

Information in the real world can be represented in computer programs. Programs store and manipulate data, such as numbers, words, colors, and images. The type of data determines the actions and attributes associated with it. *Different actions are available for different kinds of information. For example, sprites (character images) can be moved and turned, numbers can be added or subtracted, and pictures can be recolored or cropped.* ***Crosscutting Concept:** Abstraction* ***Connection Within Framework:** K–2. Data and Analysis.Storage*	Programming languages provide variables, which are used to store and modify data. The data type determines the values and operations that can be performed on that data. *Variables are the vehicle through which computer programs store different types of data. At this level, understanding how to use variables is sufficient, without a fuller understanding of the technical aspects of variables (such as identifiers and memory locations). Data types vary by programming language, but many have types for numbers and text. Examples of operations associated with those types are multiplying numbers and combining text. Some visual, blocks-based languages do not have explicitly declared types but still have certain operations that apply only to particular types of data in a program.* ***Crosscutting Concept:** Abstraction* ***Connection Within Framework:** 3–5. Data and Analysis. Storage*	Programmers create variables to store data values of selected types. A meaningful identifier is assigned to each variable to access and perform operations on the value by name. Variables enable the flexibility to represent different situations, process different sets of data, and produce varying outputs. *At this level, students deepen their understanding of variables, including when and how to declare and name new variables. A variable is like a container with a name, in which the contents may change, but the name (identifier) does not. The identifier makes keeping track of the data that is stored easier, especially if the data changes. Naming conventions for identifiers, and thoughtful choices of identifiers, improve program readability.* *The term variable is used differently in programming than the way it is commonly used in mathematics: a program variable refers to a location in which a value is stored, and the name used to access the value is called the identifier. A program variable is assigned a value, and that value may change throughout the execution of the program. Mathematicians typically do not make a distinction between a variable and the variable name. A mathematics variable often represents a set of values for which the statement containing the variable is true.* ***Crosscutting Concept:** Abstraction* ***Connection Within Framework:** 6–8. Data and Analysis.Storage*	Data structures are used to manage program complexity. Programmers choose data structures based on functionality, storage, and performance tradeoffs. *A list is a common type of data structure that is used to facilitate the efficient storage, ordering, and retrieval of values and various other operations on its contents. Tradeoffs are associated with choosing different types of lists. Knowledge of advanced data structures, such as stacks, queues, trees, and hash tables, is not expected. User-defined types and object-oriented programming are optional concepts at this level.* ***Crosscutting Concepts:** Abstraction; System Relationships* ***Connection Within Framework:** 6–8. Computing Systems.Hardware and Software*

CONTROL

Overview: Control structures specify the order in which instructions are executed within an algorithm or program. In early grades, students learn about sequential execution and simple control structures. As they progress, students expand their understanding to combinations of structures that support complex execution.

Computers follow precise sequences of instructions that automate tasks. Program execution can also be nonsequential by repeating patterns of instructions and using events to initiate instructions. *Computers follow instructions literally. Examples of sequences of instructions include steps for drawing a shape or moving a character across the screen. An event, such as the press of a button, can trigger an action. Simple loops can be used to repeat instructions. At this level, distinguishing different types of loops is not expected.* ***Crosscutting Concept:*** *Abstraction* ***Connections Within Framework:*** *K–2.Computing Systems.Devices; K–2.Computing Systems. Hardware and Software*	Control structures, including loops, event handlers, and conditionals, are used to specify the flow of execution. Conditionals selectively execute or skip instructions under different conditions. *Different types of loops are used to repeat instructions in multiple ways depending on the situation. Examples of events include mouse clicks, typing on the keyboard, and collisions between objects. Event handlers are sets of commands that are tied to specific events. Conditionals represent decisions and are composed of a Boolean condition that specifies actions based on whether the condition evaluates to true or false. Boolean logic and operators (e.g., AND, OR, NOT) can be used to specify the appropriate groups of instructions to execute under various conditions.* ***Crosscutting Concepts:*** *Abstraction; Communication and Coordination* ***Connection Within Framework:*** *K–2. Computing Systems.Devices*	Programmers select and combine control structures, such as loops, event handlers, and conditionals, to create more complex program behavior. *Conditional statements can have varying levels of complexity, including compound and nested conditionals. Compound conditionals combine two or more conditions in a logical relationship, and nesting conditionals within one another allows the result of one conditional to lead to another being evaluated. An example of a nested conditional structure is deciding what to do based on the weather outside. If it is sunny outside, I will further decide if I want to ride my bike or go running, but if it is not sunny outside, I will decide whether to read a book or watch TV. Different types of control structures can be combined with one another, such as loops and conditionals. Different types of programming languages implement control structures in different ways. For example, functional programming languages implement repetition using recursive function calls instead of loops. At this level, understanding implementation in multiple languages is not essential.* ***Crosscutting Concept:*** *Abstraction*	Programmers consider tradeoffs related to implementation, readability, and program performance when selecting and combining control structures. *Implementation includes the choice of programming language, which affects the time and effort required to create a program. Readability refers to how clear the program is to other programmers and can be improved through documentation. The discussion of performance is limited to a theoretical understanding of execution time and storage requirements; a quantitative analysis is not expected. Control structures at this level may include conditional statements, loops, event handlers, and recursion. Recursion is a control technique in which a procedure calls itself and is appropriate when problems can be expressed in terms of smaller versions of themselves. Recursion is an optional concept at this level.* ***Crosscutting Concepts:*** *Abstraction; System Relationships*

MODULARITY

Overview: Modularity involves breaking down tasks into simpler tasks and combining simple tasks to create something more complex. In early grades, students learn that algorithms and programs can be designed by breaking tasks into smaller parts and recombining existing solutions. As they progress, students learn about recognizing patterns to make use of general, reusable solutions for commonly occurring scenarios and clearly describing tasks in ways that are widely usable.

Complex tasks can be broken down into simpler instructions, some of which can be broken down even further. Likewise, instructions can be combined to accomplish complex tasks. *Decomposition is the act of breaking down tasks into simpler tasks. An example of decomposition is preparing for a party: it involves inviting guests, making food, and setting the table. These tasks can be broken down further. For example, setting the table involves laying a tablecloth, folding napkins, and placing utensils and plates on the table. Another example is breaking down the steps to draw a polygon.* *Composition, on the other hand, is the combination of smaller tasks into more complex tasks. To build a city, people build several houses, a school, a store, etc. To create a group art project, people can paint or draw their favorite ocean animal, then combine them to create an ecosystem.* ***Crosscutting Concept:** System Relationships* ***Connection Within Framework:** K–2. Computing Systems. Devices*	Programs can be broken down into smaller parts to facilitate their design, implementation, and review. Programs can also be created by incorporating smaller portions of programs that have already been created. *Decomposition facilitates aspects of program development, such as testing, by allowing people to focus on one piece at a time. Decomposition also enables different people to work on different parts at the same time. An example of 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 similar to instructions in other programs.* ***Crosscutting Concepts:** System Relationships; Abstraction*	Programs use procedures to organize code, hide implementation details, and make code easier to reuse. Procedures can be repurposed in new programs. Defining parameters for procedures can generalize behavior and increase reusability. *A procedure is a module (a group of instructions within a program) that performs a particular task. In this framework, procedure is used as a general term that may refer to an actual procedure or a method, function, or similar concept in other programming languages. Procedures are invoked to repeat groups of instructions. For example, a procedure, such as one to draw a circle, involves many instructions, but all of them can be invoked with one instruction, such as "drawCircle." Procedures that are defined with parameters are generalizable to many situations and will produce different outputs based on a wide range of inputs (arguments).* ***Crosscutting Concepts:** Abstraction; System Relationships*	Complex programs are designed as systems of interacting modules, each with a specific role, coordinating for a common overall purpose. These modules can be procedures within a program; combinations of data and procedures; or independent, but interrelated, programs. Modules allow for better management of complex tasks. *Software applications require a sophisticated approach to design that uses a systems perspective. For example, object-oriented programming decomposes programs into modules called objects that pair data with methods (variables with procedures). The focus at this level is understanding a program as a system with relationships between modules. The choice of implementation, such as programming language or paradigm, may vary.* ***Crosscutting Concepts:** System Relationships; Abstraction* ***Connection Within Framework:** 9–12.Computing Systems.Hardware and Software*

PROGRAM DEVELOPMENT

Overview: Programs are developed through a design process that is often repeated until the programmer is satisfied with the solution. In early grades, students learn how and why people develop programs. As they progress, students learn about the tradeoffs in program design associated with complex decisions involving user constraints, efficiency, ethics, and testing.

People develop programs collaboratively and for a purpose, such as expressing ideas or addressing problems.

People work together to plan, create, and test programs within a context that is relevant to the programmer and users. Programming is used as a tool to create products that reflect a wide range of interests, such as video games, interactive art projects, and digital stories.

Crosscutting Concept: Human–Computer Interaction

Connection Within Framework: 3–5. Impacts of Computing.Culture

People develop programs using an iterative process involving design, implementation, and review. Design often involves reusing existing code or remixing other programs within a community. People continuously review whether programs work as expected, and they fix, or debug, parts that do not. Repeating these steps enables people to refine and improve programs.

Design, implementation, and review can be further broken down into additional stages and may have different labels. The design stage occurs before writing code. This is a planning stage in which the programmers gather information about the problem and sketch out a solution. During the implementation stage, the planned design is expressed in a programming language (code) that can be made to run on a computing device. During the review stage, the design and implementation are checked for adherence to program requirements, correctness, and usability. This review could lead to changes in implementation and possibly design, which demonstrates the iterative nature of the process. A community is created by people who share and provide feedback on one another's creations.

Crosscutting Concepts: Human–Computer Interaction; System Relationships

Connection Within Framework: K–2. Computing Systems.Troubleshooting

People design meaningful solutions for others by defining a problem's criteria and constraints, carefully considering the diverse needs and wants of the community, and testing whether criteria and constraints were met.

Development teams that employ user-centered design create solutions that can have a large societal impact, such as an app that allows people with speech difficulties to translate hard-to-understand pronunciation into understandable language. Use cases and test cases are created and analyzed to better meet the needs of users and to evaluate whether criteria and constraints are met. An example of a design constraint is that mobile applications must be optimized for small screens and limited battery life.

Crosscutting Concepts: Human–Computer Interaction; Abstraction

Connection Within Framework: 3–5.Impacts of Computing.Culture

Diverse teams can develop programs with a broad impact through careful review and by drawing on the strengths of members in different roles. Design decisions often involve tradeoffs. The development of complex programs is aided by resources such as libraries and tools to edit and manage parts of the program. Systematic analysis is critical for identifying the effects of lingering bugs.

As programs grow more complex, the choice of resources that aid program development becomes increasingly important. These resources include libraries, integrated development environments, and debugging tools. Systematic analysis includes the testing of program performance and functionality, followed by end-user testing. A common tradeoff in program development is sometimes referred to as "Fast/Good/Cheap: Pick Two": one can develop software quickly, with high quality, or with little use of resources (for example, money or number of people), but the project manager may choose only two of the three criteria.

Crosscutting Concepts: Human–Computer Interaction; System Relationships; Abstraction

Connection Within Framework: 9–12. Computing Systems.Troubleshooting

Impacts of Computing

Overview: 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.

By the end of Grade 2:	By the end of Grade 5:	By the end of Grade 8:	By the end of Grade 12:
CULTURE Overview: Computing influences culture—including belief systems, language, relationships, technology, and institutions—and culture shapes how people engage with and access computing. In early grades, students learn how computing can be helpful and harmful. As they progress, students learn about tradeoffs associated with computing and potential future impacts of computing on global societies.			
Computing technology has positively and negatively changed the way people live and work. Computing devices can be used for entertainment and as productivity tools, and they can affect relationships and lifestyles. *Computing devices, such as fitness trackers, can motivate a more active lifestyle by monitoring physical activity. On the other hand, passively consuming media from computing devices may lead to a more sedentary lifestyle. In the past, the most popular form of communication was to send mail via the postal service. Now, more people send emails or text messages.* ***Crosscutting Concept:*** *Human–Computer Interaction* ***Connection Within Framework:*** *K–2.Data and Analysis.Inference and Models*	The development and modification of computing technology is driven by people's needs and wants and can affect groups differently. Computing technologies influence, and are influenced by, cultural practices. *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). Increased Internet access and speed have allowed people to share cultural information but have also affected the practice of traditional cultural customs.* ***Crosscutting Concepts:*** *Human–Computer Interaction; System Relationships* ***Connections Within Framework:*** *K–2. Algorithms and Programming. Program Development; 6–8.Computing Systems. Devices; 6–8.Algorithms and Programming.Program Development*	Advancements in computing technology change people's everyday activities. Society is faced with tradeoffs due to the increasing globalization and automation that computing brings. *The effects of globalization, such as the sharing of information and cultural practices and the resulting cultural homogeneity, are increasingly possible because of computing. Globalization, coupled with the automation of the production of goods, allows access to labor that is less expensive and creates jobs that can easily move across national boundaries. Online piracy has increased because of information access that traverses national boundaries and varying legal systems.* ***Crosscutting Concepts:*** *Human–Computer Interaction; System Relationships* ***Connection Within Framework:*** *6–8. Data and Analysis.Inference and Models*	The design and use of computing technologies and artifacts can improve, worsen, or maintain inequitable access to information and opportunities. *While many people have direct access to computing throughout their day, many others are still underserved or simply do not have access. Some of these challenges are related to the design of computing technologies, as in the case of technologies that are difficult for senior citizens and people with physical disabilities to use. Other equity deficits, such as minimal exposure to computing, access to education, and training opportunities, are related to larger, systemic problems in society.* ***Crosscutting Concepts:*** *Human–Computer Interaction; System Relationships* ***Connection Within Framework:*** *9–12. Computing Systems.Devices*

SOCIAL INTERACTIONS

Overview: Computing can support new ways of connecting people, communicating information, and expressing ideas. In early grades, students learn that computing can connect people and support interpersonal communication. As they progress, students learn how the social nature of computing affects institutions and careers in various sectors.

Computing has positively and negatively changed the way people communicate. People can have access to information and each other instantly, anywhere, and at any time, but they are at the risk of cyberbullying and reduced privacy.

Online communication facilitates positive interactions, such as sharing ideas with many people, but the public and anonymous nature of online communication also allows intimidating and inappropriate behavior in the form of cyberbullying. Privacy should be considered when posting information online; such information can persist for a long time and be accessed by others, even unintended viewers.

Crosscutting Concepts: Human–Computer Interaction; Privacy and Security

Connections Within Framework: K–2. Data and Analysis.Storage; K–2.Data and Analysis.Visualization and Transformation

Computing technology allows for local and global collaboration. By facilitating communication and innovation, computing influences many social institutions such as family, education, religion, and the economy.

People can work in different places and at different times to collaborate and share ideas when they use technologies that reach across the globe. These social interactions affect how local and global groups interact with each other, and alternatively, these interactions can change the nature of groups. For example, a class can discuss ideas in the same school or in another nation through interactive webinars.

Crosscutting Concepts: System Relationships; Human–Computer Interaction

Connection Within Framework: K–2. Networks and the Internet.Network Communication and Organization

People can organize and engage around issues and topics of interest through various communication platforms enabled by computing, such as social networks and media outlets. These interactions allow issues to be examined using multiple viewpoints from a diverse audience.

Social networks can play a large role in social and political movements by allowing individuals to share ideas and opinions about common issues while engaging with those who have different opinions. Computing provides a rich environment for discourse but may result in people considering very limited viewpoints from a limited audience.

Crosscutting Concepts: System Relationships; Human–Computer Interaction

Connections Within Framework: 3–5. Data and Analysis.Visualization and Transformation; 9–12.Data and Analysis. Visualization and Transformation

Many aspects of society, especially careers, have been affected by the degree of communication afforded by computing. The increased connectivity between people in different cultures and in different career fields has changed the nature and content of many careers.

Careers have evolved, and new careers have emerged. For example, social media managers take advantage of social media platforms to guide the presence of a product or company and increase interaction with their audience. Global connectivity has also changed how teams in different fields, such as computer science and biology, work together. For example, the online genetic database made available by the Human Genome Project, the algorithms required to analyze the data, and the ability for scientists around the world to share information have accelerated the pace of medical discoveries and led to the new field of computational biology.

Crosscutting Concepts: System Relationships; Human–Computer Interaction

Connection Within Framework: 9–12. Networks and the Internet.Network Communication and Organization

SAFETY, LAW, AND ETHICS

Overview: Legal and ethical considerations of using computing devices influence behaviors that can affect the safety and security of individuals. In early grades, students learn the fundamentals of digital citizenship and appropriate use of digital media. As they progress, students learn about the legal and ethical issues that shape computing practices.

People use computing technology in ways that can help or hurt themselves or others. Harmful behaviors, such as sharing private information and interacting with strangers, should be recognized and avoided.

Using computers comes with a level of responsibility, such as not sharing login information, keeping passwords private, and logging off when finished. Rules guiding interactions in the world, such as "stranger danger," apply to online environments as well.

Crosscutting Concept: Privacy and Security

Connection Within Framework: K–2. Networks and the Internet.Cybersecurity

Ethical complications arise from the opportunities provided by computing. The ease of sending and receiving copies of media on the Internet, such as video, photos, and music, creates the opportunity for unauthorized use, such as online piracy, and disregard of copyrights, such as lack of attribution.

Online piracy, the illegal copying of materials, is facilitated by the ability to make identical-quality copies of digital media with little effort. Other topics related to copyright are plagiarism, fair use, and properly citing online sources. Knowledge of specific copyright laws is not an expectation at this level.

Crosscutting Concepts: System Relationships; Privacy and Security

Connection Within Framework: 3–5. Networks and the Internet.Cybersecurity

There are tradeoffs between allowing information to be public and keeping information private and secure. People can be tricked into revealing personal information when more public information is available about them online.

Social engineering is based on tricking people into breaking security procedures and can be thwarted by being aware of various kinds of attacks, such as emails with false information and phishing. Security attacks often start with personal information that is publicly available online. All users should be aware of the personal information, especially financial information, that is stored on the websites they use. Protecting personal online information requires authentication measures that can often make it harder for authorized users to access information.

Crosscutting Concepts: Privacy and Security; Communication and Coordination

Connection Within Framework: 6–8. Networks and the Internet.Cybersecurity

Laws govern many aspects of computing, such as privacy, data, property, information, and identity. These laws can have beneficial and harmful effects, such as expediting or delaying advancements in computing and protecting or infringing upon people's rights. International differences in laws and ethics have implications for computing.

Legal issues in computing, such as those related to the use of the Internet, cover many areas of law, reflect an evolving technological field, and can involve tradeoffs. For examples, laws that mandate the blocking of some file-sharing websites may reduce online piracy but can restrict the right to freedom of information. Firewalls can be used to block harmful viruses and malware but can also be used for media censorship. Access to certain websites, like social networking sites, may vary depending on a nation's laws and may be blocked for political purposes.

Crosscutting Concepts: System Relationships; Privacy and Security; Abstraction

Connection Within Framework: 9–12. Data and Analysis.Collection