

Computer Science Implementation Framework

Grade 4

2025

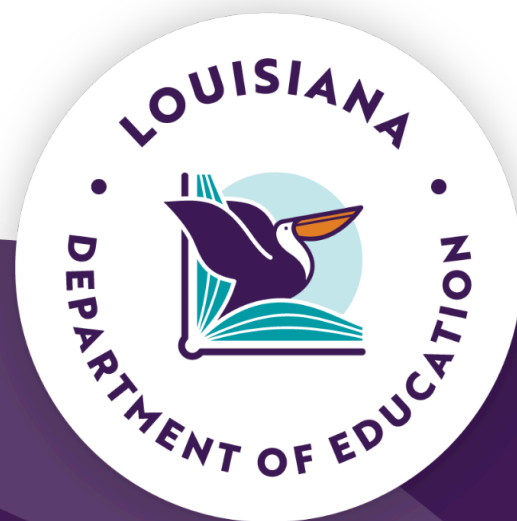


Table of Contents

Background	3
Finding Connections to Core Content Areas	4
Components of the Computer Science Implementation Framework	5
Core Concept 1: Computing Systems (CS)	7
Complexity Statements	8
Thinking Across Disciplines in Grade 4	10
Core Concept 2: Networks and the Internet (NI)	14
Complexity Statements	15
Thinking Across Disciplines in Grade 4	17
Core Concept 3: Data and Analysis (DA)	20
Complexity Statements	22
Thinking Across Disciplines in Grade 4	28
Core Concept 4: Algorithms and Programming (AP)	34
Complexity Statements	36
Thinking Across Disciplines in Grade 4	44
Core Concept 5: Impacts of Computing (IC)	51
Complexity Statements	52
Thinking Across Disciplines in Grade 4	55

Background

[Louisiana Act 541 \(2022\)](#), the Computer Science Education Act, aimed to establish a comprehensive and seamless statewide computer science education program across all educational levels to meet the demands of a dynamic economy. To achieve this goal, Act 541 created the Computer Science Education Advisory Commission (CSEAC). CSEAC was responsible for advising the State Board of Elementary and Secondary Education (BESE) through the state Department of Education (LDOE) to develop and implement a state action plan for delivering education in computer science in all public schools.

The first key action within the [Louisiana K-12 Computer Science Education Plan](#) calls for establishing content standards for computer science. From May to August 2024, the K-12 Computer Standards Writing Committee drafted student-centered and developmentally appropriate computer science standards within defined grade bands: elementary (K-5), middle school (6-8), and high school (9-12). These standards were carefully designed to reflect the increasing complexity and depth of understanding expected as students progress through each educational stage, with a focus on analytical and critical thinking skills, digital literacy, digital citizenship, and technology skills. In October 2024, BESE approved the [K-12 Louisiana Student Standards for Computer Science](#). These standards now serve as a guiding framework to support high-quality computer science instruction in Louisiana.

To support standards implementation, the [Louisiana K-12 Computer Science Education Plan](#) also calls for the development of computer science frameworks for standards implementation, which show the overlap with existing content standards.

Purpose of the Frameworks

The frameworks support Key Action 1 of the [Louisiana K-12 Computer Science Education Plan](#) by providing guidance for integrating the [K-12 Louisiana Student Standards for Computer Science](#) into existing instruction, emphasizing the connections between computer science, math, and science. By utilizing the frameworks, teachers will equip students with essential analytical thinking, digital literacy, and technology skills needed for success in society and future career opportunities.

Connections between Computer Science, Mathematics, and Science

A key feature of the Computer Science Implementation Frameworks is highlighting the connections between the [K-12 Louisiana Student Standards for Computer Science](#) and mathematics and science instruction. These three content areas are closely related. Many computer science concepts, such as sequencing, patterns, and logic, align with mathematical reasoning and scientific inquiry. Computational thinking strategies, such as decomposition and algorithmic thinking, support problem-solving approaches in math and science. Integrating computer science within mathematics and science helps students see connections between the content areas, encouraging students to view technology as a tool for solving real-world problems in a variety of professional fields.

Finding Connections to Core Content Areas

In elementary education, computer science implementation emphasizes integrating computer science across content areas. Computer science standards provide the foundation for teachers to identify and incorporate cross-disciplinary computer science content opportunities within lessons and content skills.

Unpacking the Standards

Computer science standards are essential for effective computer science implementation. When considering which computer science standards best fit a lesson, summarize the standard's ultimate purpose or goals by asking, “What should students accomplish by the end of grade 5 to master this standard?” Identifying the underlying skill or content required to master the standard helps in determining its best fit within a specific content area.

Finding Connections

Connecting standards to the real world makes standards more meaningful and relevant to students. Making connections between computer science standards and content or skills in another content standards demonstrates the real-world applications of computer science. Connections may be found between computer science core practices and practices of another content area. Computer science skills also reinforce skills emphasized in other content areas. For a full explanation of the core concepts and core practices of the computer science standards, see the [K-12 Louisiana Student Standards for Computer Science](#).

Computational Thinking Skills

Computational thinking involves breaking down complex problems into smaller, manageable parts (decomposition), recognizing patterns (pattern recognition), creating step-by-step solutions (algorithms), and generalizing solutions to other problems (abstraction). These skills are not exclusive to computer science; they are applicable in other content areas including math and science. By understanding computational thinking, teachers can identify opportunities to apply these problem-solving strategies in multiple contexts.

Computational Thinking Practice	In Mathematics	In Science
Abstraction	Represent concepts using visualizations such as graphs, charts, or diagrams.	Create simplified simulations of complex scientific phenomena.
Algorithmic Thinking	Create step-by-step instructions for solving problems or geometric constructions.	Create models of scientific processes like the water cycle or food chain.
Decomposition	Break down multi-step word problems into smaller, manageable parts.	Break down complex investigations into smaller, testable hypotheses.
Pattern Recognition	Identify number patterns, geometric patterns, or patterns in data sets.	Identify patterns in data, such as weather, animal behavior, or plant growth.

Components of the Computer Science Implementation Framework

Complexity Statements

Complexity statements clarify the increasing depth and rigor of learning expectations within a content standard as students progress through each grade level. These statements break down each standard into a vertical progression of learning within the grade band and ensure an age-appropriate level of knowledge for each standard. Standards and complexity statements allow flexibility and guidance to implement instruction at each grade band and each grade level, depending on a particular school's or system's implementation plan.

Complexity statements are foundational for effectively integrating computer science by providing a clear, grade-level-specific model for instruction. By outlining the increasing depth and rigor of learning expectations across grade levels, these statements ensure computer science concepts are introduced and reinforced in a developmentally appropriate manner, preventing both over- and under-challenging students. This clarity in learning goals empowers teachers to create integrated lessons that are both effective and appropriately challenging, ultimately fostering a deeper, more interconnected understanding across all disciplines.

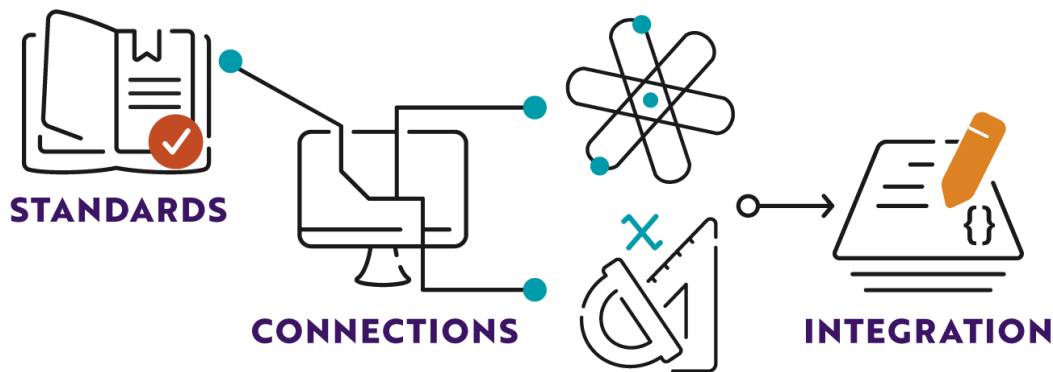
Example

E.NI.1A. Explain how networks connect computers to other computing systems and the internet.	
Grade	Complexity Statement
K	Define what a network is and how it is used to connect to other people and places around the world.
1	Compare and contrast wireless versus plugged-in networks.
2	Model and describe how computers break down data to share on networks.
3	Explain how data is transmitted in packets.
4	Using a model, demonstrate how information is shared within a network.
5	Diagram and summarize how data can be shared between multiple networks and users via the Internet.

Thinking Across Disciplines

Thinking Across Disciplines (T.A.D.) is a systemic approach guiding the integration of computer science concepts and practices into other content areas.

THINKING ACROSS DISCIPLINES (T.A.D.)



The T.A.D. Process

Initially, teachers gain a thorough understanding of the computer science standards for their grade band and the overarching core practices. Next, teachers identify meaningful connections between the standards and practices of computer science and those of other content areas, such as math and science. Finally, teachers create or modify existing lessons and activities that blend computer science concepts with existing instruction.

Benefits of T.A.D.

This interdisciplinary approach makes learning more relevant and engaging for students by demonstrating the practical applications of computer science in real-world contexts. It also reinforces critical thinking, problem-solving, and computational thinking skills, which transfer across multiple disciplines. The computer science implementation frameworks contain examples of the T.A.D. process as a starting point for teachers to discover their own connections between computer science and other content areas.

Core Concept 1: Computing Systems (CS)

Overview

Students interact with a wide variety of computers each day. Computers are devices that can collect, store, analyze, and act upon information in ways that can positively and negatively affect human capabilities. Humans (users) operate, program, and maintain the physical components (or hardware) and instructions (or software) that make up a computer. The hardware, software, and users are collectively called computing systems. Users leverage their understanding of hardware and software to resolve problems within computing systems in a process called troubleshooting.

Elementary Standards for Core Concept 1: Computing Systems

Subconcepts and Core Practices	Elementary (K-5)
Subconcept 1: Hardware and Software <i>Core Practices: Collaborating Around Computing; Recognizing and Defining Computational Problems; Communicating about Computing</i>	E.CS.1A. Identify and select the appropriate hardware to complete computing tasks.
	E.CS.1B. Identify and select the appropriate software to complete computing tasks.
	E.CS.1C. Evaluate hardware and software types to meet users' needs in completing various computing tasks.
Subconcept 2: Troubleshooting <i>Core Practices: Collaborating Around Computing; Recognizing and Defining Computational Problems; Testing and Refining Computation Artifacts; Communicating about Computing</i>	E.CS.2A. Propose potential ways to address computing problems using appropriate hardware or software.

Complexity Statements

Core Concept 1: Computing Systems

Complexity statements for the standards included in **Core Concept 1: Computing Systems** break down each standard into manageable, age-appropriate components. This breakdown ensures that instruction fits the developmental needs of students at each grade level. For instance, kindergarten focuses on describing the basic uses of hardware and software, but by grade 5 students justify and apply knowledge of hardware and software to navigate complex scenarios. Grade 5 students might analyze a scenario where a computer program runs slowly and propose a plan to troubleshoot whether the issue stems from the hardware (like insufficient memory) or the software (like too many applications running).

E.CS.1A. Identify and select the appropriate hardware to complete computing tasks.	
Grade	Complexity Statement
K	Describe the ways people use hardware to perform computing tasks. (e.g., keyboard, mouse, monitors, laptops, tablets)
1	Explain how common hardware pieces function.
2	Compare and contrast the functions and uses of various hardware.
3	Identify the appropriate hardware to complete a given task.
4	Analyze the capabilities and limitations of hardware for a particular use in a given scenario.
5	Justify and use the proper hardware to complete a given task.

E.CS.1B. Identify and select the appropriate software to complete computing tasks.	
Grade	Complexity Statement
K	Describe the ways people use software to perform computing tasks. (e.g., apps to gather information, access the internet, etc.)
1	Explain the function of common software components.
2	Compare and contrast the functions and uses of various software.
3	Identify the appropriate software to complete a given task.
4	Analyze the capabilities and limitations of software for a particular use in a given scenario.

5	Justify and use the proper software to complete a given task.
---	---

E.CS.1C. Evaluate hardware and software types to meet users' needs in completing various computing tasks.

Grade	Complexity Statement
K	Identify ways that computing devices help users accomplish simple tasks.
1	Identify the basic hardware and software used to meet users' needs for common tasks.
2	When given a scenario, describe the function of the hardware and software in accomplishing a specific task.
3	Describe the limitations of certain computing devices in given scenarios.
4	When given a scenario, determine the most appropriate hardware and software to accomplish a task.
5	In a given scenario, evaluate a user's unique needs and recommend the most appropriate hardware and software to meet their needs.

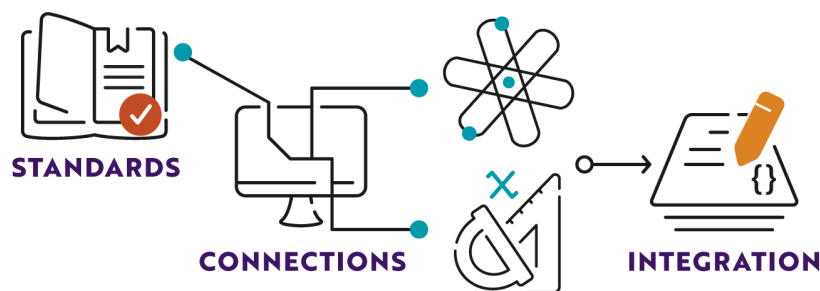
E.CS.2A. Propose potential ways to address computing problems using appropriate hardware or software.

Grade	Complexity Statement
K	Identify a basic hardware problem.
1	Identify a basic software problem.
2	Differentiate between a hardware problem and a software problem using observations.
3	Describe and hypothesize causes for a basic hardware problem and a basic software problem.
4	When given an error message, describe the appropriate user response to correct a basic problem with hardware or software.
5	When given various error messages, propose and test a plan to address basic computing problems using hardware or software.

Thinking Across Disciplines in Grade 4

Core Concept 1: Computing Systems

THINKING ACROSS DISCIPLINES (T.A.D.)



The standards in **Core Concept 1: Computing Systems** build student understanding of the hardware and software that make up computers and guide the intentional and effective use of these tools.

- **Problem-solving** is a skill emphasized across many content areas. Students need to be able to understand and address problems of various kinds to strengthen analytical abilities. Troubleshooting issues that emerge when using computers provides a new context for students to practice problem-solving skills.
- Evaluating hardware and software applications and selecting the best tools to meet specific user needs requires **critical thinking**. Strong critical thinking skills allow students to draw informed conclusions, make reliable recommendations, and justify their decisions.
- Understanding the purpose and function of hardware and software supports effective technology integration from early stages of education. Developing **competence and familiarity with technology** equips students for success in both the workforce and higher education. A solid foundation in hardware, software, and troubleshooting deepens students' knowledge of technology tools and their applications.

T.A.D. Examples

Standard	<p>E.CS.1A. Identify and select the appropriate hardware to complete computing tasks.</p> <p>Complexity Statement: Analyze the capabilities and limitations of hardware for a particular use in a given scenario.</p> <p>Core Practices: Collaborating Around Computing; Recognizing and Defining Computation Problems; Communicating about Computing</p>
Connections	<p>Mathematics 4.MD.C.6 Measure angles in whole-number degrees using a protractor. 4.MP.5 Use appropriate tools strategically</p> <p>Science 4-PS3-2 Make observations to provide evidence that energy can be transferred from place to place by sound, light, heat, and electric currents.</p>
Integration	<p>Just as students use a protractor to measure angles accurately in math, they learn to identify and select appropriate computing hardware for specific tasks. For instance, when measuring angles digitally, a student might choose a tablet with a stylus over a standard mouse because the stylus offers greater precision, demonstrating strategic tool selection across disciplines.</p> <p>As students investigate how energy transfers through sound, light, heat, or electricity, they also consider which hardware tools best support their observations. For example, when exploring sound energy, a student may choose to use a microphone connected to a computer to capture and analyze sound waves, recognizing the microphone as the appropriate hardware for the task.</p>

Standard	<p>E.CS.1B. Identify and select the appropriate software to complete computing tasks.</p> <p>Complexity Statement: Analyze the capabilities and limitations of software for a particular use in a given scenario.</p> <p>Core Practices: Collaborating Around Computing; Recognizing and Defining Computation Problems; Communicating about Computing</p>
Connections	<p>Mathematics 4.MD.C.6 Measure angles in whole-number degrees using a protractor.</p> <p>Science 4-PS 3-2 Make observations to provide evidence that energy can be transferred from place to place by sound, light, heat,</p>

Integration	Just as students select a protractor to measure angles accurately, they also learn to choose appropriate software to support their work. For instance, a student collecting and analyzing angle measurements in mathematics or experiment results in science, might use a digital spreadsheet instead of paper to organize data and perform calculations more efficiently, demonstrating thoughtful software selection.
--------------------	---

Standard	<p>E.CS.1C. Evaluate hardware and software types to meet users’ needs in completing various computing tasks.</p> <p>Complexity Statement: When given a scenario, determine the most appropriate hardware and software to accomplish a task.</p> <p>Core Practices: Collaborating Around Computing; Recognizing and Defining Computation Problems; Communicating about Computing</p>
Connections	<p>Mathematics 4.MP.5 Use appropriate tools strategically.</p> <p>Science 4-ESS3-1 Earth and Human Activity: Obtain and combine information to describe that energy and fuels are derived from renewable and non-renewable resources and how their uses affect the environment.</p>
Integration	When solving mathematical problems, collecting data, or presenting research, students must evaluate which tools best meet the needs of the task. For example, a student researching renewable and non-renewable energy sources may choose presentation software over a physical project board to more easily organize, update, and share their findings — demonstrating strategic tool use and thoughtful evaluation of software options.

Standard	<p>E.CS.2A. Propose potential ways to address computing problems using appropriate hardware or software.</p> <p>Complexity Statement: When given an error message, describe the appropriate user response to correct a basic problem with hardware or software.</p> <p>Core Practices: Collaborating Around Computing; Recognizing and Defining Computational Problems; Communicating about Computing</p>
Connections	<p>Mathematics 4.MP.1 Make sense of problems and persevere in solving them.</p>

Integration

As students learn to make sense of problems and persevere in solving them, they also consider how different tools can support their solutions. For example, when encountering a computing problem, a student may propose using a specific app or device that best fits the task, showing how thoughtful problem-solving involves evaluating and selecting appropriate hardware or software.

Core Concept 2: Networks and the Internet (NI)

Overview

Computing systems typically do not operate in isolation. Networks connect computers to share information and resources and are integral to computer and data science. Networks enable critical communication for the computing systems that drive our economy and career sectors. The increased level of connectivity brought about by the internet provides fast and secure communication that facilitates innovation.

Elementary Standards for Core Concept 2: Networks and the Internet

Subconcepts and Core Practices	Elementary (K-5)
Subconcept 1: Hardware and Network Communication <i>Core Practices: Fostering responsible cyber citizenship; Collaborating around computing; Recognizing and defining computational problems; Developing and using abstractions; Creating computational artifacts; Testing and refining computational artifacts; Communicating about computing</i>	E.NI.1A. Explain how networks connect computers to other computing systems and the internet.
Subconcept 2: Cybersecurity <i>Core Practices: Fostering responsible cyber citizenship; Collaborating around computing; Recognizing and defining computational problems; Developing and using abstractions; Creating computational artifacts; Testing and refining computational artifacts; Communicating about computing</i>	E.NI.2A. Describe personally identifiable information (PII) and identify practices for when and where sharing PII is appropriate.
	E.NI.2B. Identify ways to maintain data security when using networks.

Complexity Statements

Core Concept 2: Networks and the Internet

Complexity statements for the standards included in **Core Concept 2: Networks and the Internet** break down each standard into manageable, age-appropriate components. This breakdown ensures that instruction fits the developmental needs of students at each grade level. For instance, kindergarten focuses on defining basic network concepts and distinguishing between public and private information. By grade 5, students diagram network interactions and analyze the complexities of data sharing across networks, including the importance of responsible online behavior and security practices.

E.NI.1A. Explain how networks connect computers to other computing systems and the internet.	
Grade	Complexity Statement
K	Define what a network is and how it is used to connect to other people and places around the world.
1	Compare and contrast wireless versus plugged-in networks.
2	Model and describe how computers break down data to share on networks.
3	Explain how data is transmitted in packets.
4	Using a model, demonstrate how information is shared within a network.
5	Diagram and summarize how data can be shared between multiple networks and users via the Internet.

E.NI.2A. Describe personally identifiable information (PII) and identify practices for when and where sharing PII is appropriate.	
Grade	Complexity Statement
K	Distinguish between public and private information.
1	Define personally identifiable information (PII).
2	Evaluate the risks and benefits of sharing PII.
3	Evaluate mechanisms of password generation and best practices for password management.
4	Describe the role of authentication and authorization.

5	Identify and determine the appropriate resources and individuals to notify if PII is inappropriately shared.
---	--

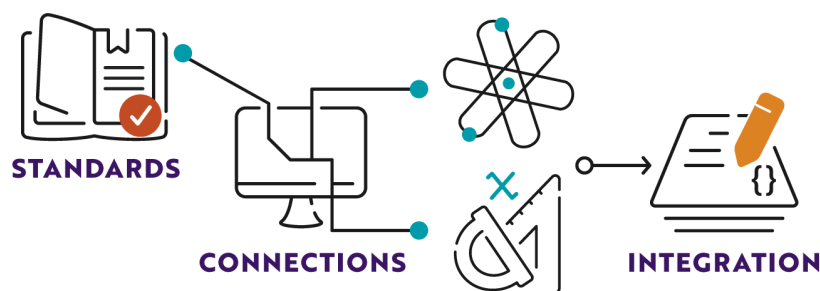
E.NI.2B. Identify ways to maintain data security when using networks.

Grade	Complexity Statement
K	Describe how usernames and passwords protect information and network access.
1	Distinguish between the six top-level domains and describe what information they share (.com, .net, .edu, .org, .gov, and .mil).
2	Given scenarios, identify and classify a network as secure or insecure.
3	Identify and assess when an email is a phishing attempt within the examples.
4	Explain what antivirus software is and how it protects a user, network, and data sharing.
5	Explain the ways a student can protect their home and school networks.

Thinking Across Disciplines in Grade 4

Core Concept 2: Networks and the Internet

THINKING ACROSS DISCIPLINES (T.A.D.)



The standards in **Core Concept 2: Networks and the Internet** develop an understanding of how networks connect computers to the internet and other computing systems, along with safe practices for sharing personally identifiable information. These skills are tied to concepts teachers already address in their classrooms.

- **Obtaining, evaluating, and communicating information** is an interdisciplinary skill. Understanding how information is transmitted through a network safely strengthens the ability to obtain, evaluate, and communicate information electronically.
- Explaining safe practices for sharing personally identifiable information on a network requires **critical thinking** to evaluate what information is appropriate to share, when to share it, and how to do so responsibly. This decision-making process supports the development of responsible technology use.
- Maintaining data security when using networks involves recognizing common risks and applying strategies to protect sensitive information. Practicing data protection builds **digital responsibility** and supports safer online interactions.

T.A.D. Examples

Standard	<p>E.NI.1A. Explain how networks connect computers to other computing systems and the internet.</p> <p>Complexity Statement: Using a model, demonstrate how information is shared within a network.</p> <p>Core Practices: Fostering responsible cyber citizenship; Collaborating around computing; Recognizing and defining computational problems; Developing and using abstractions; Creating computational artifacts; Testing and refining computational artifacts; Communicating about computing</p>
Connections	<p>Science 4-PS3-2 Make observations to provide evidence that energy can be transferred from place to place by sound, light, heat, and electric currents.</p> <p>Science 4-PS3-1 Use evidence to construct an explanation relating the speed of an object to the energy of that object.</p>
Integration	<p>As students explore how energy transfers through light, heat, sound, or electricity, they can model how networks transfer information using similar principles. For example, students might send messages through a simple circuit acting as a network, observing how energy flows from a sender to a receiver. They can then compare this to how data moves through computing networks — faster speeds requiring more energy — drawing parallels between physical energy transfer and digital communication systems.</p>
Standard	<p>E.NI.2A. Describe personally identifiable information (PII) and identify practices for when and where sharing PII is appropriate.</p> <p>Complexity Statement: Describe the role of authentication and authorization.</p> <p>Core Practices: Fostering responsible cyber citizenship; Collaborating around computing; Recognizing and defining computational problems; Developing and using abstractions; Creating computational artifacts; Testing and refining computational artifacts; Communicating about computing</p>
Connections	<p>Mathematics 4.OA.B.4A Using Whole number in the range 1-100, find all factor pairs for a given whole number.</p>

Integration	When finding factor pairs for whole numbers, students may complete and share a digital worksheet using a login system. This process involves authentication (logging in with a username and password) and authorization (granting a classmate access to review their work). Through this activity, students experience how protecting personally identifiable information and managing access are part of responsible digital practices.
Standard	<p>E.NI.2B. Identify ways to maintain data security when using networks.</p> <p>Complexity Statement: Explain what antivirus software is and how it protects a user, network, and data sharing.</p> <p>Core Practices: Fostering responsible cyber citizenship; Collaborating around computing; Recognizing and defining computational problems; Developing and using abstractions; Creating computational artifacts; Testing and refining computational artifacts; Communicating about computing</p>
Connections	<p>Science</p> <p>Science & Engineering Practices: Asking questions and defining problems</p>
Integration	Just as students ask questions about how to prevent the spread of germs, they can explore how antivirus software protects computers and networks from digital threats. While using technology, students can discuss how tools like antivirus programs help maintain data security – drawing parallels between protecting human health and protecting digital information.

Core Concept 3: Data and Analysis (DA)

Overview

Computing systems function by processing and storing data. Data is abundant due to the growing number of connected devices worldwide. As the volume of data expands, so does the demand for accurate and efficient data analysis methods. Data science is the cross-disciplinary use of data to inform decision-making, test hypotheses, predict trends, and develop precise models that drive innovation across industries.

Elementary Standards for Core Concept 3: Data and Analysis

Subconcepts and Core Practices	Elementary (K-5)
Subconcept 1: Data Representation <i>Core Practices: Collaborating around computing; Recognizing and defining computational problems; Creating computational artifacts</i>	E.DA.1A. Organize and present data visually to highlight relationships and support claims.
	E.DA.1B. Classify types of data and describe the attributes used to sort data.
Subconcept 2: Data Collection <i>Core Practices: Fostering responsible cyber citizenship; Collaborating around computing; Recognizing and defining computational problems; Developing and using abstractions; Creating computational artifacts; Testing and refining computational artifacts; Communicating about computing</i>	E.DA.2A. Select the appropriate data collection tool and technique to gather data to support a claim or communicate information.
	E.DA.2B. Describe and collect data utilizing the appropriate units of measure and discuss how data format impacts a computing system.
Subconcept 3: Data Storage <i>Core Practices: Fostering responsible cyber citizenship; Collaborating around computing; Recognizing and defining computational problems; Developing and using abstractions; Creating computational artifacts; Testing and refining computational artifacts; Communicating about computing</i>	E.DA.3A. Compare and contrast ways to store data using technology.
	E.DA.3B. Explain how to save and name data, search for data, retrieve data, modify data, and delete data using a computing device.
Subconcept 4: Visualizations and Transformations <i>Core Practices: Creating computational artifacts; Communicating about computing</i>	E.DA.4A. Organize and present data visually in at least three ways to highlight relationships and evaluate a claim.

	E.DA.4B. Evaluate data quality and clean data when indicated using the criteria of validity, accuracy, completeness, consistency, and uniformity.
Subconcept 5: Inference and Models <i>Core Practices: Recognizing and defining computational problems; Developing and using abstractions; Creating computational artifacts; Testing and refining computational artifacts; Communicating about computing</i>	E.DA.5A. Utilize data to create models, answer investigative questions, and make predictions.
	E.DA.5B. Analyze data for patterns and relationships.

Complexity Statements

Core Concept 3: Data and Analysis

Complexity statements for the standards included in **Core Concept 3: Data and Analysis** break down each standard into manageable, age-appropriate components. This breakdown ensures that instruction fits the developmental needs of students at each grade level. For instance, in kindergarten, the focus is on classifying and counting data into simple categories, while by grade 5, students analyze multiple types of data to support claims, compare different datasets for accuracy, and generate appropriate computational visualizations to explain their findings.

E.DA.1A. Organize and present data visually to highlight relationships and support claims.	
Grade	Complexity Statement
K	Classify data into given categories based on attributes; count the number of objects in each category and sort the categories by count. Enter the counts into a data table on the computer with support.
1	Computationally organize, represent, and interpret data with up to three categories. (e.g., ask and answer questions about the total number of data points, how many are in each category, and how many more or less are in one category or another with support.)
2	Computationally create bar graphs (with single-unit scale) to represent a data set with up to four categories. Solve put-together, take apart, and compare problems using information presented in the bar graphs with support.
3	Computationally create scaled bar graphs representing a data set with more than four categories. Solve one- and two-step “how many more” and “how many less” problems using information presented in the scaled bar graphs. (e.g., create a bar graph in which each square in the bar might represent five alligators in real life.)
4	Compare computational generated graphs (e.g., bar and pie) to their corresponding data tables to validate their accuracy.
5	Computationally generate line graphs from data that is collected and entered into data tables to support claims.

E.DA.1B. Classify types of data and describe the attributes used to sort data.	
Grade	Complexity Statement
K	Explain what the term data means and give examples.
1	Select the appropriate type of data to support or disprove a claim.

E.DA.1B. Classify types of data and describe the attributes used to sort data.

Grade	Complexity Statement
2	Define and describe digital, non-digital, numerical, text-based, audio, visual, and video data types.
3	Evaluate the tradeoffs of using one data type over another to support or refute a given claim.
4	Analyze two or more types of data to support or disprove a claim.
5	Identify specific types of data, such as metadata, and explain what the information means, including its application.

E.DA.2A. Select the appropriate data collection tool and technique to gather data to support a claim or communicate information.

Grade	Complexity Statement
K	Collect and organize data computationally in a table or chart with support.
1	Identify and use the appropriate digital tool (e.g., thermometer, scale, probe) to collect various data computationally in tables or charts with support.
2	Collect and compile data computationally to evaluate a claim's validity.
3	Create and implement practices for collecting valid and accurate data using a data-gathering tool.
4	Compare and contrast various data sets collected for accuracy and describe potential sources of instrumentation error.
5	Compare and contrast data sets collected from two or more instruments and use the data to support a claim.

E.DA.2B. Describe and collect data utilizing the appropriate units of measure and discuss how data format impacts a computing system.

Grade	Complexity Statement
K	Define and describe various measurable attributes of objects (e.g., length, weight) and practice comparing two or more objects with a common attribute. Computationally collect the data (e.g., "more of," "less of," or "the same amount of") in a table with support.

E.DA.2B. Describe and collect data utilizing the appropriate units of measure and discuss how data format impacts a computing system.

Grade	Complexity Statement
1	Generate measurement data by measuring the lengths of various objects, then record the data computationally with units with support.
2	Recognize the unit of measure's impact on data input for a computer. Computationally sort data into table sets based on their units of measure (e.g., inches, meters, centimeters, feet, dollars, and cents) from various sources and tools.
3	Describe how user errors in data entry can change or alter the computing system outputs with examples of impacts on real-world business problems.
4	Collect data in measurements that require fractions (e.g., $\frac{1}{2}$, $\frac{1}{4}$, and $\frac{3}{8}$) and practice computationally entering the data in decimal notation. Explain the use of decimal notation in computing for accuracy (e.g., money, medication doses, manufacturing products, shipping costs).
5	Convert among different-sized standard measurement units within a given measurement system (e.g., convert 5cm to 0.05m; 9 ft to 108in), and explain why the data entry format can impact how a computing program can access the use of data.

E.DA.3A. Compare and contrast ways to store data using technology.

Grade	Complexity Statement
K	Apply the process to save a data file in a computing system.
1	Apply the process of saving and naming a computational data file on an external storage device (e.g., USB, flash drive, servers, external hard drive).
2	Describe the cybersecurity risks of using external data storage devices with a computing system (e.g., introducing viruses, malware, or storage devices getting lost or stolen).
3	Explain what cloud data storage is and compare it to external storage devices.
4	Explain and apply the appropriate data units (e.g., bits, bytes, kilobytes, megabytes, gigabytes, terabytes) to numerical quantities of data.
5	Compare and contrast the storage size capacity for computational storage devices and select the appropriate format for given tasks.

E.DA.3B. Explain how to save and name data, search for data, retrieve data, modify data, and delete data using a computing device.

Grade	Complexity Statement
K	Locate and open a data file on a computing system.
1	Locate the proper computing application and create a new file.
2	Explain how to delete a file and return or recover the file if accidentally deleted.
3	Design and apply a file naming structure to organize computational data for sharing with others so they can use and locate the files
4	Modify the names of files and describe why file naming is connected to function in school or workplace settings.
5	Compare and contrast the uses and data stored on the file types: JPEG, GIF, PDG, DOC, or DOCX, XLS, or DLSX, PPT or PPTX, MP3 and WAV.

E.DA.4A. Organize and present data visually in at least three ways to highlight relationships and evaluate a claim.

Grade	Complexity Statement
K	Describe patterns of similarity to organize data into countable amounts and record them in a computational data tool.
1	Generate numerical predictions (based on prior experiences) and compare them with data collected from a tool (observed data) in a computational table and on bar graphs.
2	Collect data and record it in a computational table. Use the table to generate a bar graph and a pie graph computationally. Utilize the 3 data visualizations to evaluate a claim.
3	Create a computational data table using data visualizations (e.g., graphs, or pie charts) from multiple data collection events on the same phenomenon. Analyze newly created data tables for patterns and hypothesize what they mean for the phenomenon.
4	Collect and use data to generate the appropriate computational visualizations (e.g., data table, bar, pie, line graph) to explain a hypothesis.
5	Collect five or more data sets, visually represent the data computationally, and then compare and contrast the visualization made and findings/conclusions created with those of peers.

E.DA.4B. Evaluate data quality and clean data when indicated using the criteria of validity, accuracy, completeness, consistency, and uniformity.

Grade	Complexity Statement
K	Recognize data used to determine valid groupings. (e.g., identify, sort, and group objects by their type, explaining the reason for inclusion or exclusion from the data set)
1	Recognize the need for consistency and completeness when evaluating data. (e.g., students trying to track the temperature for a given week when given a set of incomplete data. For instance, temperatures were supposed to be taken every day, but 3 measurements were missed.)
2	Propose and design a protocol for collecting data, ensuring that it is valid, consistently measured, and complete.
3	Measure and estimate liquid volumes and masses of objects using standard units of grams, kilograms, and liters and record the data computationally.
4	When given a scenario that requires a preset place value and system of measurement, computationally sort and analyze a data set that contains various units of measurement (e.g., feet, inches, meters, centimeters) and decimals for accuracy. Explain and justify the data conversions or strategies they use to analyze the data.
5	When given a set of data and the constraints of a scenario, evaluate it for the criteria of validity, accuracy, completeness, consistency, and uniformity. Explain any issues with the data set and propose corrections to a fictitious client.

E.DA.5A. Utilize data to create models, answer investigative questions, and make predictions.

Grade	Complexity Statement
K	Computationally create and describe shapes (squares, circles, triangles, rectangles, hexagons, cubes, cones, cylinders, and spheres).
1	Computationally create partitions of circles and rectangles into two and four equal shares, describe each share with the labels halves, fourths, and quarters, and use the phrases half of, fourth of, and quarter of. Describe the whole as two of or four of the shares. Understand for these examples that decomposing into more equal shares creates smaller ones.
2	Plan and create a computational artifact to illustrate thoughts, ideas, and problems in a sequential (step-by-step) manner (e.g., story mapping, a story board, or a sequential graphic organizer with clip art).

E.DA.5A. Utilize data to create models, answer investigative questions, and make predictions.

Grade	Complexity Statement
3	Using a computational data artifact (e.g., chart, program's code, story map), make predictions and judge if there is enough data to support a claim.
4	Recreate sample shapes computationally, and classify two-dimensional shapes based on the presence or absence of parallel or perpendicular lines or the presence or absence of angles of a specified size. Draw a line and test if symmetry is possible for each figure.
5	Computationally represent real-world and mathematical problems by creating a data table and graphing points in the first quadrant of the coordinate plane and interpreting coordinate values of points in the context of the situation or examining a statement for validity.

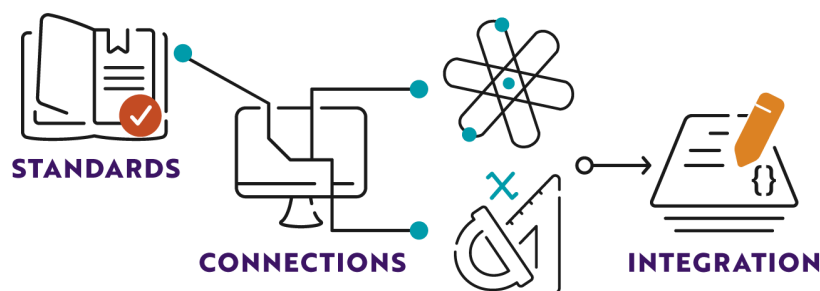
E.DA.5B. Analyze data for patterns and relationships.

Grade	Complexity Statement
K	Classify and count objects into given observable categories, then use a computational data table to record data.
1	Computationally solve word problems that call for the addition of three whole numbers whose sum is less than or equal to 20 (e.g., use three different clip art images for three different objects and a template divided into four sections to represent item 1 count + item two count + item three count = total item count).
2	Using block-based coding, create a square or rectangular geometric shape using a repeat function. Explain the relationship between adding more or less than four repeats and the picture's outcome.
3	Create computational addition or multiplication tables and identify arithmetic patterns. Apply the patterns identified to predict the next three outcomes.
4	Generate a number pattern that follows a given code rule using block-based coding.
5	Generate two numerical patterns using two given rules and block-based code. Identify apparent relationships between corresponding terms. For ordered pairs consisting of corresponding terms from the two patterns, create a data table and graph the ordered pairs computationally.

Thinking Across Disciplines in Grade 4

Core Concept 3: Data and Analysis

THINKING ACROSS DISCIPLINES (T.A.D.)



The standards in **Core Concept 3: Data and Analysis** focus on collecting and representing digital data, understanding how data is stored, and creating and analyzing computational data for pattern recognition. These skills are closely tied to concepts teachers already address in their classrooms.

- Organizing and representing data visually in different formats strengthens the ability to **highlight relationships and support conclusions**. This skill enhances communication and interpretation of complex information.
- Drawing conclusions based on evidence depends on understanding various data types and using data to make and support claims. Strengthening this skill supports **evidence-based reasoning**.
- Classifying data and choosing appropriate collection methods support accurate analysis and deepen understanding of content. These practices promote **thoughtful data use** in scientific investigations, social research, math applications, and more.
- Analyzing digital data for patterns and relationships enhances **informational literacy** by reinforcing the ability to locate, use, and evaluate data effectively. This skill develops **critical thinking** and reasoning abilities necessary for making predictions, constructing explanations, and creating models, supporting inquiry and problem-solving processes.

T.A.D. Examples

Standard	<p>E.DA.1A. Organize and present data visually to highlight relationships and support claims.</p> <p>Complexity Statement: Compare computational generated graphs (e.g., bar and pie) to their corresponding data tables to validate their accuracy.</p> <p>Core Practices: Collaborating around computing; Recognizing and defining computational problems; Creating computational artifacts</p>
Connections	<p>Science</p> <p>4-ESS2-2 Analyze and interpret data from maps to describe patterns of Earth's features.</p>
Integration	<p>As students analyze data from maps to identify patterns in Earth's features, they can use digital tools to organize and present the information visually. For example, a student might compare a data table to a graph created from that same data to better highlight patterns, which reinforces how visual representations can support scientific claims and deepen data analysis.</p>

Standard	<p>E.DA.1B. Classify types of data and describe the attributes used to sort data.</p> <p>Complexity Statement: Analyze two or more types of data to support or disprove a claim.</p> <p>Core Practices: Collaborating around computing; Recognizing and defining computational problems; Creating computational artifacts</p>
Connections	<p>Science</p> <p>4-ESS2-1 Plan and conduct investigations on the effects of water, ice, wind, and vegetation on the relative rate of weathering and erosion.</p>
Integration	<p>As students investigate the effects of water, ice, wind, and vegetation on weathering and erosion, they can classify and use multiple types of data (such as numerical measurements and written observations) to support or challenge a claim. This reinforces the importance of recognizing data types and sorting them by relevant attributes to strengthen scientific reasoning.</p>

Standard	<p>E.DA.2A. Select the appropriate data collection tool and technique to gather data to support a claim or communicate information.</p> <p>Complexity Statement: Compare and contrast various data sets collected for accuracy and describe potential sources of instrumentation error.</p> <p>Core Practices: Fostering responsible cyber citizenship; Collaborating around computing; Recognizing and defining computational problems; Developing and using abstractions; Creating computational artifacts; Testing and refining computational artifacts; Communicating about computing</p>
Connections	<p>Science</p> <p>Science & Engineering Practices: Analyzing and interpreting data</p>
Integration	<p>When analyzing and interpreting data, students learn to recognize possible causes of inaccuracy, such as outliers from instrumentation error. For example, while reviewing measurements from scientific tools, students can evaluate which data collection methods were used and determine whether the tools or techniques selected were appropriate, reinforcing the connection between accurate data gathering and reliable analysis.</p>

Standard	<p>E.DA.2B. Describe and collect data utilizing the appropriate units of measure and discuss how data format impacts a computing system.</p> <p>Complexity Statement: Collect data in measurements that require fractions (e.g., $\frac{1}{2}$, $\frac{1}{4}$, and $\frac{3}{4}$) and practice computationally entering the data in decimal notation. Explain the use of decimal notation in computing for accuracy (e.g., money, medication doses, manufacturing products, shipping costs).</p> <p>Core Practices: Fostering responsible cyber citizenship; Collaborating around computing; Recognizing and defining computational problems; Developing and using abstractions; Creating computational artifacts; Testing and refining computational artifacts; Communicating about computing</p>
Connections	<p>Mathematics</p> <p>4.NF.C Understand decimal notation for fractions, and compare decimal fractions.</p>
Integration	<p>Grade 4 students solve problems involving measurement and convert fractions into decimal forms. These skills can be used to enter data in decimal notation. For example, they might measure objects and enter lengths using both fraction and decimal notation, recognizing how choosing different data formats affects how computing systems store and process information.</p>

Standard	<p>E.DA.3A. Compare and contrast ways to store data using technology.</p> <p>Complexity Statement: Explain and apply the appropriate data units (e.g., bits, bytes, kilobytes, megabytes, gigabytes, terabytes) to numerical quantities of data.</p> <p>Core Practices: Fostering responsible cyber citizenship; Collaborating around computing; Recognizing and defining computational problems; Developing and using abstractions; Creating computational artifacts; Testing and refining computational artifacts; Communicating about computing</p>
Connections	<p>Mathematics</p> <p>4.OA.A.3 Solve multi-step word problems posed with whole numbers and having whole-number answers using the four operations, including problems in which remainders must be interpreted. Represent these problems using equations with a letter standing for the unknown quantity. Assess the reasonableness of answers using mental computation and estimation strategies including rounding. <i>Example: Twenty-five people are going to the movies. Four people fit in each car. How many cars are needed to get all 25 people to the theater at the same time?</i></p>
Integration	<p>Students apply multi-step problem-solving skills to understand data storage units and capacities. For example, when calculating if John has enough computer storage for a 24-megabyte file, students use operations and estimation to determine how much additional space is needed, while also comparing different data storage methods and units to deepen their understanding of digital information.</p>

Standard	<p>E.DA.3B. Explain how to save and name data, search for data, retrieve data, modify data, and delete data using a computing device.</p> <p>Complexity Statement: Modify the names of files and describe why file naming is connected to function in school or workplace settings.</p> <p>Core Practices: Fostering responsible cyber citizenship; Collaborating around computing; Recognizing and defining computational problems; Developing and using abstractions; Creating computational artifacts; Testing and refining computational artifacts; Communicating about computing</p>
Connections	<p>Science</p> <p>4-PS3-1 Use evidence to construct an explanation relating the speed of an object to the energy of that object.</p>
Integration	<p>When constructing explanations about how an object’s speed relates to its energy, students practice organizing their work by saving digital files with clear, relevant names. For example, naming a document “LastName_FirstName_SpeedEnergyExplanation” helps students efficiently locate and manage their data, reinforcing good digital habits alongside scientific inquiry.</p>

Standard	<p>E.DA.4A. Organize and present data visually in at least three ways to highlight relationships and evaluate a claim.</p> <p>Complexity Statement: Collect and use data to generate the appropriate computational visualizations (e.g., data table, bar, pie, line graph) to explain a hypothesis.</p> <p>Core Practices: Creating computational artifacts; Communicating about computing</p>
Connections	<p>Science</p> <p>Science & Engineering Practices: Analyzing and interpreting data</p>
Integration	<p>Collecting and interpreting data supports effective communication. Students can select from various computational tools (such as word processors, spreadsheets, or online graphing platforms) to organize and present data visually in multiple formats, enhancing their ability to highlight relationships and evaluate scientific claims.</p>

Standard	<p>E.DA.4B. Evaluate data quality and clean data when indicated using the criteria of validity, accuracy, completeness, consistency, and uniformity.</p> <p>Complexity Statement: When given a scenario that requires a preset place value and system of measurement, computationally sort and analyze a data set that contains various units of measurement (e.g., feet, inches, meters, centimeters) and decimals for accuracy. Explain and justify the data conversions or strategies they use to analyze the data.</p> <p>Core Practices: Creating computational artifacts; Communicating about computing</p>
Connections	<p>Mathematics</p> <p>4.MD.A Solve problems involving measurement and conversion of measurements from a larger unit to a smaller unit.</p>
Integration	<p>Converting measurements from larger to smaller units can be done computationally to analyze the data. Students solve measurement conversion problems by entering data into digital tools such as spreadsheets or forms. They then sort and review the data to evaluate its quality by checking for accuracy and consistency and make corrections as needed. This process reinforces both math skills and data management practices.</p>

Standard	<p>E.DA.5A. Utilize data to create models, answer investigative questions, and make predictions.</p> <p>Complexity Statement: Recreate sample shapes computationally, and classify two-dimensional shapes based on the presence or absence of parallel or perpendicular lines or the presence or absence of angles of a specified size. Draw a line and test if symmetry is possible for each figure.</p> <p>Core Practices: Recognizing and defining computational problems; Developing and using abstractions; Creating computational artifacts; Testing and refining computational artifacts; Communicating about computing</p>
Connections	<p>Mathematics</p> <p>4.G.A.2 Classify two-dimensional figures based on the presence or absence of parallel or perpendicular lines, or the presence or absence of angles of a specified size. Recognize right triangles as a category, and identify right triangles.</p>
Integration	<p>Students use block-based coding platforms to create two-dimensional figures such as right triangles and parallelograms by programming the angles and lines. They can then review each other's code to predict and classify shapes based on properties like parallel or perpendicular lines and specific angle measures, combining geometric reasoning with computational modeling.</p>

Standard	<p>E.DA.5B. Analyze data for patterns and relationships.</p> <p>Complexity Statement: Generate a number pattern that follows a given code rule using block-based coding.</p> <p>Core Practices: Recognizing and defining computational problems; Developing and using abstractions; Creating computational artifacts; Testing and refining computational artifacts; Communicating about computing</p>
Connections	<p>Science</p> <p>4-PS3-1 Use evidence to construct an explanation relating the speed of an object to the energy of that object.</p>
Integration	<p>Students use block-based coding platforms to create animations that illustrate the relationship between an object's speed and its energy. For example, they can program moving objects whose speed changes, accompanied by visual indicators such as a speedometer or energy bar that grows with increased speed. This activity helps students analyze patterns and relationships between speed and energy through computational modeling.</p>

Core Concept 4: Algorithms and Programming (AP)

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 systems. Algorithms and programs control all computing systems, empowering people to communicate with the world in novel ways and solve compelling problems. The development process to create meaningful and efficient programs involves choosing which information to use, how to process the data, and how to store the information. The decomposition of more significant problems into simpler ones, combined with recombining existing solutions and analyzing various solutions, helps determine the most appropriate solution to a problem.

Elementary Standards for Core Concept 4: Algorithms and Programming

Subconcepts and Core Practices	Elementary (K-5)
Subconcept 1: Variables and Algorithms <i>Core Practices: Collaborating around computing; Recognizing and defining computational problems; Developing and using abstractions; Creating computational artifacts; Testing and refining computational artifacts; Communicating about computing</i>	E.AP.1A. Create clearly named variables representing different data types and perform operations on the variables' values.
	E.AP.1B. Create, use, and apply an algorithm to complete a task. Compare the results of algorithm usage trials and refine the algorithm.
Subconcept 2: Control Structures <i>Core Practices: Collaborating around computing; Recognizing and defining computational problems; Developing and using abstractions; Creating computational artifacts; Testing and refining computational artifacts; Communicating about computing</i>	E.AP.2A. Define what a control structure is and create programs that include sequences, conditionals, events, and loops.
Subconcept 3: Modularity <i>Core Practices: Collaborating around computing; Recognizing and defining computational problems; Developing and using abstractions; Creating computational artifacts; Testing and refining computational artifacts; Communicating about computing</i>	E.AP.3A. Define and apply decomposition to a complex problem in order to create smaller subproblems that can be solved through step-by-step instructions.
	E.AP.3B. Modify, remix, or incorporate parts of an existing problem's solution to develop something new or add more advanced features to a program.

Subconcept 4: Program Development <i>Core Practices: Fostering responsible cyber citizenship; Collaborating around computing; Recognizing and defining computational problems; Developing and using abstractions; Creating computational artifacts; Testing and refining computational artifacts; Communicating about computing</i>	E.AP.4A. Create a simple program to achieve a goal with expected outcomes.
	E.AP.4B. Test and debug a program or algorithm to ensure the program produces the intended outcome.
	E.AP.4C. Collaborate with a team of peers to design, implement, test, and review the stages of program development.
	E.AP.4D. Identify intellectual property rights and apply the appropriate attribution when creating or remixing programs.
	E.AP.4E. Describe and justify the steps taken and choices made during a program's development.
	E.AP.4F. Using an iterative process, test a program step-by-step and document areas of refinement.

Complexity Statements

Core Concept 4: Algorithms and Programming

Complexity statements for the standards included in **Core Concept 4: Algorithms and Programming** break down each standard into manageable, age-appropriate components. This breakdown ensures that instruction fits the developmental needs of students at each grade level. For instance, in kindergarten, the focus is on creating simple step-by-step plans and understanding basic sequencing, while by grade 5, students create efficient algorithms by structuring repeated actions, decompose complex tasks into manageable steps, and design multi-step programs using block-based coding to create animations or solve problems with various constraints.

E.AP.1A. Create clearly named variables representing different data types and perform operations on the variables' values.	
Grade	Complexity Statement
K	Count and group items under broad categorical (variable) names (e.g., apples + bananas + grapes + kiwi = total amount of fruit).
1	Understand subtraction as an unknown-addend problem.
2	Use addition and subtraction within 100 to solve one- and two-step word problems involving situations of adding to, taking from, putting together, taking apart and comparing, with unknowns in all positions. (e.g., by using drawings and equations with a symbol for the unknown number to represent the problem.)
3	Solve real-world and mathematical problems involving perimeters of polygons, including finding the perimeter given the side lengths, finding an unknown side length, and exhibiting rectangles with the same perimeter and different areas or with the same area and different perimeters.
4	Represent unknown quantities in equations with a letter. Define what a variable is and why a letter can be utilized as a placeholder or representative of the numerical value.
5	Apply the area and perimeter formulas for rectangles in real-world and mathematical problems. (e.g., find the width of a rectangular room, given the area of the flooring and the length, by viewing the area formula as a multiplication equation with an unknown factor.)

E.AP.1B. Create, use, and apply an algorithm to complete a task. Compare the results of algorithm usage trials and refine the algorithm.

Grade	Complexity Statement
K	Create an algorithm with set step-by-step stages to complete a task.
1	Apply the properties of operations to add and subtract.
2	Work with equal groups of objects to gain the foundations for multiplication. Identify how many ways exist to make equal groups and write equations that express the total as a sum of equal addends. Compare the results of the equations written and share solutions with peers.
3	Solve problems involving the four operations, and identify and explain patterns in arithmetic.
4	Develop and refine algorithms to solve multi-step word problems that only use whole numbers, including whole number remainders. Represent these equations with a letter standing for the unknown quantity. Assess the reasonableness of answers using mental computation and estimation strategies, including rounding. When possible, share and compare thought processes and attempt to solve the problem with a new strategy or approach.
5	Create efficient algorithms by structuring repeated actions using parentheses or brackets and optimizing lengthy instructions into simpler equations using multiplication, addition, and subtraction.

E.AP.2A. Define what a control structure is and create programs that include sequences, conditionals, events, and loops.

Grade	Complexity Statement
K	Use block-based coding to code (program) a simple set of instructions to make a character, programmable object, figure, or robot move along a teacher-specified path. Explain what the commands do to their program.
1	Using block-based coding, code (program) a multi-step set of instructions to make a figure or robot move in a teacher-specified sequence that includes stops, starts, and turns. Define the term control structure in programming and give examples.
2	Using block-based coding, code (program) a multi-step set of instructions to make a figure or robot move using loops that enable the iteration of commands. Define loops with examples and explain how it is a control structure for programming.

3	Using block-based coding, code (program) a multi-step set of instructions to make a figure or robot move based upon conditionals (e.g., if-else structures or blocking the program from execution until an event occurs). Define conditionals with examples and explain how they are a control structure for programming.
4	Using block-based coding, code (program) a multi-step set of instructions to make a figure or robot move with conditionals, loops, and sequences to achieve a teacher-specified complex task. Compare and contrast the three types of control structures being used.
5	When given a complex task, create a solution using block-based coding, test it, and compare the solution code with peers to refine it for efficient control structure usage.

E.AP.3A. Define and apply decomposition to a complex problem in order to create smaller subproblems that can be solved through step-by-step instructions.

Grade	Complexity Statement
K	With teacher assistance and modeling, divide a task into smaller parts and identify the steps to solve the smaller parts.
1	Define decomposition and apply it to fact families (e.g., given three numbers, students explore the four ways to write them to achieve correct addition and subtraction questions). Explain thinking.
2	Apply the decomposing and composing subtraction strategy to work with three-digit numbers (e.g., $257-132 =$).
3	Diagram and determine the appropriate steps to achieve a mathematical solution to a multi-step word problem (e.g., John read 541 pages this summer. To reach his personal reading goal of 650 pages, how many more pages should he read?)
4	Compare color bar models, addends, unit fraction addition formulas, and number bond charts to decompose fractions and explain the reasoning for each model.
5	Decompose and create steps for solving complex problems that address various constraints such as cost, efficiency, length of time, number of people needed, etc. (e.g., The principal has asked our class to paint a mural on the side of the school that measures six feet tall and twelve feet long. What is the best _____ way to do this?).

E.AP.3B. Modify, remix, or incorporate parts of an existing problem’s solution to develop something new or add more advanced features to a program.

Grade	Complexity Statement
K	Analyze out-of-order sequential images of a process and reconstruct them in the correct order. Justify the chosen sequence with peers (e.g., give students pictures of the steps in making a white bread ham, cheese, lettuce, tomato, and mayonnaise sandwich. Ask them to put the steps (pictures) in order to make a sandwich. Prompt upon completion with questions such as: “Does the order of items matter? Why/why not?” “Could we change an ingredient and still make a sandwich? Which one(s)?”)
1	Using block-based coding, code (program) a multi-step set of instructions to make a figure or robot move in a teacher-specified sequence that includes stops, starts, and turns. Rreorganize the steps to make the figure return to its starting point (remixing).
2	Diagram and determine the appropriate steps to multi-step coding word problems on paper to achieve a desired outcome. Using block-based coding, code (program) the multi-step set of instructions to make a figure or robot move. For an additional challenge, attempt to successfully modify the code in response to the teacher changing two or more parts of the original task. (e.g., make the rabbit go over four, up six, and to the left eight spaces to escape a fox. Challenge: The rabbit needs to go back to three spaces below his starting point but can’t cross over his previous route or the Fox catches him.)
3	Collaborate to design and implement a step-by-step plan for completing a task (e.g., cup stacking with symbol-based instructions). Exchange plans, test, and provide feedback to peers to refine the plan.
4	Design a step-by-step plan (algorithm) to draw a polygon. Collaborate with peers to combine and improve these plans, then modify the best plan to create different polygons.
5	Using block-based coding, code (program) a multi-step set of instructions to make an animation for a story through the practices of decomposition, programming planning, and code optimization. (e.g., students decompose the story into parts, identify the actions and code needed on paper, review the code for parts that can be reused or recycled, and simplify the written code before adding it to the program. For each scene, students need a background, they need to place characters, and to program actions sequentially.)

E.AP.4A. Create a simple program to achieve a goal with expected outcomes.

Grade	Complexity Statement
K	Create a step-by-step plan using pictures. (e.g., students place picture cards showing how a two-story house is built in the correct sequence.)

1	When given a scenario, sort and organize the appropriate written steps to solve the problem.
2	When given a scenario, create written steps to solve the problem with minimal prompting from the teacher.
3	Collaborate to propose solutions for each stage of a multi-stage problem or scenario (e.g., making a cake, icing the cake, and then decorating the cake using figurines).
4	Collaborate to propose solutions for each stage of a multi-stage problem or scenario. Share proposed solutions with others and refine solutions using peer feedback.
5	When given a scenario that involves three or more distinct stages to create a solution, independently propose solutions. Trade solutions and analyze peers' work, making suggestions for refinement.

E.AP.4B. Test and debug a program or algorithm to ensure the program produces the intended outcome.

Grade	Complexity Statement
K	Find and fix errors in a picture-based step-by-step process. Use visual cards that illustrate a step-by-step process from start to finish but with errors. Identify the errors and propose fixes with guidance from the teacher (e.g., remove a repeated step, resequence a step, add a missing step).
1	Using block-based coding, test the code (program) of a multi-step set of instructions to make a figure or robot move in a specified sequence that includes stops, starts, and turns which does not work due to errors in the code. Debug the code and work towards achieving the originally intended outcome.
2	Using block-based coding, test the code (program) of a multi-step set of instructions to make a figure or robot move using loops that enable the iteration of commands which contain errors preventing the loops from working. Working with a partner, debug the code to explain how to properly correct the control structure, then test the program.
3	Using block-based coding, test code (program) of a multi-step set of instructions to make a figure or robot move based upon conditionals which contains errors preventing the conditionals from working (e.g., if-else structures or blocking the program from execution until an event occurs). With a partner, debug the code to explain how to properly correct this control structure, then test the program.
4	Using block-based coding, test the code (program) of a multi-step set of instructions to make a figure or robot move with conditionals, loops, and sequences which contains errors preventing the various control structures from working. With a partner, debug to explain how to properly correct this control structure, then test the program.

5	When given a complex task, create a solution using block-based coding and test it. Then, introduce up to 4 errors into the code and write it out (with errors on paper). Trade the flawed code with a peer. Test and debug the code. Reflect and write down what was challenging and why.
---	---

E.AP.4C. Collaborate with a team of peers to design, implement, test, and review the stages of program development.

Grade	Complexity Statement
K	Define what a programmer is and apply what they do by modeling the creation of a simple step-by-step program to accomplish a task.
1	Collaborate in teams of three to complete an outcome-driven coding task. Apply the steps of planning, designing, and testing to a simple code sequence using block-based coding or written steps. Share responsibilities and discuss challenges encountered during each step.
2	Visualize the decomposition of a complex task into smaller steps using hierarchy structure/flowcharts. Assess the chart to identify the appropriate written steps or block-based code instructions to write code and test the code.
3	When provided with a code, work in groups to test and debug it. Once testing is complete, write a sentence story (manual) that provides an overview of the program's function and how to address the errors that were encountered. (This is to simulate the documentation phase of program development.)
4	Identify times when a program may need maintenance and explain how this can happen in the real world with relevant examples (e.g., new versions of software versus updates).
5	Review the stages of program development. Consider a scenario with a real-world objective outcome. Follow the stages and document progress in writing appropriate code.

E.AP.4D. Identify intellectual property rights and apply the appropriate attribution when creating or remixing programs.

Grade	Complexity Statement
K	With teacher support and modeling, explore how to describe what code does using simple words or pictures. Look at examples of code with explanatory comments (some clear, some unclear), observe what the code does, and talk about whether the explanations provided make sense.
1	Define and identify examples of attribution, plagiarism, and piracy in computing and digital media visual examples.

2	Define what ownership means in computing. Using teacher-demonstrated examples, identify the identification symbology and text indicators of copyright, patent, and trademark. With teacher guidance, discuss what each term means and create class definitions which include rules for using each term.
3	Analyze examples of plagiarism, copyright infringement, piracy, trademark infringement, and patent violation. Determine if inappropriate use has occurred and why.
4	Define and describe what the public domain is. Using examples or Internet research, describe what penalties can occur for not following attribution laws.
5	Define and model appropriate attribution for using another programmer's code. Create simple programs and share the code with peers to incorporate into their own programs, and apply appropriate attribution guidelines to the merged codes.

E.AP.4E. Describe and justify the steps taken and choices made during a program's development.

Grade	Complexity Statement
K	Sequence visual steps to a familiar process and collaboratively discuss the reasoning behind the chosen order. (e.g., using chart paper with visual cards, students place the numbers 1, 2, 3, etc., next to the images that represent steps in a process. The teacher leads the class in discussing why steps must happen in a certain order.)
1	Draw and create the steps of a familiar process (e.g., making a sandwich). Exchange work with a peer who can add additional step drawings on sticky notes to the author's work. Justify the changes made to the original sequence.
2	Compare and contrast the steps of solving a math problem or doing a science lab to the steps of coding. Explain the choices made in each situation and how it can impact the task.
3	Create a group presentation (e.g., PowerPoint or slide deck) on how to modify erroneous code to debug a program.
4	Define commenting on code with examples. Using a program that has multiple errors (printed on paper), annotate changes or suggestions to debug the code on the margins of the paper.
5	Define clarity in written comments for other programmers. Given several examples of annotated code (some correct and some incorrect), test each and provide written feedback on both the effectiveness of the annotations and the clarity.

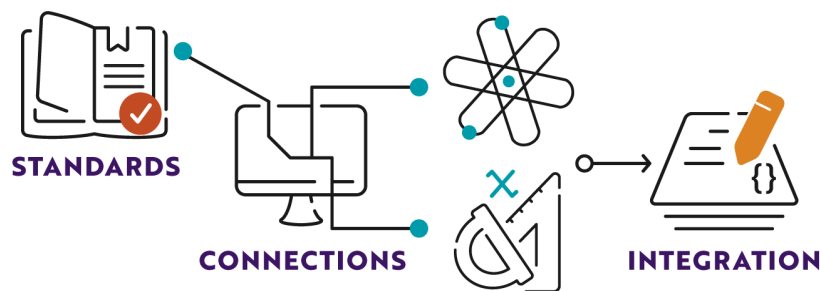
E.AP.4F. Using an iterative process, test a program step-by-step and document areas of refinement.

Grade	Complexity Statement
K	With teacher guidance, examine a process that is lengthy and capable of being shortened. Identify which parts can be removed to shorten the process and still attain the objective.
1	Define and apply the term iterative process. Through multiple rounds of testing and refinement, collectively develop an optimized solution to a problem. (e.g., provide students with a complex set of instructions (e.g., 20 steps) for how to stack eight plastic cups using only arrows ←↑→↓↖↗↘↙ and commas. In groups, students will modify the given instructions, document their changes, and test their revised process three times. After all iterations are complete, have the class create a clarification to the original stacking system (e.g., down to 5-6 steps) that results in the desired outcome.)
2	Using the iterative process, refine an error-filled code in groups. Document the changes and compare the work with another set of peers. As a class, identify and describe similarities and differences in the strategies tried and why.
3	Using block-based coding, test flawed code (program) of a multi-step set of instructions to make a figure or robot move with conditionals, loops, and sequences to achieve a teacher-specified complex task in which errors preventing the various control structures from working. Partner code and debug to explain how to properly correct the control structure, then test the program and document the changes. Once complete, generate a new code to achieve the objective and explain the refinements in writing.
4	Describe common strategies to test and reasons for wanting to improve a program. Using a previous example of the student-made code (from another project or earlier school year's work), document and refine the codes to address one of the improvement areas.
5	Iteratively improve a given multi-step complex program in groups, documenting all modifications and the reasoning behind each change. Through a gallery walk, compare and contrast different approaches to improving the code, culminating in a discussion led by group leaders interpreting the goal of "improve the code."

Thinking Across Disciplines in Grade 4

Core Concept 4: Algorithms and Programming

THINKING ACROSS DISCIPLINES (T.A.D.)



The standards included in **Core Concept 4: Algorithms and Programming** focus on the foundational programming skills, including creating and manipulating variables, designing and refining algorithms to solve tasks, and understanding control structures. These standards emphasize problem-solving through decomposition, writing, testing, debugging, and collaborating on programs while promoting ethical practices through proper attribution. These skills are closely tied to concepts teachers already address in their classrooms.

- Through decomposition, students break complex problems into smaller, manageable pieces solvable with step-by-step instructions, providing a practical context for practicing **problem-solving skills**. Additionally, using an iterative approach to test, refine, and improve solutions cultivates persistence and adaptability in problem-solving.
- When working with a team of peers, students build **collaboration skills** that support academic growth as well as social and emotional development. The collaborative development of solutions enhances collective creativity. Using the work of others and respecting intellectual property supports academic integrity in creating and modifying work.
- **Computational thinking** is a way of expressing solutions to problems as steps that can be carried out. Just as computers need step-by-step instructions to process information, humans use similar processes to complete tasks.

T.A.D. Examples

Standard	<p>E.AP.1A. Create clearly named variables representing different data types and perform operations on the variables' values.</p> <p>Subconcept 1: Variables and Algorithms</p> <p>Complexity Statement: Represent unknown quantities in equations with a letter. Define what a variable is and why a letter can be utilized as a placeholder or representative of the numerical value.</p> <p>Core Practices: Collaborating around computing; Recognizing and defining computational problems; Developing and using abstractions; Creating computational artifacts; Testing and refining computational artifacts; Communicating about computing</p>
Connections	<p>Mathematics</p> <p>4.OA.A Use the four operations with whole numbers to solve problems.</p>
Integration	<p>A variable is often used where students need to “fill in the blank” with a number to solve a problem. Students use variables to represent unknown values when solving problems, such as finding “n” in the equation $7 \times n = 35$. In coding, they create clearly named variables to store data and perform operations, reinforcing how variables function both in math and programming contexts.</p>

Standard	<p>E.AP.1B. Create, use, and apply an algorithm to complete a task. Compare the results of algorithm usage trials and refine the algorithm.</p> <p>Complexity Statement: Develop and refine algorithms to solve multi-step word problems that only use whole numbers, including whole number remainders. Represent these equations with a letter standing for the unknown quantity. Assess the reasonableness of answers using mental computation and estimation strategies, including rounding. When possible, share and compare thought processes and attempt to solve the problem with a new strategy or approach.</p> <p>Core Practices: Collaborating around computing; Recognizing and defining computational problems; Developing and using abstractions; Creating computational artifacts; Testing and refining computational artifacts; Communicating about computing</p>
Connections	<p>Mathematics</p> <p>4.OA.A.3 Solve multi-step problems posed with whole numbers and having whole-numbers using the four operations, including problems in which remainders must be interpreted. Represent these problems using equations with a letter standing for the unknown quantity. Assess the reasonableness of answers using mental computation and estimation strategies, including rounding.</p>

Integration	<p>Solving problems in both mathematics and computer science requires logical thinking and a structured approach. In the classroom, students solve multi-step math problems by outlining each step needed to reach a solution, using strategies such as equations, estimation, or mental math. In parallel, they apply the concept of algorithms in computer science by creating and following step-by-step instructions to complete tasks. After testing their approach, they compare results and refine their methods. This process helps reinforce problem-solving skills and supports the development of logical reasoning across both subjects.</p>
Standard	<p>E.AP.2A. Define what a control structure is and create programs that include sequences, conditionals, events, and loops.</p> <p>Complexity Statement: Using block-based coding, code (program) a multi-step set of instructions to make a figure or robot move with conditionals, loops, and sequences to achieve a teacher-specified complex task. Compare and contrast the three types of control structures being used.</p> <p>Core Practices: Collaborating around computing; Recognizing and defining computational problems; Developing and using abstractions; Creating computational artifacts; Testing and refining computational artifacts; Communicating about computing</p>
Connections	<p>Mathematics</p> <p>4.G.A Draw and identify lines and angles, and classify shapes by properties of their lines and angles.</p>
Integration	<p>Combining geometry with programming helps students connect mathematical concepts to real-world applications of computer science. As students explore two-dimensional shapes, they can write algorithms to draw polygons and use a physical robot or an online simulation to visualize the path of the shape. For example, students create block-based code using loops to draw a square, reinforcing their understanding of equal angles and line lengths. This activity allows students to apply their knowledge of lines and angles while practicing control structures such as sequences, loops, and events in programming.</p>

Standard	<p>E.AP.3A. Define and apply decomposition to a complex problem in order to create smaller subproblems that can be solved through step-by-step instructions.</p> <p>Complexity Statement: Compare color bar models, addends, unit fraction addition formulas, and number bond charts to decompose fractions and explain the reasoning for each model.</p> <p>Core Practices: Collaborating around computing; Recognizing and defining computational problems; Developing and using abstractions; Creating computational artifacts; Testing and refining computational artifacts; Communicating about computing</p>
Connections	<p>Mathematics</p> <p>4.NF.A Extend understanding of fraction equivalence and ordering.</p>
Integration	<p>Just as complex problems can be broken into smaller, manageable subproblems solved step-by-step (through a process known as decomposition), students use color bar models to decompose fractions. For example, they explore how different fractions combine or break down to form a whole or another fraction, applying problem decomposition skills from math and computer science.</p>

Standard	<p>E.AP.3B. Modify, remix, or incorporate parts of an existing problem's solution to develop something new or add more advanced features to a program.</p> <p>Complexity Statement: Design a step-by-step plan (algorithm) to draw a polygon. Collaborate with peers to combine and improve these plans, then modify the best plan to create different polygons.</p> <p>Core Practices: Collaborating around computing; Recognizing and defining computational problems; Developing and using abstractions; Creating computational artifacts; Testing and refining computational artifacts; Communicating about computing</p>
Connections	<p>Mathematics</p> <p>4.G.A Draw and identify lines and angles, and classify shapes by properties of their lines and angles.</p>
Integration	<p>Creating a step-by-step plan (or algorithm) to draw polygons helps students and teachers identify understanding and address misconceptions. For example, students write algorithms to draw polygons, then collaborate to assess and modify these algorithms, remixing and enhancing solutions to create different shapes while applying geometric concepts of lines and angles.</p>

Standard	<p>E.AP.4A. Create a simple program to achieve a goal with expected outcomes.</p> <p>Complexity Statement: Collaborate to propose solutions for each stage of a multi-stage problem or scenario. Share proposed solutions with others and refine solutions using peer feedback.</p> <p>Core Practices: Fostering responsible cyber citizenship; Collaborating around computing; Recognizing and defining computational problems; Developing and using abstractions; Creating computational artifacts; Testing and refining computational artifacts; Communicating about computing</p>
Connections	<p>Mathematics</p> <p>4.G.A Draw and identify lines and angles, and classify shapes by properties of their lines and angles.</p>
Integration	<p>Collaborating to propose solutions is an important skill used in each core content. As an example, students could collaborate to design a house using polygons, with each group coding a different part. After creating their shapes, they propose ways to combine the polygons into a complete house and provide peer feedback on each other's algorithms, developing problem-solving and programming skills alongside geometric understanding.</p>

Standard	<p>E.AP.4B. Test and debug a program or algorithm to ensure the program produces the intended outcome.</p> <p>Complexity Statement: Using block-based coding, test the code (program) of a multi-step set of instructions to make a figure or robot move with conditionals, loops, and sequences which contains errors preventing the various control structures from working. With a partner, debug to explain how to properly correct this control structure, then test the program.</p> <p>Core Practices: Fostering responsible cyber citizenship; Collaborating around computing; Recognizing and defining computational problems; Developing and using abstractions; Creating computational artifacts; Testing and refining computational artifacts; Communicating about computing</p>
Connections	<p>Mathematics</p> <p>4.OA.B.4.D Determine whether a given whole number is prime or composite.</p>
Integration	<p>As students begin learning about prime and composite numbers, they can apply this understanding in a coding context to reinforce both content areas. For example, students can analyze a block-based program designed to move a figure forward if the input is a prime number and backward if it is composite. When the figure does not move as expected, students investigate the code to identify logic or syntax errors. Through this process, they debug the algorithm, apply their knowledge of prime and composite numbers, and strengthen problem-solving and computational thinking skills.</p>

Standard	<p>E.AP.4C. Collaborate with a team of peers to design, implement, test, and review the stages of program development.</p> <p>Complexity Statement: Identify times when a program may need maintenance and explain how this can happen in the real world with relevant examples (e.g., new versions of software versus updates).</p> <p>Core Practices: Fostering responsible cyber citizenship; Collaborating around computing; Recognizing and defining computational problems; Developing and using abstractions; Creating computational artifacts; Testing and refining computational artifacts; Communicating about computing</p>
Connections	<p>Science</p> <p>Science & Engineering Practices: Asking questions and defining problems</p>
Integration	<p>Most people have experienced a time when a device or software needed to be updated. If using computational devices in the classroom, chances are the device has needed to be updated. When devices or software require updates, students discuss how programmers review and improve programs by identifying problems and implementing fixes. This process parallels software updates and bug fixes on everyday devices like cell phones, illustrating the stages of program development and collaboration in problem solving.</p>

Standard	<p>E.AP.4D. Identify intellectual property rights and apply the appropriate attribution when creating or remixing programs.</p> <p>Complexity Statement: Define and describe what the public domain is. Using examples or Internet research, describe what penalties can occur for not following attribution laws.</p> <p>Core Practices: Fostering responsible cyber citizenship; Collaborating around computing; Recognizing and defining computational problems; Developing and using abstractions; Creating computational artifacts; Testing and refining computational artifacts; Communicating about computing</p>
Connections	<p>Science</p> <p>Science & Engineering Practices: Obtaining, evaluating, and communicating information</p>
Integration	<p>Obtaining, evaluating, and communicating information from online sources is an important scientific skill. Students need to be able to research information, evaluate the validity of the information, and properly attribute the source of information. For example, students research topics using online sources, learning to evaluate the validity of information and distinguish between public domain and copyrighted content. They apply appropriate attribution when creating or remixing digital work, understanding intellectual property rights and the importance of giving proper credit.</p>

Standard	<p>E.AP.4E. Describe and justify the steps taken and choices made during a program’s development.</p> <p>Complexity Statement: Define commenting on code with examples. Using a program that has multiple errors (printed on paper), annotate changes or suggestions to debug the code on the margins of the paper.</p> <p>Core Practices: Fostering responsible cyber citizenship; Collaborating around computing; Recognizing and defining computational problems; Developing and using abstractions; Creating computational artifacts; Testing and refining computational artifacts; Communicating about computing</p>
Connections	<p>Mathematics</p> <p>4.G.A Draw and identify lines and angles, and classify shapes by properties of their lines and angles.</p>
Integration	<p>As students create code, it is important to be able to identify bugs (or errors) in the code and understand ways to correct the error. An example is students reviewing block code intended to draw a square that contains errors. Using their understanding of angles and shape properties, they identify bugs, annotate corrections, and explain the reasoning behind their debugging process.</p>

Standard	<p>E.AP.4F. Using an iterative process, test a program step-by-step and document areas of refinement.</p> <p>Complexity Statement: Describe common strategies to test and reasons for wanting to improve a program. Using a previous example of the student-made code (from another project or earlier school year’s work), document and refine the codes to address one of the improvement areas.</p> <p>Core Practices: Fostering responsible cyber citizenship; Collaborating around computing; Recognizing and defining computational problems; Developing and using abstractions; Creating computational artifacts; Testing and refining computational artifacts; Communicating about computing</p>
Connections	<p>Mathematics</p> <p>4.G.A Draw and identify lines and angles, and classify shapes by properties of their lines and angles.</p>
Integration	<p>Designing a program is an ongoing, cyclical process that involves creating, testing, gathering feedback, and making refinements. In mathematics, as students work with block coding to create specific angles, they apply this iterative approach by testing their programs step-by-step. During testing, students document areas where the code could be improved and make adjustments, reinforcing their understanding of geometric concepts and programming practices simultaneously.</p>

Core Concept 5: Impacts of Computing (IC)

Overview

The impacts of computing can be positive and negative, enabling innovation, communication, and access to information, while also raising concerns around ethics, privacy, and fairness. Individuals and communities not only shape computing systems through interactions, behaviors, industry practices, and laws, but are also shaped by interactions with computing systems. Computer and data science create new means of communication by accelerating information exchange and establishing cyberspace as a dynamic workspace for individuals.

Elementary Standards for Core Concept 5: Impacts of Computing

Subconcepts and Core Practices	Elementary (K-5)
Subconcept 1: Intellectual Achievements <i>Core Practices: Fostering responsible cyber citizenship; Collaborating around computing; Testing and refining computational artifacts; Communicating about computing</i>	E.IC.1A. Describe how computing has changed the ways people live and work.
Subconcept 2: Social Interaction <i>Core Practices: Fostering responsible cyber citizenship; Collaborating around computing; Testing and refining computational artifacts; Communicating about computing</i>	E.IC.2A. Identify and describe examples of appropriate versus inappropriate computer communications.
	E.IC.2B. Identify examples of cyberbullying with age-appropriate responses.
Subconcept 2: Laws, Safety, and Industry Practices <i>Core Practices: Fostering responsible cyber citizenship; Collaborating around computing; Testing and refining computational artifacts; Communicating about computing</i>	E.IC.3A. Explain how online actions have real-world consequences and that laws and rules may also apply online.
	E.IC.3B. Describe the safe versus unsafe uses of computing systems at age-appropriate levels.
	E.IC.3C. Explain how the school and school system's computing rules and policies keep students safe.

Complexity Statements

Core Concept 5: Impacts of Computing

Complexity statements for the standards under **Core Concept 5: Impacts of Computing** break down each standard into manageable, age-appropriate components. This breakdown ensures that instruction fits the developmental needs of students at each grade level. For instance, in kindergarten, the focus is on understanding basic computer usage rules and recognizing how computing systems are used daily. By grade 5, students describe modifications to increase usability and accessibility, explain age restrictions on digital communication, and model appropriate online behaviors and responses to issues like cyberbullying.

E.IC.1A. Describe how computing has changed the ways people live and work.	
Grade	Complexity Statement
K	List and describe ways that computing systems are used by students every day. With teacher guidance, identify how selected examples may decrease student workload.
1	Sketch or list all of the computing systems that are in the classroom and one other school location. In groups, collaboratively discuss how these systems and students interact.
2	Compare and contrast doing a task with and without computing systems (e.g., reading a story from a book versus on a digital screen) and discuss the pros and cons of the task completion each way.
3	Explain how an invention has changed through the addition of a computing system. (e.g., electric power, a car, the telephone, or an airplane).
4	Identify and describe a career that relies on technology.
5	Describe the ways that computing systems can be modified to increase usability and accessibility.

E.IC.2A. Identify and describe examples of appropriate versus inappropriate computer communications.	
Grade	Complexity Statement
K	Discuss and model working respectfully and responsibly.
1	Explain when to and when not to share log-in information.
2	Create a class list of feedback comments that offer constructive feedback and practice using them in class.

3	Collaborate as a group to design a digital presentation and have students point out, critique, and suggest feedback to peers via comments.
4	Examine computational communication examples and identify where appropriate and inappropriate comments are made.
5	Explain why certain digital communication applications are age restricted.

E.IC.2B. Identify examples of cyberbullying with age-appropriate responses.

Grade	Complexity Statement
K	Describe what cyberbullying is and who to tell if it happens.
1	Identify ways to respond to inappropriate cyber-communication.
2	Explain who bystanders, allies, and upstanders are with example scenarios.
3	Classify examples of online behaviors as either being cyberbullying or a normal part of the computing conversation.
4	Explain what responsible digital citizenship is.
5	Create presentations on what to do and who to contact if you are cyberbullied.

E.IC.3A. Explain how online actions have real-world consequences and that laws and rules may also apply online.

Grade	Complexity Statement
K	Explain that there are rules for using computers in comparison to real-world classroom rules and examples.
1	Describe how making threats or using abusive language can lead to legal consequences.
2	Explain why it is not acceptable to purchase items online without the proper permissions.
3	Explain what a copyright is and why permission is needed to use certain images, digital media, and other intellectual properties.
4	Describe what unauthorized access is and how participating in it can lead to legal repercussions (e.g., hacking, jailbreaking phones, illegal sharing of copyrighted materials).

5	Research relevant computing laws and create awareness posters for peers.
---	--

E.IC.3B. Describe the safe versus unsafe uses of computing systems at age-appropriate levels.

Grade	Complexity Statement
K	Review and list responsible and safe online behaviors each time computing systems are used.
1	Explain why clicking on pop-up ads is unsafe.
2	List and categorize safe versus unsafe uses of computing systems.
3	Propose and apply class norms for group project collaborative computing for use in the classroom (e.g., Google Docs, commenting to peers through messaging options, sharing work versus electronically copying it).
4	Describe what computer viruses are and how to protect against them.
5	Explain the risks of responding to or chatting with unknown individuals online.

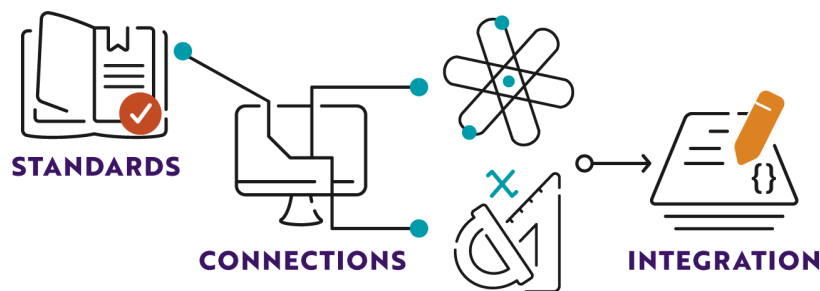
E.IC.3C. Explain how the school and school system's computing rules and policies keep students safe.

Grade	Complexity Statement
K	Discuss and model the rules for computer use at school.
1	Explain why the use of another student's log-in is not allowed.
2	Identify and discuss the appropriate use of the Internet under the school's computer usage rules.
3	Classify computer usage examples as appropriate or inappropriate under the school's computer usage plan.
4	Explain why downloading or uploading programs puts all users of the school's computer network at risk.
5	Describe and model the appropriate behaviors and practices for participating in virtual calls, collaborating with peers, and making public presentations.

Thinking Across Disciplines in Grade 4

Core Concept 5: Impacts of Computing

THINKING ACROSS DISCIPLINES (T.A.D.)



The standards in **Core Concept 5: Impacts of Computing** promote digital citizenship, online safety, and responsible computing practices. These standards emphasize understanding the impact of computing on daily life and work, recognizing appropriate and inappropriate online behavior, and addressing issues like cyberbullying with effective responses. Additionally, they highlight the importance of understanding online consequences, adhering to laws and rules, and following school policies to ensure safe and ethical use of technology. These skills are tied to concepts teachers already address in their classrooms.

- Interacting responsibly online in a safe environment builds upon student **communication skills**. Understanding safe and appropriate online behavior and the impacts of unethical behavior support effective communication strategies.
- **Responsibility and accountability** represent essential life skills that apply in both physical and digital contexts. Understanding responsible computing practices and the consequences of unethical behavior reinforces ethical decision-making.
- Distinguishing safe versus unsafe uses of computing systems at developmentally appropriate levels enables **critical judgement and decisions-making skills** that protect personal information, maintain security, and promote responsible participation in digital environments

T.A.D. Examples

Standard	<p>E.IC.1A. Describe how computing has changed the ways people live and work.</p> <p>Complexity Statement: Identify and describe a career that relies on technology.</p> <p>Core Practices: Fostering responsible cyber citizenship; Collaborating around computing; Testing and refining computational artifacts; Communicating about computing</p>
Connections	<p>Science</p> <p>4-PS3-4 Apply scientific ideas to design, test, and refine a device that converts energy from one form to another.</p>
Integration	<p>Computing devices play an important role in converting energy from one form to another, such as electrical energy into light or sound. In science class, as students design and test devices that demonstrate energy conversion, they can also explore how similar technologies are used in real-world settings. This provides an opportunity to identify careers that rely on these types of technologies, helping students make connections between classroom learning and how computing has changed the way people live and work.</p>

Standard	<p>E.IC.2A. Identify and describe examples of appropriate versus inappropriate computer communications.</p> <p>Complexity Statement: Examine computational communication examples and identify where appropriate and inappropriate comments are made.</p> <p>Core Practices: Fostering responsible cyber citizenship; Collaborating around computing; Testing and refining computational artifacts; Communicating about computing</p>
Connections	<p>Science</p> <p>Science & Engineering Practices: Obtaining, evaluating, and communicating information</p>
Integration	<p>As students engage with others through computer-based platforms, whether offering peer feedback, participating in discussions, or submitting assignments, it is essential to teach responsible interaction. For example, during a classroom activity where students provide feedback on one another's work, teachers can facilitate conversations about what constitutes appropriate versus inappropriate comments. Presenting sample comments for evaluation helps students build the skills to communicate respectfully and effectively online.</p>

Standard	<p>E.IC.2B. Identify examples of cyberbullying with age-appropriate responses.</p> <p>Complexity Statement: Explain what responsible digital citizenship is.</p> <p>Core Practices: Fostering responsible cyber citizenship; Collaborating around computing; Testing and refining computational artifacts; Communicating about computing</p>
Connections	<p>Science</p> <p>Science & Engineering Practices: Obtaining, evaluating, and communicating information</p>
Integration	<p>As students increasingly interact in online environments — whether through educational tools, games, or social media — it becomes essential to teach responsible digital citizenship. Responsible digital citizenship means using the internet in ways that are legal, safe, and respectful. Teachers can introduce this concept during peer feedback activities by discussing how to communicate responsibly online. Extend the conversation to platforms students may encounter, such as YouTube or social media, and guide students in identifying respectful behavior and recognizing inappropriate or harmful interactions, such as cyberbullying. Provide age-appropriate strategies for responding to and reporting cyberbullying when it occurs.</p>

Standard	<p>E.IC.3A. Explain how online actions have real-world consequences and that laws and rules may also apply online.</p> <p>Complexity Statement: Describe what unauthorized access is and how participating in it can lead to legal repercussions (e.g., hacking, jailbreaking phones, illegal sharing of copyrighted materials).</p> <p>Core Practices: Fostering responsible cyber citizenship; Collaborating around computing; Testing and refining computational artifacts; Communicating about computing</p>
Connections	<p>Science</p> <p>Science & Engineering Practices: Obtaining, evaluating, and communicating information</p>

Integration	<p>Digital tools are an important part of students' learning environments, and understanding responsible digital citizenship includes recognizing that actions taken online can have real-world consequences. When students log into school accounts, it is an opportunity to introduce concepts such as authorized versus unauthorized access. For example, explain that logging into someone else's account without permission, even as a joke, is a form of unauthorized access and can lead to disciplinary action or legal consequences. Real-world examples may include stories of individuals facing consequences for hacking into social media accounts or school systems. These discussions help students understand that online behaviors are governed by rules and laws, just like behaviors in the physical world.</p>
--------------------	--

Standard	<p>E.IC.3B. Describe the safe versus unsafe uses of computing systems at age-appropriate levels.</p> <p>Complexity Statement: Describe what computer viruses are and how to protect against them.</p> <p>Core Practices: Fostering responsible cyber citizenship; Collaborating around computing; Testing and refining computational artifacts; Communicating about computing</p>
Connections	<p>Science</p> <p>Science & Engineering Practices: Asking questions and defining problems</p>
Integration	<p>Students regularly use digital tools to support learning, making it important to understand how to use computing systems safely. Just as germs can make people sick, computer viruses can harm devices and compromise users' information. While using technology in the classroom, teachers can guide discussions about what computer viruses are, how they spread, and why safe computing practices are necessary. These conversations can include how antivirus software helps protect devices and why students should avoid unsafe behaviors like clicking unknown links or downloading unverified files. Through these discussions, students learn to recognize the difference between safe and unsafe uses of computing systems.</p>

Standard	<p>E.IC.3C. Explain how the school and school system's computing rules and policies keep students safe.</p> <p>Complexity Statement: Explain why downloading or uploading programs puts all users of the school's computer network at risk.</p> <p>Core Practices: Fostering responsible cyber citizenship; Collaborating around computing; Testing and refining computational artifacts; Communicating about computing</p>
-----------------	--

Connections	<p>Science</p> <p>Science & Engineering Practices: Obtaining, evaluating, and communicating information</p>
Integration	<p>Students researching topics in science on school computers have an opportunity to learn about computer safety and why it is important. As students gather information online, they might encounter a prompt or link to download a program, (perhaps one for energy calculations or organizing research). When conducting online research, teachers should ensure students know not to download just any program. School computer policies prohibit unauthorized downloads and uploads because these programs can contain harmful viruses or malware. Malicious software can spread across the entire interconnected school network, potentially disrupting or damaging computing systems, files, and data for all users. Adhering to the school's policies when obtaining and evaluating online information ensures a safe computing environment for the entire school community.</p>