# 2024
# Computer Science Standards of Learning

Grade 5 Instructional Guide

VIRGINIA DEPARTMENT OF EDUCATION

Virginia Department of Education
P.O. Box 2120
Richmond, Virginia 23218-2120
http://www.doe.virginia.gov



**Superintendent of Public Instruction**
Emily Anne Gullickson

**Assistant Superintendent of Teaching and Learning**
Michelle Wallace, Ph.D.

**Office of Educational Technology and Classroom Innovation**
Calypso Gilstrap, Associate Director
Keisha Tennessee, Computer Science Coordinator

# Table of Contents

**2024 Computer Science *Standards of Learning:*** ..... ***3***

**Introduction** ..... **3**

**Foundational Principles** ..... **3**

**Fifth Grade: 2024 Computer Science *Standards of Learning*** ..... **5**

Algorithms and Programming (AP) ..... 5

Computing Systems (CSY) ..... 6

Cybersecurity (CYB) ..... 6

Data and Analysis (DA) ..... 7

Impacts of Computing (IC) ..... 7

Networks and the Internet (NI) ..... 8

**Computer Science Instructional Guide Framework** ..... **10**

**Grade 5: Computer Science** ..... **11**

Algorithms and Programming (AP) ..... 11

Computing Systems (CSY) ..... 39

Cybersecurity (CYB) ..... 50

Data and Analysis (DA) ..... 57

Impacts of Computing (IC) ..... 77

Networks and the Internet (NI) ..... 93

**Appendix A** ..... **97**

K-5 Computer Science Skills and Practices Continuum ..... 97

**Appendix B** ..... **105**

Grade 5 Computer Science Vocabulary ..... 105

# 2024 COMPUTER SCIENCE *STANDARDS OF LEARNING*:

## Introduction

Virginia's Computer Science standards aim to raise our aspirations for computational instruction to enable students to engage and thrive in a digital world. Beginning in the earliest grades and continuing through 12th grade, students must develop a foundation of computer science knowledge and learn new approaches to problem solving that harness the power of computational thinking to become both users and creators of computing technology.

It is important for every student to engage in computer science education from the earliest ages. This early and sustained access equips students with foundational problem-solving practices, develops their understanding of how current and emerging computer science technologies work, and fosters curiosity, interest, and innovation with computer science.

## Foundational Principles

**Computer Literacy is foundational to learning and post-secondary success** as technology becomes increasingly incorporated into all aspects of everyday life. Computer Literacy provides critical knowledge and skills for all subject areas including mathematics, science, history, English, and fine arts. By applying computer science as a tool for learning and expression in a variety of disciplines and interests, students will actively and proficiently participate in a world that is increasingly influenced by digital technology.

**Computer Science fosters problem solving skills that are essential to all educational disciplines and post-secondary employment opportunities.** Understanding how multi-step solutions are executed within computer programs allows students the opportunity to use metacognitive strategies with tasks they are performing as they work and study in any topic area. Computer Science should become an essential part of Virginia K-12 education, accessible by all, rather than a vocational part of education only for those headed to technology-based employment.

**Computer Science instruction must maintain the pace of technology evolution to prepare students for the workforce.** Computer science is a core technology component for students to have the ability to adapt to the future evolution of work. The workforce of the future will increasingly require that all adults effectively work in digital environments and utilize technology both ethically and responsibly. As a result, we must prioritize preparing all students with integral computer science learning opportunities throughout their academic career to ensure they are prepared for a post-secondary success in a digital world that includes computer-based problem solving, artificial intelligence and communication rooted in the use of digital tools.

**Students should gain specific digital and computational concepts to harness the power of computer science** and derivative applications, such as machine learning, online programming, virtual reality, and Artificial Intelligence (AI), to embrace innovation and chart the future of individuals, business, and government responsibly.

## Instructional Intent and Integration

Computer science is an academic discipline that encompasses both conceptual foundations and applied practices. It can be taught effectively with or without computing devices, as many key skills, such as logical reasoning, pattern recognition, decomposition, and sequencing can be developed through with or without a computing device.

In primary grades, overlapping concepts between computer science and other content areas may be taught within the same instructional context. When doing so, it is essential that educators intentionally align instruction to ensure that the full intent and specifications of the computer science standard are addressed, even when the learning experience is shared with another content area.

As students' progress into upper elementary and beyond, instruction should be explicit, ensuring students are able to identify and understand the computer science concepts and practices embedded within those shared experiences. By naming the connections and calling out the domain specific elements of computer science, students can deepen their disciplinary understanding, build metacognitive awareness, and transfer their knowledge and skills across contexts.

It is important to recognize that not all computer science concepts will naturally overlap with other subjects. Concepts such as algorithms, data representation, networks, and programming require dedicated instructional time and may be taught independently of other content areas. Whether through integration or stand-alone instruction, computer science should be approached with the same level of intentionality and rigor as other academic subjects, ensuring students develop a coherent and comprehensive understanding from kindergarten through grade 12.

# Fifth Grade: 2024 Computer Science *Standards of Learning*

In Fifth Grade, students explore cloud computing, examine security risks and implement best practices to protect information. They research cybersecurity policies and laws, researching ways to prevent unauthorized access to data. Algorithms undergo expansion with the inclusion of multi-way branching and nested conditional control structures. Students develop programs incorporating rational and arithmetic expressions for performing calculations. Evaluation of data source reliability becomes a focus, enabling students to draw conclusions from data visualizations. Additionally, the impact of social interactions through computing technologies and the exploration of open-source licenses are addressed, along with considerations on expanding the capabilities of computing devices.

## Algorithms and Programming (AP)

### 5.AP.1 The student will apply computational thinking to identify patterns, make use of decomposition to break down problems or processes into sub-components, and design algorithms.

a. Identify patterns and repeated steps in an algorithm, problem, or process.
b. Decompose a problem or process into a subset of smaller problems or groups of sequential instructions.
c. Abstract relevant information to identify essential details.
d. Design an algorithm to solve a problem.

### 5.AP.2 The student will plan and implement algorithms that consist of sequencing, loops, variables, user input, and nested conditional control structures using a block-based programming language.

a. Describe the concept of nested conditional control structure.
b. Create a design document to trace and predict an algorithm using plain language, pseudocode, or diagrams.
c. Read, write, and interpret nested conditional control structures: “if-else” and “if-else; if-else" statements.

### 5.AP.3 The student will use the iterative design process to create, test, and debug programs containing sequencing, loops, variables, user inputs, nested conditional control structures, and two-way branching conditional control structures in a block-based programming tool.

a. Use accurate terminology to describe and explain the iterative design process.
b. Create and test programs that consist of sequencing, loops, variables, user inputs, nested conditional control structures, and two-way branching conditional control structures.
c. Trace and predict outcomes of programs.
d. Analyze and describe program results to assess validity of outcomes.
e. Revise and improve programs to resolve errors or produce desired outcomes.

## Computing Systems (CSY)

**5.CSY.1 The student will explain how computing systems are used to collect and exchange data.**

a. Identify and explain how computing systems store data representations, including images and sound.
b. Describe the role of processing speed and storage capacity when collecting and exchanging data.

**5.CSY.2 The student will describe an automated decision-making process employed by a computing system.**

a. Explore decision automation and how it is used.
b. List outcomes of a process based on automated decisions.

**5.CSY.3 The student will evaluate and implement troubleshooting strategies when a computing system is not operational.**

a. Identify and use troubleshooting protocols to resolve hardware, software, and connectivity issues.
b. Apply prior troubleshooting practices to new problems as they arise.

## Cybersecurity (CYB)

**5.CYB.1 The student will identify ways to limit unauthorized access on computing devices.**

a. Define virus, malware, and phishing.
b. Explain how viruses and malware can put personal information at risk.
c. Describe the role of human interactions in social engineering attacks.
d. Identify ways to protect personal and private information when using a computing device and the Internet.
e. Explain the importance of updating software.

**5.CYB.2 The student will explain how cybersecurity policies and laws are designed to protect individuals.**

a. Explain the importance of policies and laws related to online use of computing devices and the Internet.
b. Research and discuss current cybersecurity policies and laws that protect individuals.
c. Explain legal consequences for inappropriate use of computing technologies.

## Data and Analysis (DA)

### 5.DA.1 The student will collect data or use data sets to solve a problem or investigate a topic.

a. Identify accurate ways data can be collected.
b. Evaluate the reliability of the data source.
c. Organize data based on similarities or patterns.
d. Compare and contrast various data elements.

### 5.DA.2 The student will create multiple data representations to make predictions and conclusions.

a. Formulate questions that require the collection or acquisition of data.
b. Collect data to use in creating charts, graphs, and models.
c. Analyze data as evidence to draw conclusions and make predictions.
d. Propose solutions to problems or questions based on data analysis.

### 5.DA.3 The student will explain the significance of training data in machine learning.

a. Compare how training data is utilized in supervised, unsupervised and reinforcement learning.
b. Explain how training data is used to make classification predictions.
c. Discuss the need and significance of diverse, inclusive, and large datasets.

## Impacts of Computing (IC)

### 5.IC.1 The student will analyze the impact of inappropriate use of computing technologies.

a. Predict consequences for inappropriate uses of computing technologies.
b. Describe how technology-related problems can be avoided or prevented.
c. Develop solutions for a scenario involving inappropriate use of computing technologies.

### 5.IC.2 The student will explain the potential impact of excessive screen time on academic performance.

a. Analyze data to determine the impact of screen time on academic performance.
b. Describe how academic behaviors that lead to academic success are impacted by daily screen time.
c. Differentiate usage of screen time that benefit and hinder academic performance.

**5.IC.3 The student will identify the impact of computing technologies on the workforce, culture, and global society.**

a. Research and analyze computing technology careers in global society.
b. Explore the impact of emerging technologies on workforce, culture, and global society.

**5.IC.4 The student will observe and examine intellectual property rights when considering the use of open-source licenses and copyrights.**

a. Distinguish between open-source licenses and copyrights.
b. Research risks associated with inappropriate use of various digital information sources.
c. Describe and use strategies to protect online digital content and resources.

**5.IC.5 The student will examine the effects of social interactions due to computing technologies.**

a. List and explain how advances in computing technologies impact communication and collaboration.
b. Describe how computing technologies can be designed to engage and interact with users including those with diverse needs.
c. Evaluate activities conducted in the physical and online environments.
d. Create an artifact that illustrates a solution to address the need or want of a user.

## Networks and the Internet (NI)

**5.NI.1 The student will identify and describe cloud computing.**

a. Define cloud computing.
b. List examples of cloud computing.
c. List the advantages and disadvantages of cloud computing.
d. Identify safe practices and potential security risks when using cloud computing.

# Computer Science

# Instructional Guide

This instructional guide, a companion document to the 2024 Computer Science *Standards of Learning*, amplifies each standard by defining the core knowledge and skills in practice, supporting teachers and their instruction, and serving to transition classroom instruction from the 2017 Computer Science *Standards of Learning* to the newly adopted standards.

# Computer Science Instructional Guide Framework

This instructional guide includes, Understanding the Standard, Concepts and Connections, Opportunities for Integration, and Skills in Practice aligned to each standard. The purpose of each is explained.

### Understanding the Standard

This section is designed to unpack the standards, providing both students and teachers with the necessary knowledge to support effective instruction. It includes core concepts that students are expected to learn, as well as background knowledge that teachers can use to deepen their understanding of the standards and plan standards-aligned lessons.

### Concepts and Connections

This section outlines concepts that transcend grade levels and thread through the K through 12 computer science as appropriate at each level. Concept connections reflect connections to prior grade-level concepts as content and practices build within the discipline as well as potential connections across disciplines. The connections across disciplines focus on direct standard alignment, where concepts and practices in computer science overlap with similar ideas in other disciplines.

Computer Science connections are aligned to the: 2024 English *Standards of Learning*, 2023 History and Social Science, 2023 Mathematics *Standards of Learning*, 2020 Digital Learning Integration *Standards of Learning*, and 2018 Science *Standards of Learning*.

These cross-disciplinary concepts and practices are foundational for effective interdisciplinary integration.

### Opportunities for Integration

This section provides lesson ideas for integrating computer science with English, history and social science, mathematics, and science through multidisciplinary, interdisciplinary, and transdisciplinary approaches. Lesson ideas may involve the integration of standards that may or may not be directly aligned yet are strategically taught together to achieve a purposeful and authentic learning experience that supports meaningful student outcomes such as deeper understanding, skill transfer, and real-world application.

### Skills in Practice

This section focuses on instructional strategies that teachers can use to develop students' skills, deepen their conceptual understanding, and encourage critical thinking. These practices are designed to support curriculum writers and educators in weaving pedagogical approaches that deepen student understanding of unit and course objectives, ultimately enhancing learning outcomes. This section provides a framework for planning effective and engaging lessons

# Grade 5: Computer Science

In Fifth Grade, students explore cloud computing, examine security risks and adopt best practices to protect information. They research cybersecurity policies and laws, researching ways to prevent unauthorized access to data. Algorithms undergo expansion with the inclusion of multi-way branching and nested conditional control structures. Students develop programs incorporating rational and arithmetic expressions for performing calculations. Evaluation of data source reliability becomes a focus, enabling students to draw conclusions from data visualizations. Additionally, the impact of social interactions through computing technologies and the exploration of open-source licenses are addressed, along with considerations on expanding the capabilities of computing devices.

## Algorithms and Programming (AP)

### 5.AP.1 The student will apply computational thinking to identify patterns, make use of decomposition to break down problems or processes into sub-components, and design algorithms.

a. Identify patterns and repeated steps in an algorithm, problem, or process.
b. Decompose a problem or process into a subset of smaller problems or groups of sequential instructions.
c. Abstract relevant information to identify essential details.
d. Design an algorithm to solve a problem.

## Understanding the Standard

Computational thinking (CT) is a logical and systematic problem-solving process that uses decomposition, pattern recognition, abstraction, and algorithm thinking to foster creativity and develop solutions. It is universally applicable across various fields and allows individuals to break down complex problems and develop efficient solutions. Its role in computer science is particularly important, as it serves as the foundation for designing algorithms, analyzing data, and solving real-world challenges through the use and development of technology. Computational thinking is an integral part of Virginia's computer science standards.

The four main computational thinking components are decomposition, pattern recognition, abstraction, and algorithmic thinking.

- Decomposition is the process of breaking apart a problem, process, or task into smaller, more manageable components. This involves identifying and recognizing relationships among the parts.
- Pattern recognition involves identifying commonalities, similarities, or differences in recurring elements.
- Abstraction is a filtering process. It enables one to focus on important and relevant information, while excluding or hiding irrelevant or less important details.
- Algorithmic thinking is the creation of step-by-step instructions or algorithms to solve the problem or task.

**[5.AP.1a]** Identifying patterns and repeated steps process refers to the ability to recognize recurrent behaviors, sequences, or structures that appear in data, actions or processes. Programmers use pattern recognition to make code concise and more efficient, to make predictions, and to create smarter solutions using algorithms. Pattern recognition is a key skill in programming as it used in the construction and implementation of a program to reduce unnecessary redundancy. Instead, a programmer is able to implement loops or functions to repeat tasks automatically within the code. Consider the following example:

- A student is developing a program to check whether a password is strong. The student identifies a pattern: a strong password must include uppercase and lowercase letters, numbers, special characters, and must avoid dictionary words, repeated passwords, or personal information.
- To check these requirements, the student uses a function, such as *is_strong_password,* to evaluate whether the entered password meets all the criteria.
- The student can use a loop with conditionals (aligned to standard 5.AP.2) that repeatedly prompts the user to enter a password until a strong one is provided. This structure not only ensures the password meets all safety requirements but also makes the program more efficient and the code easier to manage.

**[5.AP.1b]** Students extend their computational thinking and apply an essential skill in algorithmic design by making use of decomposition to break down complex problems or processes into subsets of smaller problems or groups of sequential instructions. This process encourages logical thinking and promotes effective problem-solving by demonstrating how a larger problem can be broken down into smaller, more manageable components. By considering each part systematically, students are able to see how they fit together to form the overall solution. This approach not only simplifies the problem but also guides students toward designing algorithms, allowing them to solve smaller subsets and identify the necessary steps to tackle the entire challenge. In a program to determine if a password is strong enough, a student may decompose a problem into smaller checks in the following way:

Function: “is __ strong __ password”

- Length check: Is the password at least 8 characters long?
- Uppercase letter check: Does it contain at least one uppercase letter?
- Lowercase letter check: Does it contain at least one lowercase letter?
- Number check: Does it contain at least one number?
- Symbol check: Does it contain at least one symbol?
- Dictionary check: Does it contain words from a dictionary?
- Repetition check: Does it contain the same password twice?
- Personal Information check: Does it contain personal information?

**[5.AP.1c]** Abstraction is a technique used in computer science to reduce complexity by ignoring unnecessary details and only keeping the relevant or essential information. Students engage with abstraction in their daily lives. Consider the following scenarios:

- While making a sandwich, it is not necessary to know how the bread is baked, how the lettuce grows, or how the cheese is made. A student just grabs the ingredients and puts them together. The act of making a sandwich simplifies all the steps that happened before (like farming, processing, and baking), so the student needs only to focus on choosing and assembling the ingredients.

- While using a calculator to add numbers, it is not necessary to know how the calculator performs the math inside. A student just presses the numbers and the operations, and the calculator gives the result. The calculator abstracts away the mathematical steps, giving the student an easy way to get the answer without doing the hard work.

- While using a smartphone app to play a game or send a message, it is not necessary to know how the phone's processor, memory, or operating system is working behind the scenes. The student just needs to tap on an app icon to have it work. The app abstracts away all of the complexity so that all of the technical details are hidden and provides a simple interface to interact with. In this way, the student can focus on playing the game or chatting with friends.

In computer science, abstraction helps us create algorithms by letting us focus on the main steps needed to solve a problem without getting distracted by small details. When designing an algorithm, we use abstraction to outline the big ideas first, such as what needs to happen and in what order, before filling in the specific instructions. This makes it easier to plan, organize, and adjust the algorithm as needed. Consider the following examples:

- A student uses block-based computer language to create programs using code blocks that represent coding constructs hidden from the user.

- A student writes an algorithm to determine the strength of a password and removes details irrelevant to the strength of the password itself. Irrelevant details that the code does not need may include highly technical terms such as "regex." Relevant details that a student would focus on may include plain language terms such as "too many repetitive patterns" or "predictable patterns" (5.AP.2). A student may also abstract away complicated processes and replace them with simple conditions or explanations such as "a password should be 8 characters long" or "a password needs to have numbers and symbols." Abstraction simplifies the problem and allows students to write an algorithm that is focused on the essential details so that the program can run more easily.

- A student writes code to program a bot. Irrelevant detail that the code does not need to contain are details about the color of the bot, how its wheels are made, or what materials were used to make the bot. These are details that a coder would abstract away. Relevant details that a coder would focus on include the instructions the bot needs to move in a particular path (e.g., move forward, turn left, and stop). Abstraction simplifies the problem and allows students to write a program for the robot more easily.

- A student designs an algorithm for editing an audio clip for use in a presentation. The student uses audio editing software to extract specific parts of a sound clip by abstracting background noise, vocals, or certain frequencies.

**[5.AP.1d]** Students apply algorithmic thinking, along with decomposition, pattern recognition, and abstraction, to design algorithms that solve problems effectively. Algorithm design involves planning and organizing a sequence of clear, logical steps that can be translated into code. In a classroom setting, students can use these skills to develop algorithms for a wide range of tasks, from organizing data to automating simple processes.

Consider the following example: Algorithm to determine password strength

Application of Computational Thinking: A student designing an algorithm to determine password strength would begin by decomposing the problem into smaller parts such as checking the length of the password, whether it includes numbers, symbols, and both uppercase and lowercase letters, and identifying repeated or predictable patterns. They would use pattern recognition to identify common characteristics of strong versus weak passwords. Through abstraction, the student would focus only on the most important criteria, ignoring unnecessary details like font or spacing to keep the algorithm clear and manageable. Using algorithmic thinking, the student would then organize these conditions into a logical sequence of steps.

- Example of an algorithm to test password strength using pseudocode (Computer Science 5.AP.2) is shown below:

```
1. Ask the user to enter a password.
2. If the password is less than 8 characters:
   - Print "Password is too short!"
   - Stop and exit.
3. Check if there is at least one uppercase letter in the password:
   - If no, print "Password must have at least one uppercase letter."
   - Stop and exit.
4. Check if there is at least one lowercase letter in the password:
   - If no, print "Password must have at least one lowercase letter."
```

```
   - Stop and exit.
5. Check if there is at least one number in the password:
   - If no, print "Password must have at least one number."
   - Stop and exit.
6. Check if there is at least one special character (like @, #, $, etc.) in the password:
   - If no, print "Password must have at least one special character."
   - Stop and exit.
7. If all checks pass, print "Password is strong!"
```

*Pseudocode example adapted from commonly available online examples.*

Consider the following example: A class is tasked with tracking which school lunch choices are most popular in order to make recommendations to the county for future lunch menus.

Application of Computational Thinking: To create a program that tracks popular school lunch choices and recommends future menu items, students apply computational thinking throughout the process. They begin with decomposition, breaking the problem into smaller tasks such as collecting, storing, and comparing lunch selection data. Through pattern recognition, they analyze the data to identify trends in student preferences. Using abstraction, they focus on key details—like meal names and selection frequency—while excluding irrelevant information to keep the program efficient. Finally, with algorithmic thinking, students design a logical sequence of steps to process the data and generate menu recommendations based on the most frequently selected items.

- Example of an algorithm to create a simple voting system to track the lunch choices in pseudocode (Computer Science 5.AP.2) is as follows:

```
1. Initialize vote counters:
   - pizza_votes = 0
   - sandwich_votes = 0
   - salad_votes = 0
```

- pasta_votes = 0

2. Ask the user how many votes there will be (e.g., number of students voting).
   - total_votes = input("How many votes will there be?")

3. For each vote:
   - Ask the voter to choose one lunch option: "Pizza", "Sandwich", "Salad", or "Pasta".
   - If the voter chooses "Pizza":
     - Increment pizza_votes by 1.
   - If the voter chooses "Sandwich":
     - Increment sandwich_votes by 1.
   - If the voter chooses "Salad":
     - Increment salad_votes by 1.
   - If the voter chooses "Pasta":
     - Increment pasta_votes by 1.
   - If the voter enters an invalid option, ask them to choose again.

4. Display the total number of votes for each lunch option:
   - Print "Pizza: " + pizza_votes
   - Print "Sandwich: " + sandwich_votes
   - Print "Salad: " + salad_votes
   - Print "Pasta: " + pasta_votes

5. Determine the most popular lunch choice:
   - If pizza_votes is greater than the other options:
     - Print "The most popular lunch choice is Pizza!"
   - If sandwich_votes is greater than the other options:
     - Print "The most popular lunch choice is Sandwich!"
   - If salad_votes is greater than the other options:
     - Print "The most popular lunch choice is Salad!"
   - If pasta_votes is greater than the other options:
     - Print "The most popular lunch choice is Pasta!"
   - If there is a tie:
     - Print "There is a tie between lunch choices."

```
6. End.
```

*Pseudocode example adapted from commonly available online examples.*

## Concepts and Connections

### Concepts

Computational thinking is a foundational practice in computer science that enables systematic problem-solving. Identifying patterns and repeated steps allows for recognition of regularities that support efficient solutions. Decomposition breaks complex problems into smaller, manageable components for targeted analysis. Abstraction focuses attention on essential information by filtering out irrelevant details, while algorithm design constructs logical, ordered steps to solve problems effectively.

### Connections

**Within the grade level/course:** At this grade level, students engage in the computational thinking practices of decomposition, pattern analysis, and algorithmic thinking as they design algorithms to solve a problem. They expand their knowledge of computational thinking by using abstraction to identify essential details in their algorithms (5.AP.1).

**Vertical Progression:** In Grade 4, students identified and designed multiple algorithms for the purpose of comparing and contrasting them to determine which algorithm is most effective for a given task (4.AP.1). In Grade 6, students will expand on abstraction by designing algorithms for the purposes of accomplishing a task or expressing a computational process (6.AP.1).

### Across Content Areas

**English**

- **5.RL.2C** The student will use textual evidence to demonstrate comprehension and build knowledge from a variety of grade-level complex literary texts read to include poetry, fantasy, humor, mystery, adventure, realistic fiction, historical fiction, and folklore/tall tales, with a focus on fantasy.
- **5.LU.2D** The student will use the conventions of Standard English when speaking and writing, differentiating between contexts that call for formal English and situations where informal discourse is more appropriate.
- **5.C.4A** The student will develop effective oral communication and collaboration skills to build a community of learners that process, understand, and interpret content together.

**Mathematics**

- **5.CE.1b** The student will estimate, represent, solve, and justify solutions to single-step and multistep contextual problems using addition, subtraction, multiplication, and division with whole numbers.

- **5.CE.4** The student will simplify numerical expressions with whole numbers using the order of operations.
- **5.PFA.1a** The student will identify, describe, extend, and create increasing and decreasing patterns with whole numbers, fractions, and decimals, including those in context, using various representations.
- **5.MG.2ad** The student will use multiple representations to solve problems, including those in context, involving perimeter, area, and volume.

**Science**

- **5.1** The student will demonstrate an understanding of scientific and engineering practices by a) asking questions and defining problems
  *5.1 standard is integrated within science content and not taught in isolation. Potential science concepts to apply 5.1 include: 5.3 (force, mass, energy transfer), 5.5 (sound), 5.6 (light), 5.8 (rocks)*

### DIGITAL LEARNING INTEGRATION

- **3-5.ID** Students use a variety of technologies, including assistive technologies, within a design process to identify and solve problems by creating new, useful or imaginative solutions or iterations.
  A. Know and use appropriate technologies in a purposeful design process for generating ideas, testing theories, creating innovative digital works, or solving authentic problems.
  C. Use appropriate technologies to develop, test, and refine prototypes as part of a cyclical design process.
- **3-5.CT** Students develop and employ strategies for understanding and solving problems in ways that leverage the power of technological methods, including those that leverage assistive technologies, to develop and test solutions.
  A. Formulate problem definitions suited for technology- assisted methods such as data analysis, modeling and algorithmic thinking in exploring and finding solutions.
  C. Break problems into component parts, extract key information, and develop descriptive models, using technologies when appropriate, to understand complex systems or facilitate problem-solving.
  D. Understand how automation works and use algorithmic thinking to develop a sequence of steps to create and test automated solutions.

## Opportunities for Integration

Curriculum integration strengthens conceptual understanding and skill application. This can be done through multidisciplinary, interdisciplinary, and transdisciplinary approaches to integration. The examples below illustrate multiple ways to integrate computer science.

**English**

- Students examine different nonfiction text in order to identify the organizational patterns (cause/effect, problem/solution, sequence, comparison/contrast, chronological order, or description).

### Mathematics

- Students use decomposition to simplify a mathematical process and break an equation into sub-components by using the order of operations as they systematically solve each step.
- Students extend patterns in objects, lists, number lines, input/output tables, or function machines by determining the next number in the pattern, writing a rule that can be used to find any number in the pattern after the first number, and when given a rule, creating patterns of their own that follow the rule.

### Science

- Students use algorithmic thinking to model how sound is produced and transmitted through different mediums.

## Skills in Practice

Students should engage in the following practices to deepen their conceptual understanding and enhance the application of skills aligned with the Computer Science *Standards of Learning*. These practices are explained in more detail in Appendix A.

### B. Fostering Computational Thinking Practices:

1. Decompose Real-World Problems
2. Explore Common Features and Identify Patterns
3. Use Abstraction to Simplify, Represent, and Problem Solve
4. Apply Algorithmic Thinking to Problem Solve and Create
5. Apply Computational Thinking Practices to Select, Organize, and Interpret Data

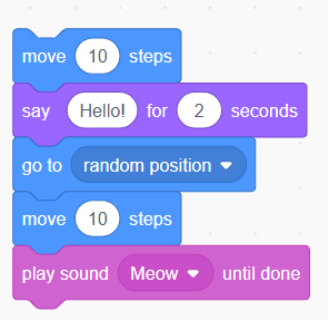
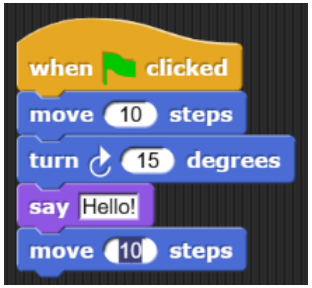
## 5.AP.2 The student will plan and implement algorithms that consist of sequencing, loops, variables, user input, and nested conditional control structures using a block-based programming language.

a. Describe the concept of nested conditional control structure.
b. Create a design document to trace and predict an algorithm using plain language, pseudocode, or diagrams.
c. Read, write, and interpret nested conditional control structures: “if-else” and “if-else; if-else" statements.

## Understanding the Standard

Students have a developing understanding of how algorithms work, how to build simple programs with block-based programming that include sequencing, loops, variables, user input, and conditional control structures, and how to use technology to solve problems.

- Sequencing is the specific order of steps, one after the other, in which instructions or steps are executed in an algorithm or program. Each step occurs in a specific order and cannot be skipped for the task to be complete. It is a fundamental programming concept and refers to arranging commands or actions in a logical, step-by-step manner within an algorithm to achieve a desired outcome or to complete different tasks. Consider the following example in a code to make a sprite move and perform different actions in a sequence of steps:

| Scratch | Snap |
|---|---|
|  |  |
| Source: https://scratch.mit.edu/?scrlybrkr=eea9261a | Source: https://snap.berkeley.edu/ |

- Loops are a set of instructions that are repeated until a specified condition is met or a predetermined number of repetitions has occurred. Their purpose is to simplify code by reducing repetition and making programs more efficient. For example, instead of having to manually repeat instructions to make a character in a game move or animate multiple times, loops allow the action to happen automatically. Consider the following example in a Scratch code to program a sprite to move multiple times:

| Without a Loop | With a Loop |
| --- | --- |
|  | |
| Source: https://scratch.mit.edu/?scrlybrkr=eea9261a | Source: https://scratch.mit.edu/?scrlybrkr=eea9261a |

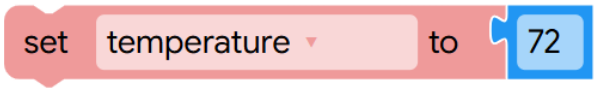
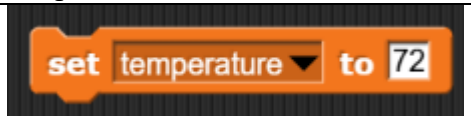
Before exploring more advanced programming concepts, students should demonstrate a solid understanding of two essential programming concepts: variables and data types. Variables provide a way to store and reference information, while data types define the kind of data being used such as numbers, text, or true/false values. Together, these concepts enable students to manage and manipulate data effectively, forming the foundation for writing dynamic, interactive programs.

- User input is the data or information that a user provides to a computer program during its execution. The program uses this input to make decisions, update values, or interact with the user. User input is crucial because it allows programs to be dynamic and interactive and to respond to what the user does or types. It can come in various forms such as text, selections from a menu, clicks, or gestures. Students recognize areas where input is needed to improve their programs. For example, in a simple math quiz program, a student identifies which type of user input is necessary by inputting the answer to a math question. The program uses the user input to provide either positive or negative feedback. Dynamic interaction with the program increases because it is responding to the users' input.

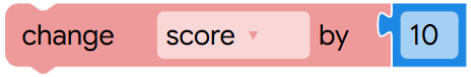
- Variables are a programming element that is a named storage location in memory that holds a value, which can be modified during the execution of a program. They are like containers that store and modify data or are analogous to "buckets" or " envelopes" where information can be maintained and referenced.

    - Think of a variable like a container with a label on the outside. This label is the name of the variable. Inside the container is an "envelope" that holds the actual data or information, which can change over time.

    - When referring to the variable, we use the name of the container, not the value stored inside.
      Example: A student creates a weather app and uses a variable named temperature to store the current weather conditions. The name temperature refers to the container in the code: the data inside (like "72°F") can change, but the name stays the same.

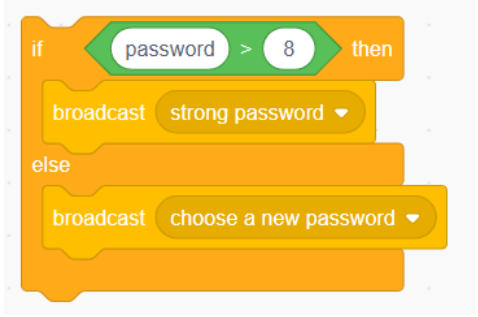
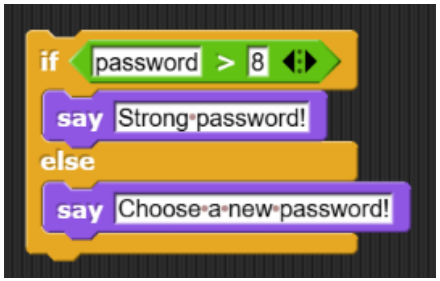
| Sample Variable: temperature | |
|---|---|
| Blockly | Snap |
|  |  |
| Source: https://developers.google.com/blockly | Source: https://snap.berkeley.edu/snap/snap.html |

- The variable "temperature" is a container that holds the temperature value. If the temperature changes (say, it becomes 80 degrees), the program can update the value inside the container, and it will display the new temperature.
- When creating a variable, it requires the user to assign it a specific data type. The data type determines the values and operations that can be performed on that data. One operation that uses data types includes adding integers together. For example, a student creates a game, including a variable called score that holds the player's score. Since the score is a whole number, it would use an integer data type.
  - score = 0
  - score = score + 10 (Increases the score by 10 after a correct answer)

| Sample Variable: Score increase | |
|---|---|
| Blockly | Snap |
|  |  |
| Source: https://developers.google.com/blockly | Source: https://snap.berkeley.edu/snap/snap.html |

- Conditional control structures use conditional logic (e.g., if-else statements) to make decisions within a computer program. These structures make decisions based upon whether or not a certain decision is true or false and allows for different actions based on certain conditions or input.
  - An "If" statement executes a block of code if a statement is true. For example, if a password is greater than eight, then the program will execute the action to state that the password is strong.

- An “Else” statement executes a block of code if a statement is false. For example, if a password is something “else” besides the true condition, then the program will execute the action to state that the player needs to create a stronger password. Consider the following example:

| Scratch | Snap |
| --- | --- |
|  |  |
| Source: https://scratch.mit.edu/?scrlybrkr=eea9261a | Source: https://snap.berkeley.edu/ |

**[5.AP.2a]** Building on prior programming skills, this standard introduces nested conditional control structures to increase complexity and interactivity in block-based programming. Nested conditionals allow programs to make multiple, layered decisions, enabling more responsive behavior based on user input or stored data. Students are expected to describe the concept of a nested conditional and begin incorporating it into algorithm design to support more dynamic and functional applications.

- Nested conditional control structures are conditional statements placed inside the body of another. Whereas conditional control structure statements are written as if / else statements, nested conditional control statements add a layer of complexity by building multiple layers of decision making into the process. These statements are constructed as if / else and if / else; if / else. Students use nested conditionals to make decisions that depend on more than one condition. Consider the following example in which a student is trying to decide which ride to go on at a theme park:
  - First question: Is the line for the roller coaster short?
    - If “yes,” then go on the roller coaster.
    - If “no,” ask a second question such as, “Is the line for the Ferris wheel short?”
    - If the answer is “yes,” go on the Ferris wheel.
    - If the answer is “no,” go grab a snack instead.

In programming, nested conditional controls allow for more complex decision-making. They allow a program to check multiple conditions in a sequence only after the first condition is met as true or false. Then, they allow the program to make decisions based on different levels of criteria and handle situations with many possible outcomes. They are read as, “If condition A is true, then check condition B.” The program continues to check each layer of a condition, allowing for multiple layers of decision making. Consider the following example in pseudocode:

```
BEGIN
   SET password TO user input
IF length of password is greater than or equal to 8 THEN
      // Step 2: Check if password has letters and numbers
      IF password contains at least one letter AND password contains at least one number THEN
         // Step 3: Check if password has special characters
         IF password contains at least one special character THEN
            PRINT "Your password is strong!"
         ELSE
            PRINT "Your password is medium strength (add special characters to make it stronger)."
         END IF
      ELSE
         PRINT "Your password is medium strength (make sure it has both letters and numbers)."
      END IF
   ELSE
      PRINT "Your password is weak (it needs to be at least 8 characters long)."
   END IF
END
```

*Pseudocode example adapted from commonly available online examples.*

**[5.AP.2b]** Planning is essential in programming and consists of the development of design documents that outline the steps and/or processes required to create a product, including the problem statement, input and output requirements, and algorithmic procedures. As students generate and organize ideas and algorithms using design documents (e.g., planning, drafting, revising, editing), they learn to develop and organize their algorithms. Students should be given a variety of opportunities to plan (e.g., outline, brainstorm) and draft (e.g., compose a first draft anticipating mistakes and corrections) using plain language, pseudocode, or diagrams to create their design documents. They also revise (e.g., add, remove, or rearrange ideas) and edit (e.g., proofread for grammar, punctuation, and spelling) their design documents to ensure that the algorithm is clear, sequential, and that their variables work as intended. Planning can be done with or without a computing device.

Design documents are a planning document that outlines the steps and/or processes required to create a product, including the problem statement, input and output requirements, and algorithmic procedures. Design documents consist of flow charts, diagrams, or story maps that students can use to brainstorm the algorithm they want to create.

- Plain language is a description of the steps and logic in simple terms that anyone can understand through the use of familiar analogies, real-life examples, and simple terms. The definition of "variable" above uses plain language and uses the analogy of a container to explain how variables store and hold values in a program.

  - Technical definition of a variable - Variables are containers for storing values, which can be changed during the program.
  - Plain language example - A variable is like a box where you can store something, like a number or a word. You can put different things in the box while your program is running, and you can even change what's inside the box if you need to.

- Pseudocode is simple, easy to read words that algorithms and programs are written in that allows students to write down the logic without getting bogged down by syntax rules. Pseudocode is often easier to read and understand than actual code. The following code uses the if / else conditional statement with a loop to check whether the password is strong or weak. Its user input requires the user to enter their password, resulting in an output where the program gives feedback based on the user's input.

```
START

  DISPLAY "What is your password?"

  REPEAT
    GET user input for password

    IF length of password >= 8 THEN
      DISPLAY "Your password is strong!"
    ELSE
      DISPLAY "Your password is weak. Try again."
    END IF
  UNTIL length of password >= 8

END
IF
```

*Pseudocode example adapted from commonly available online examples.*

Diagrams are visual representation of data, information, or concepts. Visual representations, such as flow charts, illustrate how the algorithm works. The following diagram is a flowchart.

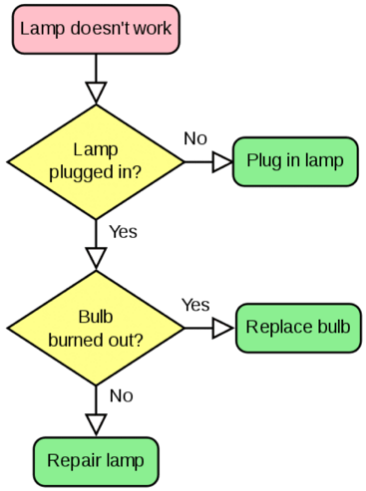



Image source: Flowcharting in Business Project Management

Students create design documents to trace how the algorithm processes input data to produce output and to note how variables change throughout the document. As they trace the document, they are able to predict the behavior of the algorithm (Mathematics 5.PFA.1.abc). In the diagram example above, students recognize that the variable is the password entered into the code, and they follow the rest of the steps to predict which direction the input (password) will follow. They are then able to predict the outcome based on the user input (password), a strong or a weak password.

**[5.AP.2c]** More complex algorithms will include nested conditional control structures [Computer Science 5.AP.2a]. To read, write, and interpret nested conditional control structures effectively, a student needs to understand how nested "if" statements work and how to follow the logic step-by-step to figure out which condition is checked first, and how decisions are made based on the results of each condition. Consider the following code for programming a bot to demonstrate the steps a student can follow to read, write, and interpret nested conditional control structures:

```
IF obstacle == "no"
   IF battery > 20
```

```
        PRINT "The robot can move forward."
    ELSE
        PRINT "The robot needs to charge."
ELSE
    PRINT "There is an obstacle. The robot cannot move forward."
```

*Pseudocode example adapted from commonly available online examples.*

To be able to read the above code, a student must follow these steps:

- First, identify the first outer condition in a code. The first “if” statement: IF obstacle == "no". (“if obstacle is equal to no") is where the computer begins making decisions. In this case, it must be determined if there is an obstacle in the bot’s way.
- If there is no obstacle, the student reads the code to determine that the next step is to check the battery level. IF battery > 20 (“if battery is greater than 20”)
- If there is an obstacle, the student reads the code to determine that they can skip checking the battery level and have the bot move forward.
- Next, identify the inner condition. In this code, the inner condition is nested in the first condition and contains the “else” statements. A student reads that there is no obstacle and begins to check the battery level. Upon checking the battery, the student determines if the bot can move forward or if the bot needs to be charged.

Teachers can support students’ understanding of nested conditional control structures by providing multiple models for analysis. Examples may include determining a letter grade based on numerical scores, selecting an outfit based on weather and precipitation, classifying animals by ecological niche, or identifying numbers as prime or composite. These varied contexts help illustrate how nested conditionals function across domains.

Engaging in these tasks promotes decomposition, as students must break down complex problems into smaller, manageable conditions. The process also involves abstraction, as students identify and focus on the most relevant details necessary to build a logical and efficient sequence of steps within their algorithm.

By examining multiple examples of nested conditional control structures, students develop familiarity with common patterns and logic flows. This exposure supports both imitation—applying similar structures in their own algorithms—and interpretation, as students trace the logic to predict outcomes and evaluate program behavior.

## Concepts and Connections

### CONCEPTS

Nested conditional control structures enable complex decision-making by allowing conditions to be evaluated within other conditional branches. Developing design documents using plain language, pseudocode, or diagrams supports logical planning and prediction of algorithm behavior. Proficiency in reading, writing, and interpreting "if-else" and nested "if-else; if-else" statements is essential for implementing dynamic and responsive program logic.

### CONNECTIONS

**Within the grade level/course:** At this grade level, students build upon their algorithmic design by describing, reading, writing, and interpreting nested conditional control structures (5.AP.2).

**Vertical Progression:** In Grade 4, Students built upon their block-based programming language by adding loops, variables, and user-input to their events and using conditional control structures (4.AP.2). In Grade 6, students will incorporate a collection of numerical data such as Boolean, integer, and decimal number variables in order to predict the results of logic expressions that use Boolean operators in their block or text-based programming (6.AP.2).

### ACROSS CONTENT ALIGNMENT

**English**

- **5.RI.2A** The student will use textual evidence to demonstrate and build knowledge from a variety of grade level complex informational texts heard or read by describing the overall organization patterns of texts (e.g., cause/effect, comparison/contrast, problem/solution, description, sequence, and chronological) and how each successive part builds on earlier sections, using available transitional words and phrases. *(Note: reading grade level texts about CS content)*
- **5.W.1A** Write personal or fictional narratives in prose or poetic form that organize the writing around a central problem, conflict, or experience using descriptions or dialogue to develop the experience(s).
- **5.W.1B** Write expository texts to examine a topic and convey ideas that develops the focus with relevant facts, concrete details, or examples from multiple sources and are grouped logically.
- **5.W.1C** Write persuasive pieces on topics or texts, including media messages, supporting a clear perspective with adequate facts, reasons, and logically grouped information.
- **5.W.1D** Write in response to texts read (including summaries, reflections, and descriptions) in which students demonstrate their thinking with details, examples, and other evidence from the text that are logically grouped.

**History and Social Science**

- **USI.7** The student will apply history and social science skills to describe the challenges faced by the new nation by

- **USI.8** The student will apply history and social science skills to explain westward expansion and reform in America from 1801 to 1861.

### Mathematics

- **5.CE.1b** The student will estimate, represent, solve, and justify solutions to single-step and multistep contextual problems using addition, subtraction, multiplication, and division with whole numbers.
- **5.CE.2** The student will estimate, represent, solve, and justify solutions to single-step and multistep problems, including those in context, using addition and subtraction of fractions with like and unlike denominators (with and without models), and solve single-step contextual problems involving multiplication of a whole number and a proper fraction, with models.
- **5.CE.4** The student will simplify numerical expressions with whole numbers using the order of operations.
- **5.MG.1** The student will reason mathematically to solve problems, including those in context, that involve length, mass, and liquid volume using metric units.
- **5.MG.2ad** The student will use multiple representations to solve problems, including those in context, involving perimeter, area, and volume.
- **5.PS.3ab** The student will determine the probability of an outcome by constructing a model of a sample space and using the Fundamental (Basic) Counting Principle.
- **5.PFA.1a** The student will identify, describe, extend, and create increasing and decreasing patterns with whole numbers, fractions, and decimals, including those in context, using various representations.
- **5.PFA.2** The student will investigate and use variables in contextual problems.

### Science

- **5.1** The student will demonstrate an understanding of scientific and engineering practices by a) asking questions and defining problems; b) planning and carrying out investigations *5.1 standard is integrated within science content and not taught in isolation. Potential science concepts to apply 5.1 include: 5.3 (force, mass, energy transfer), 5.5 (sound), 5.6 (light), 5.8 (rocks).*

## DIGITAL LEARNING INTEGRATION

- **3-5.ID** Students use a variety of technologies, including assistive technologies, within a design process to identify and solve problems by creating new, useful or imaginative solutions or iterations.
  A. Know and use appropriate technologies in a purposeful design process for generating ideas, testing theories, creating innovative digital works or solving authentic problems.
  C. Use appropriate technologies to develop, test, and refine prototypes as part of a cyclical design process.

## Opportunities for Integration

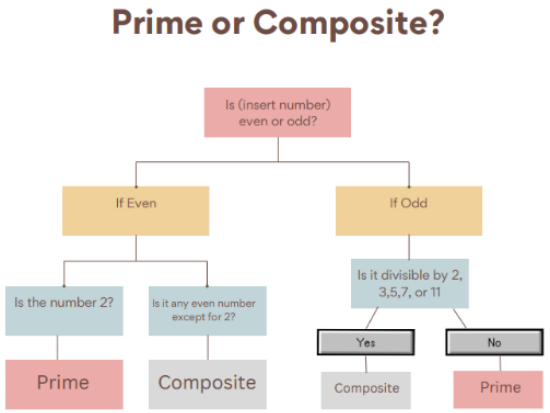
Curriculum integration strengthens conceptual understanding and skill application. This can be done through multidisciplinary, interdisciplinary, and transdisciplinary approaches to integration. The examples below illustrate multiple ways to integrate computer science.

### English

- Students use their knowledge of organizational patterns and key transitional words and phrases to write paragraphs that model the different patterns.

### Mathematics

- Students create a design document with nested control structures to help them determine if a number is prime or composite. They implement the use of a variable when they apply different divisibility rules to the number in order to determine its outcome (prime or composite). By way of example:




### Science

- Students implement an algorithm in the form of a provided slime recipe in order to examine properties of matter. They make observable changes to the recipe in order to alter the state of the slime. They introduce a “loop” into the recipe by implementing an independent variable (adding additional saline solution) until they create a slime that acts more like a solid than a liquid.

## Skills in Practice

Students should engage in the following practices to deepen their conceptual understanding and enhance the application of skills aligned with the Computer Science *Standards of Learning*. These practices are explained in more detail in Appendix A.

### B. Fostering Computational Thinking Practices:

1. Decompose Real-World Problems
2. Explore Common Features and Identify Patterns
3. Use Abstraction to Simplify, Represent, and Problem Solve
4. Apply Algorithmic Thinking to Problem Solve and Create
5. Apply Computational Thinking Practices to Select, Organize, and Interpret Data

### C. Fostering Iterative Design Practices:

1. Identify, Define, and Evaluate Real-world Problems
2. Plan and Design Artifacts
3. Create, Communicate and Document Solutions
4. Test and Optimize Artifacts

## 5.AP.3 The student will use the iterative design process to create, test, and debug programs containing sequencing, loops, variables, user inputs, nested conditional control structures, and two-way branching conditional control structures in a block-based programming tool.

a. Use accurate terminology to describe and explain the iterative design process.
b. Create and test programs that consist of sequencing, loops, variables, user inputs, nested conditional control structures, and two-way branching conditional control structures.
c. Trace and predict outcomes of programs.
d. Analyze and describe program results to assess validity of outcomes.
e. Revise and improve programs to resolve errors or produce desired outcomes.

## Understanding the Standard

**[5.AP.3a]** The iterative design is a systematic approach to creating and refining products, systems, or solutions through repeated cycles of design, evaluation, and improvement. This is an essential process used in programming that provides students with hands-on experience in problem solving. It consists of the refinement and improvement of code through the application of a repeated cycle. By engaging in the iterative process, students learn that programming is not a one-step process. It's about continuously improving, testing, and refining based on user experience and results.

The iterative design process includes the following: problem definition, design, implementation or development, testing, evaluation, and iteration.

- Problem definition involves clearly identifying the problem or challenge that needs to be solved. Students can identify problems that they desire to solve in the games, apps, animations, websites or other programs that they create.
- Design involves creating a plan or prototype of the proposed solution. This may include creating a design document to outline the steps needed to solve the problem in the program [Computer Science 5.AP.2].
- Implementation or development involves building a solution through the use of a functioning product or prototype.
- Testing involves running the program to see if the program works as expected and to identify areas requiring improvement.
- Evaluation involves reviewing the test results and feedback to assess the product's effectiveness and further identify the changes that need to be made.
- Iteration involves repeating the cycle as many times as necessary to refine the solution based on feedback and testing.

**[5.AP.3b]** Students apply the iterative design process as they use block-based programming, and the criteria defined in (Computer Science 5.AP.2) to create and test their programs. Building upon previous programming concepts and skills such as sequences, loops, variables, user inputs, and nested conditional control structures, students will add two-way branching conditional control structures within their programming skillset.

Nested conditional controls and two-way branching controls are closely related, but a two-way branching control builds on the if / else structure of the nested conditional controls.

- Nested conditional control involves multiple levels of decision-making, where conditions are layered within each other to refine outcomes based on more than one decision point.
- Two-way branching conditional control structures is a programming concept that allows a program to make a decision based on two different paths. This works like an if / else statement. If a specified condition is true, the program executes one set of instructions; otherwise, it follows an alternative path. This structure enables programs to respond dynamically to different inputs or scenarios, making it fundamental for decision-making in code. Understanding this concept helps students begin to build more interactive and responsive programs.Consider this example where a student is answering questions on an application for twelve-year-old students to attend Space Camp.
    - The first question asks, “Will you be twelve on or before the due date of the application?”
    - If the student answers, “Yes,” then they are moved to the next question on the application.
    - If the student answers, “No,” they are taken to a screen that asks them to try again when they meet the age requirement.

Two-way branching and nested conditionals are both programming structures used to control the flow of a program based on decision-making. Two-way branching, typically written as an *if/else* statement, evaluates a single condition and chooses between two possible outcomes. In contrast, nested conditionals involve placing one conditional inside another, allowing for more complex, layered decisions based on multiple criteria. While both use Boolean logic to evaluate conditions and direct program behavior, nested conditionals offer greater flexibility for handling multi-step logic, whereas two-way branching is suited for simpler, binary choices.

**[5.AP.3c]** Students practice algorithmic thinking in the iterative design process as they trace their programs. In programming, tracing refers to the process of following the flow of an algorithm or program step by step to track how variables change throughout execution. This is done by systematically analyzing the program's logic, either manually or using debugging tools, to predict variable values and understand how data is manipulated at different stages.

- A recommended approach in tracing the flow of an algorithm or program includes the following:
    1. Read the entire program to understand its overall structure and purpose.
    2. Identify the variables used and note their initial values.
    3. Start at the first line of code and move line by line in order the program runs.
    4. Record the value of each variable as it changes after every line of execution.
    5. Follow control structures like conditionals (e.g., *if/else*) and loops, and note which path or iteration is taken based on the conditions.
    6. Predict the program’s output based on the variable values and logic you’ve traced.
    7. Compare traced output to the actual result when the program runs to check for accuracy and identify any logic or syntax errors.

**[5.AP.3d]** After the execution of a program, the output should be analyzed to determine whether the results align with the expected outcomes. This process involves assessing the program's correctness, identification of errors or unexpected behaviors, such as variables not updating properly or conditions failing to respond and recognizing patterns that indicate flaws in the program's logic. This iterative process reinforces the importance of refining and improving computational solutions.

When algorithms do not function as intended, a systematic review is required to identify the issue, trace the sequence of flawed steps, and determine potential solutions. Adjustments may involve modifying variable assignments, restructuring control flow, or refining conditional statements to achieve the desired outcome. Once adjustments are identified they are implemented and tested, reinforcing the iterative design process as a method of continuous improvement in programming.

This is where students assess whether the program is working correctly, and identify any errors or unexpected behavior (e.g., variables not updating correctly, conditions not triggering as expected).

Two main types of errors students need to debug in their algorithms include syntax and logic errors.

- Syntax is the rules or structure of a programming language. Syntax errors occur when there's a mistake in the code's grammar preventing the program from running. They appear more frequently in text-based coding and can include typing mistakes like missing parentheses, commas, or incorrect keywords. Block-based programming reduces the chance of syntax errors because the blocks only fit together in the correct way. The program is automatically structured properly and reduces the risk of common syntax errors.
- Logic refers to the set of rules and reasoning that determine how a program makes decisions and controls the flow of execution. It involves using conditionals (like if statements), comparisons, and Boolean expressions (true/false values) to guide what actions the program takes under different circumstances. Logic errors are errors that occur when a program is executed and produces incorrect or unintended output without causing the program to crash or display an error message. They are analogous to giving a person directions that they follow exactly, but they end up in the wrong location because the directions are incorrect. Consider the following logic error examples:

| Text-Based Programming Logic Errors | Block-Based Programming Logic Errors |
|---|---|
| • Incorrect comparisons in an "if" statement<br>• Forgetting to update a variable<br>• Using the wrong variable<br>• Using loops incorrectly | • Using the wrong block sequences<br>• Using the wrong block for the job<br>• Using the wrong variables<br>• Using an incorrect condition in if / else blocks |

*[Please note: Identifying and addressing syntax and logic errors is not within the scope of the Grade 5 student learning objectives. However, this expectation is introduced by Grade 7. The information provided here is intended solely to support educator understanding and instructional context..]*

**[5.AP.3e]** Revising and improving an algorithm is a key part of the debugging process, where students identify and fix problems in their code to ensure it works correctly. Debugging is the process of identifying, isolating, and resolving errors, commonly known as "bugs", within a set of instructions, code, or a system. This critical skill allows students to analyze unexpected behavior, correct flaws in logic, and refine their code to ensure the program operates as intended and meets its defined objectives. Effective debugging involves systematically tracing variable changes, examining control flow, and using debugging tools to pinpoint errors.

- For example, while testing a program, students notice that a loop runs too many times, causing incorrect output. To debug the issue, they analyze the loop condition and identify that a variable controlling the number of iterations is not updating correctly. Students revise the condition to ensure the loop executes as expected, improving both accuracy and efficiency.

Consider the following example checklist for debugging a program:

1. Syntax Errors
    - Are there any typos (spelling mistakes, incorrect keywords)?
    - Are parentheses, braces, and quotes properly placed?
    - Is there correct indentation or block structure?
2. Logic Errors
    - Does the program do what I expect? (i.e. correct math? right output?)
    - Are the if condition (if-statements) correct?
    - Are loops running too many or too few times?
3. Variables
    - Is the variable name spelled correctly?
    - Is the variable being used consistently throughout the code?
    - What is stored in the variable?
    - Is it the correct type of value?

They revise the code to improve clarity, efficiency, or to meet the desired outcomes (e.g., fixing a loop that runs too many times or adjusting a condition to trigger at the right time). Students continue the iterative process until the program produces correct and valid results as intended.

## Concepts and Connections

### Concepts

The iterative design process is a cyclical approach to program development involving planning, testing, evaluation, and refinement, and should be described using accurate technical terminology. Through iterative development, students create and test programs that incorporate sequencing, loops, variables, user input, and both nested and two-way conditional control structures. This process includes tracing program logic, predicting outcomes, analyzing results for correctness, and revising code to correct errors and optimize functionality.

### Connections

**Within the grade level/course:** At this grade level, students expand their knowledge of the iterative design process as they use accurate terminology to describe and explain it. Their programs include two-way branching conditional control structures, and they trace and predict the outcomes of programs so that they can revise and improve the programs by debugging them (5.AP.3).

**Vertical Progression:** In Grade 4, students used block-based programming to explore user inputs and to create and use variables for the purpose of storing and processing data. They further explored variables by tracing and predicting the value of the included variables as part of the iterative design process (4.AP.3). In Grade 6, students will create and test block or text-based programs that use multiple conditional control structures, incorporate existing code, media, or libraries into their existing programs, and incorporate feedback in order to refine their programs as part of the iterative design process (6.AP.3).

### Across Content Areas

**English**

- **5.W.2A** The student will compose various works for diverse audiences and purposes, linked to grade five content and texts by engaging in writing as a process to compose well-developed paragraphs.
- **5.W.3A**The student will compose various works for diverse audiences and purposes, linked to grade five content and texts. With guidance and support from peers and adults, develop and strengthen writing as needed by revising for quality of ideas, organization, sentence fluency, and precise word choice.

**History and Social Science**

- **USI.7** The student will apply history and social science skills to describe the challenges faced by the new nation.

**Mathematics**

- **5.PFA.1** The student will identify, describe, extend, and create increasing and decreasing patterns with whole numbers, fractions, and decimals, including those in context, using various representations.

**Science**

- **5.1** The student will demonstrate an understanding of scientific and engineering practices by a) asking questions and defining problems; b) planning and carrying out investigations; c) interpreting, analyzing, and evaluating data; e) developing and using models; f) obtaining, evaluating, and communicating information *5.1 standard is integrated within science content and not taught in isolation. Potential science concepts to apply 5.1 include: 5.4 (rocks) and 5.8 (rocks)*

### DIGITAL LEARNING INTEGRATION

- **3-5.ID** Students use a variety of technologies, including assistive technologies, within a design process to identify and solve problems by creating new, useful or imaginative solutions or iterations.
  A. Know and use appropriate technologies in a purposeful design process for generating ideas, testing theories, creating innovative digital works, or solving authentic problems.
  C. Use appropriate technologies to develop, test, and refine prototypes as part of a cyclical design process.

## Opportunities for Integration

Curriculum integration strengthens conceptual understanding and skill application. This can be done through multidisciplinary, interdisciplinary, and transdisciplinary approaches to integration. The examples below illustrate multiple ways to integrate computer science.

### English

- Students engage in the writing process by brainstorming, drafting, revising, and editing narrative, expository, or persuasive texts. During the revision and editing phase, they may use a checklist to systematically review their drafts, identify and correct errors, and make purposeful changes that enhance clarity, coherence, and overall quality before producing a final draft.

### Mathematics

- Students solve contextual problems by following a structured, iterative process: they identify and analyze known information, develop a solution plan by selecting operations and estimating outcomes, carry out the plan while showing their thinking, and evaluate the reasonableness of their solution to ensure it answers the original question.

## Skills in Practice

Students should engage in the following practices to deepen their conceptual understanding and enhance the application of skills aligned with the Computer Science *Standards of Learning*. These practices are explained in more detail in Appendix A.

## C. Fostering Iterative Design Practices:

1. Identify, Define, and Evaluate Real-world Problems:
2. Plan and Design Artifacts
3. Create, Communicate and Document Solutions
4. Test and Optimize Artifacts

Back to Algorithms and Programming (AP)

## Computing Systems (CSY)

### 5.CSY.1 The student will explain how computing systems are used to collect and exchange data.

a. Identify and explain how computing systems store data representations, including images and sound.
b. Describe the role of processing speed and storage capacity when collecting and exchanging data.

## Understanding the Standard

A computing system is an integrated group of hardware (the physical components like the processor, memory, and input/output devices) and software (the programs and operating systems) that work together to perform various tasks.

Computing systems perceive the world by using sensors. Sensors are a computing component that detects, collects, or measure data that would otherwise be difficult to gather manually. Computers use sensors and other input devices to collect information just like a human uses their senses to understand their surroundings.

**[5.CSY.1a]** Computing devices use inputs and outputs to collect and exchange data. In computer science, input and output may be referred to as I/O is the communication between the information processing system and a human being or another computing device. Computer hardware and software help the collection and exchange more user friendly.

The use of computing systems to collect and exchange data improves efficiency and accuracy in the data collection process. This is achieved through the use of input devices. Input devices are the hardware components that allow users to enter data into a computing device and can include keyboards, sensors, cameras, and microphones to capture various types of data. These devices enable the collection of different data representations, including text, images, and sound.

When a computing system collects data, it may utilize input devices. Sharing and recalling the data may require the use of output devices, which include monitors, speakers, and printers. When data representations are stored or exchanged, the real-world information must be digitized or translated to a language that all computer systems understand. This typically refers to the binary number system, which uses only two digits: 0 and 1.

When computers systems are used to collect data, the method and tools of collection are based on the type of data and the type of systems being utilized. Data may be stored locally on a hard drive on the device or on the Internet with cloud computing. The data will need to be accessible at a later date and time without any change to the data representation. It is considered good practice to store important data in a secure format and place; as well as the need to backup up data to protect against loss.

Not all types of data are stored the same way. The following table shows how images and sound are stored by computers:

| Storing Data | |
|---|---|
| **Images** | **Sound** |
| • Computers store images as a colorful grid made up of tiny squares called pixels.<br>• Each pixel has its own color, and when all the pixels are put together, they make up the full image.<br>• Each color is stored as a specific binary code which helps the computer recall the information later.<br>• The computer stores information about each pixel's color, which helps it remember the picture and show it to the user later. | • Computers store sound waves as numbers.<br>• These numbers represent things like pitch (high or low sounds) and volume (loud or soft sounds).<br>• Each number is translated into binary code.<br>• When the sound is played, the computer turns the numbers back into sound waves that users can hear |

**[5.CSY.1b] Processing** speed and storage capacity both impact collecting and exchanging data.

- Processing speed is how fast a computer can take in information, think, and complete tasks. Processing speed helps collect data quickly and send and receive data fast. Faster processing speeds allow computers to accomplish more work in less time.

- Storage capacity is the amount of space available to save data, which includes photos, sound, and files. Everything that is stored on a computer takes us space in the storage capacity of the device. With appropriate storage capacity, files can be stored and organized more easily. When there is less space, computing systems may slow down or even stop working.

When computing systems have both a large storage capacity and high processing speed users are able to accomplish tasks such as sharing videos and playing video games with fewer errors.

## Concepts and Connections

### CONCEPTS

Computing systems store data representations such as images and sound using binary encoding formats that translate visual and auditory information into digital form. Processing speed and storage capacity play critical roles in determining how efficiently data can be collected, stored, and exchanged, directly impacting system performance and user experience.

### CONNECTIONS

**Within the grade level/course:** At this grade level, students apply computational thinking practices and are able to better understand the role of processing speed and storage space on how information is processed (5.CSY.1)

**Vertical Progression:** In Grade 4, students explored how computing systems operate (4.CSY.1). In Grade 6, students will look at operating systems and other applications found within a computing system (6.CSY.1).

### ACROSS CONTENT AREAS

**Mathematics:**

- **5.PFA.1** The student will identify, describe, extend, and create increasing and decreasing patterns with whole numbers, fractions, and decimals, including those in context, using various representations.

**Science:**

- **5.6d** The student will investigate and understand that visible light has certain characteristics and behaves in predictable ways. Key ideas include radiant energy can be transformed into thermal, mechanical, and electrical energy.

### DIGITAL LEARNING INTEGRATION

- **3-5.CC** Students communicate clearly and express themselves creatively for a variety of purposes using appropriate technologies (including assistive technologies), styles, formats, and digital media appropriate to their goals.
  C. Communicate complex ideas clearly and effectively by creating or using a variety of digital objects such as visualizations, models, or simulations.

## Opportunities for Computer Science Integration

Curriculum integration strengthens conceptual understanding and skill application. This can be done through multidisciplinary, interdisciplinary, and transdisciplinary approaches to integration. The examples below illustrate multiple ways to integrate computer science.

**English**

- Students may explore how words are decoded by associating sounds with letter. Students can explore how computers utilize binary code to represent information and then must decode the information by knowing what each combination of zeros and ones represents in the alphabet. Students may wish to try writing a vocabulary word using binary code and asking a friend to decode the information.

### Science

- Students can explore how light and sound travel and compare to how computing systems collect and exchange information. Students may make observations about the similarities and differences in these processes.

## Skills in Practice

Students should engage in the following practices to deepen their conceptual understanding and enhance the application of skills aligned with the Computer Science *Standards of Learning*. These practices are explained in more detail in Appendix A.

### B. Fostering Computational Thinking Practices:

2. Explore Common Features and Identify Patterns
5. Apply Computational Thinking Practices to Select, Organize, and Interpret Data

## 5.CSY.2 The student will describe an automated decision-making process employed by a computing system.

a. Explore decision automation and how it is used.
b. List outcomes of a process based on automated decisions.

## Understanding the Standard

**[5.CSY.2a]** Automated decision-making occurs when a computer uses predefined algorithms or programs to make decisions independently based on the information it receives. The quality of the input data can significantly influence the decision-making process. This process follows a set of rules or steps, allowing the computer to determine its next actions without human intervention.

The special instructions that computers follow to make these decisions are called algorithms. An algorithm is a finite, specified set of step-by-step instructions designed to solve a problem or perform a task. They tell the computer exactly what to do in different situations, which allows it to appear as though a computer is able to make decisions on its own.

**[5.CSY.2b]** Automated decision making is the use of predefined algorithms or programs that enable a computing device the ability to make decisions independently based on the information it receives. Automated decisions make roads safer with traffic control, maintain comfortable temperatures in homes while being energy efficient, provide better suggestions to movies and music, organize emails, and can provide transportation with self-driving cars. Consider the following examples of real-life automated decision-making:

- Traffic lights have sensors that detect when cars are waiting. The traffic light system uses an automated decision-making process to decide when to turn the light green, yellow, or red. It follows a set of rules to make sure cars take turns safely.
- Smart thermostats automatically adjust the temperature in a house by using sensors to detect things like temperature or human activity. The thermostat uses the data to make decisions based on pre-determined rules (if / then statements). Thermostats can also make decisions based on what time of day it is or if people are home, making the room warmer or cooler as needed.
- Spam filters in email programs use filters that look for key words and phrases, the sender's address, email behavior (like did it go to a lot of people). If an email meets more than one of these requirements, it is often sent to a spam folder.
- Recommendation systems in apps like YouTube or Netflix. The system looks at shows you have watched, ratings or feedback, and behavioral patterns (like only watching the news at 1 time of day). The system uses this information to make suggestions about what else you might like.

In school settings, students may experience outcomes based on automated experiences when using a library check out system, when making a lunch choice, or when using online learning software. Consider the following examples of school based automated decision making:

- Library check out systems use automated experiences to send overdue notices to specific students when books have not been returned by their due dates.
- Lunch choice systems allow the cafeteria to be notified with the specific number of each choice needed and can also take into account the number of students present in school on a specific day.
- When student leaning software suggests going to the next level after getting a predetermined number of correct answers on the current level, this is an automated experience.

Automated decision-making helps computers work faster and make decisions without needing help. It also keeps things consistent because the computer will always follow the same rules.

## Concepts and Connections

### Concepts

Decision automation involves the use of computing systems to make rule-based or data-driven decisions without human intervention, streamlining processes across various domains. Understanding automated decision-making includes identifying common applications and predicting possible outcomes generated by these automated processes.

### Connections

**Within the grade level/course:** At this grade level, students are exploring automation within computing systems and how humans make decisions (5.CSY.3).
**Vertical Progression:** In Grade 4, students explored the types of machine learning (4.CSY.3). In Grade 6, students will explore Artificial Intelligence (6.CSY.3).

### Digital Learning Integration

- **3-5.DC** Students develop and employ strategies for understanding and solving problems in ways that leverage the power of technological methods, including those that leverage assistive technologies, to develop and test solutions.
  D. Understand how automation works and use algorithmic thinking to develop a sequence of steps to create and test automated solutions.

## Opportunities for Computer Science Integration

Curriculum integration strengthens conceptual understanding and skill application. This can be done through multidisciplinary, interdisciplinary, and transdisciplinary approaches to integration. The examples below illustrate multiple ways to integrate computer science.

### English

- Students learn how to navigate the school library's digital catalog or an on-line library collection by using keywords instead of exact titles or author names. Using predetermined topics or themes, they practice identifying effective search terms and locating relevant books. Students then write step-by-step instructions to guide others in finding similar books using keyword searches. This activity builds research skills while reinforcing algorithmic thinking and the importance of precise search strategies.

### Mathematics

- Students complete a mystery picture activity using a spreadsheet program, where solving math problems correctly reveals parts of a hidden image. This task helps students experience how computers make decisions using formulas and conditional logic. After completing the activity, students explore the formulas behind the image reveal and reflect on how this process models the use of algorithms that automate tasks in everyday life. This activity supports mathematical reasoning while introducing foundational computer science concepts in an applied, visual context.

## Skills in Practice

Students should engage in the following practices to deepen their conceptual understanding and enhance the application of skills aligned with the Computer Science *Standards of Learning*. These practices are explained in more detail in Appendix A.

### B. Fostering Computational Thinking Practices:

1. Decompose Real-World Problems
2. Explore Common Features and Identify Patterns
3. Use Abstraction to Simplify, Represent, and Problem Solve
4. Apply Algorithmic Thinking to Problem Solve and Create
5. Apply Computational Thinking Practices to Select, Organize, and Interpret Data

### C. Fostering Iterative Design Practices:

1. Identify, Define, and Evaluate Real-world Problems
2. Plan and Design Artifacts
3. Create, Communicate and Document Solutions
4. Test and Optimize Artifacts

## 5.CSY.3 The student will evaluate and implement troubleshooting strategies when a computing system is not operational.

a. Identify and use troubleshooting protocols to resolve hardware, software, and connectivity issues.
b. Apply prior troubleshooting practices to new problems as they arise.

## Understanding the Standard

**[5.CSY.3a]** As with any system there are times that a computer system does not work as intended. Computing systems are composed of an interconnected system of hardware and software.

Computer hardware consists of the physical components of a computing device that you can touch, such as the processor, memory, keyboard, and display. Computer software is a set of instructions that tells the computer how to act and respond but cannot be seen or touched. In the age of cloud computing, storing and accessing information on the Internet, connectivity may also impact the functionality of a computing system. Connectivity is when different computing devices can communicate and/or share information between each other.

Troubleshooting strategies for hardware, software, and connectivity should utilize systematic procedures or protocol. Troubleshooting involves identifying and correcting faults in a computing system. These procedures frequently include if / then process to help determine the problem and the solution.

Hardware Troubleshooting Protocol Example:

- Check power and connections (plugged into power, power turned on, screen turned on).
- Check cables and connections (cords completely attached, remove extra devices, battery check for portable devices).
- Restart the computer.
- Listen for sounds.
- Check basic settings (screen brightness, volume, Internet connection).

Software Troubleshooting Protocol Example:

- Identify the problem (software or app name and error messages).
- Check Internet connection.
- Restart the software or app.
- Restart the computer.
- Check for updates.
- Check for issues on other devices using the same program or app.

Connectivity Troubleshooting Protocol Example:

- Check the Wi-Fi Connection (reconnect if needed).
- Check Airplane Mode (this should be turned off).
- Restart the device.
- Check other websites or apps.
- Check for issue on other devices using the same Wi-Fi connection.
- Move closer to Wi-Fi.
- Disconnect and Reconnect to the network

## Concepts and Connections

### CONCEPTS

Troubleshooting in computing involves using structured methods to diagnose and resolve hardware, software, and connectivity issues. By applying previously learned strategies, students develop the ability to transfer and adapt effective solutions to new and unfamiliar technical problems.

### CONNECTIONS

**Within the grade level/course:** At this grade level, students are evaluating and implementing troubleshooting strategies for issues with computing systems. Students are also applying troubleshooting protocols for new problems (5.CSY.3).
**Vertical Progression:** In Grade 4, students applied troubleshooting strategies when a computing system is not working as intended (4.CSY.2). In Grade 6, Students will identify and explain problems with hardware, software, and connectivity and their solutions as well as available resources for troubleshooting these issues (6.CSY.2).

### ACROSS CONTENT AREAS

**Mathematics**

- **5.CE.4** The student will simplify numerical expressions with whole numbers using the order of operations.
- **5.MG.2** The student will use multiple representations to solve problems, including those in context, involving perimeter, area, and volume.

### DIGITAL LEARNING INTEGRATION

- **3-5.EL** Students communicate clearly and express themselves creatively for a variety of purposes using appropriate technologies (including assistive technologies), styles, formats, and digital media appropriate to their goals.

D. Understand the various fundamental concepts of technology operations, demonstrate the ability to choose, use, and troubleshoot technologies and transfer knowledge to explore emerging technologies.

## Opportunities for Computer Science Integration

Curriculum integration strengthens conceptual understanding and skill application. This can be done through multidisciplinary, interdisciplinary, and transdisciplinary approaches to integration. The examples below illustrate multiple ways to integrate computer science.

### English

- Students create help desk-style guides to support peers with common school or home technology challenges. After drafting their instructions, classmates test the guides to see if they can successfully follow the steps and resolve the problem. If issues arise, students work in pairs to revise and "debug" their documents for clarity and effectiveness. Final versions can be compiled into a class help desk book or shared more broadly with the school community, and students may also create instructional videos or offer peer tech support during homeroom time.

### Science

- Students engage in an iterative problem-solving process through investigations or design challenges. When initial solutions do not yield the desired results, they analyze outcomes, refine their approach, and repeat steps to improve effectiveness. This repeated refinement reflects the role of iteration in both scientific inquiry and engineering design. The process also parallels troubleshooting and debugging in computing, enabling students to make meaningful connections across disciplines and recognize iteration as a key strategy for improving solutions.

## Skills in Practice

Students should engage in the following practices to deepen their conceptual understanding and enhance the application of skills aligned with the Computer Science *Standards of Learning*. These practices are explained in more detail in Appendix A.

### B. Fostering Computational Thinking Practices:

1. Decompose Real-World Problems
2. Explore Common Features and Identify Patterns
3. Use Abstraction to Simplify, Represent, and Problem Solve
4. Apply Algorithmic Thinking to Problem Solve and Create
5. Apply Computational Thinking Practices to Select, Organize, and Interpret Data

## C. Fostering Iterative Design Practices:

1. Identify, Define, and Evaluate Real-world Problems
2. Plan and Design Artifacts
3. Create, Communicate and Document Solutions
4. Test and Optimize Artifacts

Back to Computing Systems (CSY)

## Cybersecurity (CYB)

### 5.CYB.1 The student will identify ways to limit unauthorized access on computing devices.

a. Define virus, malware, and phishing.
b. Explain how viruses and malware can put personal information at risk.
c. Describe the role of human interactions in social engineering attacks.
d. Identify ways to protect personal and private information when using a computing device and the Internet.
e. Explain the importance of updating software.

## Understanding the Standard

Computer networks, including the Internet, can connect people to other people, places, information, and ideas. Computing devices frequently store personal and private information of the user. When information is accessed without the permission of the owner, it is called unauthorized access. When unauthorized users access the information on a computing device, it can be changed, deleted, or even held for ransom, all of which are significant security problems for the rightful owner. Protecting from unauthorized access to computing devices and private information on those devices is the responsibility of everyone using Internet connected computing devices.

Malware, viruses, and phishing are three cyber threats that may result from unauthorized access to a device or information, these are not the only problems that may occur. Other threats, like ransomware or spyware are addressed in later grades.

**[5.CYB.1ab]**

- Malware is an umbrella term for any type of software designed to harm, exploit, or otherwise compromise computers and devices. It can corrupt computing devices, steal data, or result in ransomware. Malware includes:
  - Virus is a type of malicious software (or malware) designed to spread from one computer to another, often without the user's knowledge.
  - Worms are a type of malware designed to replicate itself and spread across computers or networks without needing to attach to a host program or file. Unlike a computer virus, which requires human interaction (e.g., opening an infected file), worms spread automatically and exploit vulnerabilities in systems.
  - Spyware is a type of malware designed to secretly monitor and collect information about a user's activities without their knowledge or consent. The collected data can include personal information, such as passwords, browsing habits, credit card details, and other sensitive data, which is often sent to a third party.

Viruses and malware are harmful programs that can infect a computer or device without the user's knowledge. Once installed, they can collect, steal, or damage personal information such as passwords, financial data, or private files. Some malware can track user activity or give unauthorized access to attackers. This puts individuals at risk of identity theft, financial loss, and privacy violations.

**[5.CYB.1c]** Social engineering is a method used by malicious individuals to trick people into giving away confidential information or access to secure systems.

Human interactions and emotions impact the effectiveness of social engineering attacks. Because they rely on the psychological manipulation of their victims rather than technical skills, they are often effective in gaining access to the information or secure system that is being pursued by the attacker. Consider the following examples:

- Social engineers often impersonate someone the victim knows and trusts. After making contact, the impersonator creates a sense of urgency or fear to “help” immediately.
- Social engineers may use peer pressure or mimic authority figures to get the victim to do what they want.

Phishing is a deceptive online attack where scammers pretend to be a trusted source, such as an individual or a business, and trick individuals to share personal identifiable information.

**[5.CYB.1d]** Protecting personal and private information when connecting a computing device to the Internet should be ongoing for everyone. The following basic steps can be very effective in the prevention of personal and private information getting to someone without authorization:

- Use strong passwords.
- Keep passwords private.
- Be cautious about sharing personal information.
- Use privacy settings.
- Think before you click.
- Use secure websites.
- Log out of devices when finished using them.

**[5.CYB.1e]** Updating software is another way to help keep personal and private information from falling into the wrong hands. Software updates fix weak holes that may allow unauthorized access to information.

- Use 2-Factor Authentication (2FA).
- Disable unnecessary features like Bluetooth and location services when not needed

## Concepts and Connections

### CONCEPTS

Viruses, malware, and phishing are malicious tactics used to compromise computing systems and access sensitive information. These threats can expose personal data, damage devices, or manipulate users into revealing private information. To mitigate risks, individuals should adopt protective measures to address security vulnerabilities.

### CONNECTIONS

**Within the grade level/course:** At this grade level, students explore ways in which data may be obtained from unauthorized access to computing devices (5.CYB.1).
**Vertical Progression:** In Grade 4, students explored appropriate and inappropriate use of computing devices (4.CYB.1). In Grade 6, students will evaluate the risks and benefits of sharing information (6.CYB.1).

### ACROSS CONTENT AREAS

**History and Social Science**

- **USI Skillsh** The student will apply history and social science skills to the content by engaging and communicating as a civil and informed individual with persons with different perspectives.

**Mathematics**

- **5.PS.1** The student will apply the data cycle (formulate questions; collect or acquire data; organize and represent data; and analyze data and communicate results) with a focus on line plots (dot plots) and stem-and-leaf plots.

### DIGITAL LEARNING INTEGRATION

- **3-5.DC** Students recognize the rights, responsibilities and opportunities of living, learning and working in an interconnected digital world, and they act in ways that are safe, legal, and ethical.
  B. Engage in positive, safe, legal, and ethical behavior when using technology, including social interactions online or when using networked devices.
  D. Manage their personal data to maintain digital privacy and security and are aware of data collection technology used to track their activity online.

## Opportunities for Computer Science Integration

Curriculum integration strengthens conceptual understanding and skill application. This can be done through multidisciplinary, interdisciplinary, and transdisciplinary approaches to integration. The examples below illustrate multiple ways to integrate computer science.

### English

- After learning about the risks of unauthorized access to information online, students work in small groups to write a skit demonstrating ways to stay safe online. They can perform their skits for classmates or younger students to help promote online safety across grade levels. If group work is not feasible, students can create a comic strip that illustrates how to handle common cybersecurity issues in schools. This activity reinforces key concepts in digital safety while encouraging creativity and peer-to-peer learning.

### Mathematics

- Students research the number of cyber incidents in Virginia and the United States, and, if available, collect data on victim ages. They create graphs to visualize the data and analyze patterns across age groups, locations, and industries. Applying computational thinking, students use abstraction to filter out irrelevant information when identifying trends. Finally, they use the existing data to estimate the probability of future cyber incidents.

## Skills in Practice

Students should engage in the following practices to deepen their conceptual understanding and enhance the application of skills aligned with the Computer Science *Standards of Learning*. These practices are explained in more detail in Appendix A.

### D. Fostering Digital Literacy Practices:

1. Responsible Use Practices
2. Safeguard Well-Being of Self and Others
3. Evaluate Resources and Recognize Contributions

## 5.CYB.2 The student will explain how cybersecurity policies and laws are designed to protect individuals.

a. Explain the importance of policies and laws related to online use of computing devices and the Internet.
b. Research and discuss current cybersecurity policies and laws that protect individuals.
c. Explain legal consequences for inappropriate use of computing technologies.

## Understanding the Standard

**[5.CYB.2a]** Cybersecurity policies are rules and guidelines created by organizations (like schools or companies) to protect computers, networks, and data from cyber threats. They help establish safe practices for using technology and create a secure online environment.

**[5.CYB.2b]** Cybersecurity laws are legal regulations set by governments to protect individuals from cybercrimes, such as hacking, identity theft, and online harassment. Age restrictions are often embedded within laws to protect minors. For example, the Children's Online Privacy Protection Act (COPPA) is a U.S. law that protects the privacy of children under 13 by requiring websites and online services to obtain parental consent before collecting personal information.

Cybersecurity policies and laws are designed to protect personal information and privacy. These laws help prevent and mitigate cyberthreats and are important for national and global security. They hold people accountable for their actions while on the Internet and other interconnected technologies.

**[5.CYB.2c]** In order to keep students safe, schools and divisions have rules on the appropriate use of technology. As students increase their use of the networks and interact with others outside of the school or home environment, digital safety is an increasing concern. Students should be aware of what is allowed and not allowed when using division/school technology. Students should be aware there are consequences for violating school / division policy. The government has laws to protect privacy, fraud, abuse, and identity theft. There are consequences for breaking the law which could include fines or even jail time.

## Concepts and Connections

### Concepts

Cybersecurity policies and laws are essential for protecting individuals' privacy, data, and digital rights in online environments. Understanding current legal frameworks helps students recognize how these protections operate and the consequences of violating them. Inappropriate use of technology can lead to serious legal repercussions, emphasizing the need for responsible and ethical digital behavior.

### Connections

**Within the grade level/course:** At this grade level, students explore government policies and laws designed to protect individuals when using computing devices. They begin to examine the consequences of violating these rules and consider how such actions impact individuals and society (5.CYB.2).

**Vertical Progression:** In Grade 4, students learned the importance of protecting private information online (4.CYB.2 and 4.CYB.3). In Grade 6, students will begin to explore usage agreements and how they provide protection for the individual user (6.CYB.2).

### DIGITAL LEARNING INTEGRATION

- **3-5.DC** Students recognize the rights, responsibilities and opportunities of living, learning and working in an interconnected digital world, and they act in ways that are safe, legal, and ethical.
  B. Engage in positive, safe, legal, and ethical behavior when using technology, including social interactions online or when using networked devices.
  D. Manage their personal data to maintain digital privacy and security and are aware of data collection technology used to track their activity online.
- **3-5.CC** Students communicate clearly and express themselves creatively for a variety of purposes using appropriate technologies (including assistive technologies), styles, formats, and digital media appropriate to their goals.
  A. Choose the appropriate technologies and resources for meeting the desired objectives of their creation or communication.
- **3-5.GC** Students use appropriate technologies, including assistive technologies, to broaden their perspectives and enrich their learning by collaborating with others and working effectively in teams locally and globally.
  A. Use appropriate technologies to connect with learners from a variety of backgrounds and cultures, engaging with them in ways that broaden mutual understanding and learning.

## Opportunities for Computer Science Integration

Curriculum integration strengthens conceptual understanding and skill application. This can be done through multidisciplinary, interdisciplinary, and transdisciplinary approaches to integration. The examples below illustrate multiple ways to integrate computer science.

### English

- After learning about cybersecurity policies and laws, students write a persuasive paragraph arguing whether future policies should be strengthened or relaxed. Prior to writing, students analyze real-world case studies or current events related to cybersecurity to support their claims with evidence.

### History and Social Science

- While researching current laws pertaining to cybersecurity in the United States and Virginia, students could also research what types of technology was available to the majority of people when the laws were passed and other events happening at the time.

## Skills in Practice

Students should engage in the following practices to deepen their conceptual understanding and enhance the application of skills aligned with the Computer Science *Standards of Learning*. These practices are explained in more detail in Appendix A.

### D. Fostering Digital Literacy Practices:

1. Responsible Use Practices
2. Safeguard Well-Being of Self and Others
3. Evaluate Resources and Recognize Contributions

Back to Cybersecurity (CYB)

## Data and Analysis (DA)

### 5.DA.1 The student will collect data or use data sets to solve a problem or investigate a topic.

a. Identify accurate ways data can be collected.
b. Evaluate the reliability of the data source.
c. Organize data based on similarities or patterns.
d. Compare and contrast various data elements.

## Understanding the Standard

Data consists of individual pieces of information about people, things, or events that can be processed, stored, and analyzed by computing devices. The primary purpose of collecting data is to answer questions (e.g., "What type of cell phone do fifth graders prefer?", "What type of weather does a particular region experience?" or "What is the fifth grades' most popular lunch choice?"). The primary purpose of interpreting data is to inform decisions (e.g., which type of clothing to pack for a trip based on a weather graph or which type of lunch to serve based upon class favorites).

**[5.DA.1a]** Accurate data collection is essential for ensuring that the information used to make decisions, train algorithms, or test models is reliable and representative of the desired outcomes. This means the data must closely reflect real-world conditions, be measured correctly, and be free from errors, duplications, or inconsistencies that could distort results. Teaching students how to identify accurate ways to collect data involves understanding the tools, techniques, and methods that ensure the data is valid, reliable, and useful for answering questions or solving problems. Students learn how to approach data collection systematically in the following ways:

- Understanding the purpose for data collection by asking, "What is the problem I am trying to solve?" or "What topic am I investigating?" Knowing the goal helps students determine the type of data they need to collect and the most accurate methods for gathering that data. A sample problem or investigation may include building a machine learning model to determine if an email is spam or not.

- Choosing the right type of data by knowing what type of data to collect in order to solve the problem or investigate a topic. There are two types of data that students can collect:

    - Categorical data (e.g., qualitative) are observations about characteristics that can be sorted into groups or categories. It may include photos, text images, videos, non-numerical values, survey responses, true/false values, or colors. In the above example, categorical data would include labels indicating whether an email is spam or not.

    - Numerical data (e.g., quantitative) are values or observations that can be measured. In the above example, numerical data may include the length of the email, the number of hyperlinks, or the number of keywords used.

- Selecting accurate tools and instruments to collect data by knowing the source and type of data so that data the data is reliable. In the above example, the student could either use real-life examples of spam (with permission from their email provider or teacher), create fictional examples, or use a pre-existing dataset for educational purposes, like a sample email dataset (many are available for educational purposes) to collect their data.

    - Categorical data tools may include labels in a spreadsheet (e.g., "spam," or "not spam").
    - Numerical data tools may include tallies within the spreadsheet to determine how many times an email is identified as "spam" or "not spam."

- Considering the variables and controls in the data ensures that the investigation and data is accurate by keeping measurement intervals, conditions, and environmental factors in the investigations consistent. Consistency in how and when data is collected reduces the chances of errors and makes it easier to compare data across different groups or conditions. Consider the following example in a spam detection program:

    - A student considers the variables and controls by maintaining consistent labeling, feature selection, data formatting, and testing practices (the control).
    - The student uses variables that might include subject lines, body content, senders, links or URLs, attachments, or the time the email was sent. This ensures that the program learns from data that is structured and reliable, ultimately resulting in a more effective spam detection model.

**[5.DA.1b]** Not all data sources are equally reliable (i.e., a well-known solar panel company versus an opinion blog or social media review), and students must be able to think critically about where information comes from, how it is collected, and whether it is supported by evidence or expert opinion so that they can make informed decisions and draw accurate conclusions.

- Students should be able to use search terms effectively as they collect information from multiple digital and print sources. They must organize and synthesize information from the print and digital resources, evaluating their relevance, reliability, and credibility (English 5.R.1.C).
- When students consider a larger variety of sources and gauge the quality of information provided, students are able to determine which information will help solve their problem or investigate their topic. Students can evaluate the reliability of data sources based upon its source, methodology, consistency among other sources, citations and references, age of the data, purpose, peer reviews, and consistency and accuracy. (English 5.R.1BC)

**[5.DA.1c]** Application of computer science practices requires that students organize simple data sets to reveal patterns that suggest relationships (Science 5.1). This involves grouping, sorting, or categorizing data elements based on shared characteristics, values, or behaviors. By recognizing and organizing these similarities and patterns, students better analyze, process, and use data to solve problems or optimize solutions. This is a key part of tasks in data analysis, machine learning, and algorithm design. Consider the following example in a simple data set of a game development program:

| Student Name | Game Type | Hours Spent Developing | Bugs Fixed | Multiplayer? |
|---|---|---|---|---|
| Mia | Adventure | 10 | 15 | Yes |
| Liam | Puzzle | 8 | 10 | No |
| Ava | Racing | 6 | 8 | Yes |
| Noah | Strategy | 12 | 20 | No |
| Emma | Adventure | 9 | 14 | Yes |

- The student groups the data according to game type.
- The student sorts the data by number of fixed bugs.
- The student categorizes the data by multiplayer features.
- The student analyzes the data to determine patterns in the relationship between time spent and bugs fixed, shared behaviors, and multiplayer trends.

**[5.DA.1d]** Comparing and contrasting variable data elements means looking at different types of data (or variables) that are collected in a program and identifying how they are similar and different from each other. Variables in computer science are containers that store data values, which can be anything from numbers and words to more complex data like lists or objects. Consider the following example:

A student is creating a quiz program. Three variables the student could use include:

1. Player Name
    a. Data Type: String (e.g., "Sophia")
    b. Purpose: To store the name of the person taking the quiz.
2. Score

a. Data Type: Integer (e.g., 8)
b. Purpose: To keep track of how many questions the player answered correctly.

3. Is Quiz Finished?
    a. Data Type: Boolean (e.g., True or False)
    b. Purpose: To check if the quiz is completed or still in progress.

*Teacher-created example for educational purposes, referencing ideas from online examples.*

In the context of comparing and contrasting, students learn to examine how different variables behave, how they are used, and how their values change during the execution of a program. This helps fifth graders see how different variables work together in a program while being distinct in their purpose and behavior.

## Concepts and Connections

### CONCEPTS

Accurate data collection requires choosing the right methods and tools to gather information that matches the intended purpose. To maintain data integrity, it is important to evaluate the credibility and reliability of sources. Organizing the data by identifying patterns or similarities helps make the analysis more meaningful. Comparing and contrasting different data elements leads to deeper understanding and more informed conclusions.

### CONNECTIONS

**Within this grade level:** At this grade level, expand their data analysis skills as they identify accurate ways to collect data, organize data based on patterns, and compare and contrast the various data elements to solve a problem or investigate a topic (5.DA.1).
**Vertical progression:** In Grade 4, Students engaged with data analysis to determine what type of data is needed to solve a problem or answer a question. They evaluated the reliability of data sources and begin using numeric values to represent non-numeric ideas for the purpose of collecting, storing, cleaning, and organizing their data to prepare it for visualizations (4.DA.1). In Grade 6, students will focus on using computational tools to collect, organize, clean, and analyze their data.

### ACROSS CONTENT AREAS

**English**

- **5.C.4** The student will develop effective oral communication and collaboration skills to build a community of learners that process, understand, and interpret content together.
  B. Identify the purpose, intended audience, and credibility of information (e.g., auditory, visual, and written media messages) being presented.

C. Compare and contrast techniques used in a variety of media messages (e.g., animation, famous images and words, music and sound, photo-editing).

### History and Social Science

- **USI.9** The student will apply history and social science skills to understand the cause, major events, and effects of the Civil War.

### Mathematics

- **5.PS.1** The student will apply the data cycle (formulate questions; collect or acquire data; organize and represent data; and analyze data and communicate results) with a focus on line plots (dot plots) and stem-and-leaf plots.

### Science

- **5.1** The student will demonstrate an understanding of scientific and engineering practices by a) interpreting, analyzing, and evaluating data; b) planning and carrying out investigations; *5.1 standard is integrated within science content and not taught in isolation. Potential science concepts to apply 5.1 include: 5.2 (energy), 5.3 (force, mass, energy transfer),5.4 (energy), 5.5 (sound), 5.6 (light), 5.7 (matter).*

## DIGITAL LEARNING INTEGRATION

- **3-5.KC.** Evaluate the accuracy, perspective, credibility, and relevance of information, media, data, and other digital sources
  A. Plan and employ effective research strategies to locate information and other digital sources for their intellectual or creative pursuits.
  C. Curate information from digital sources using a variety of tools and methods to create collections of resources that demonstrate meaningful connections or conclusions.
- **3-5.CT.** Collect data or identify relevant data sets, use appropriate technologies to analyze them, and represent data in various ways to facilitate problem solving and decision-making. Computational Thinker Students develop and employ strategies for understanding and solving problems in ways that leverage the power of technological methods, including those that leverage assistive technologies, to develop and test solutions.
  Formulate problem definitions suited for technology assisted methods such as data analysis, modeling and algorithmic thinking in exploring and finding solutions.
  D. Understand how automation works and use algorithmic thinking to develop a sequence of steps to create and test automated solutions
- **3-5.GC** Students use appropriate technologies, including assistive technologies, to broaden their perspectives and enrich their learning by collaborating with others and working effectively in teams locally and globally.
  D. Explore local and global issues use collaborative technologies to work with others to investigate solutions.

## Opportunities for Computer Science Integration

Curriculum integration strengthens conceptual understanding and skill application. This can be done through multidisciplinary, interdisciplinary, and transdisciplinary approaches to integration. The examples below illustrate multiple ways to integrate computer science.

### English

- Students evaluate the reliability of data sources by comparing historical fiction and nonfiction primary source accounts of the same event, examining each for consistency and accuracy. As they read, students gather evidence, identify discrepancies, and assess the credibility of each source, refining their understanding as new information emerges. This process emphasizes the importance of identifying appropriate data, verifying its accuracy and reliability, and using that data to construct well-supported interpretations of historical events.

### Science

- Students work collaboratively to design and carry out an investigation that demonstrates how vibrating materials produce sound and transfer energy. As part of the investigation, they decide what data to collect, choose appropriate tools and methods for data collection, and record their observations. Students then organize the data to identify patterns and use their findings to explain the relationship between vibration, sound, and energy transfer.

## Skills in Practice

Students should engage in the following practices to deepen their conceptual understanding and enhance the application of skills aligned with the Computer Science *Standards of Learning*. These practices are explained in more detail in Appendix A.

### B. Fostering Computational Thinking Practices:

1. Decompose Real-World Problems
2. Explore Common Features and Identify Patterns
3. Use Abstraction to Simplify, Represent, and Problem Solve
4. Apply Algorithmic Thinking to Problem Solve and Create
5. Apply Computational Thinking Practices to Select, Organize, and Interpret Data

### C. Fostering Iterative Design Practices:

1. Identify, Define, and Evaluate Real-world Problems
2. Plan and Design Artifacts

3. Create, Communicate and Document Solutions
4. Test and Optimize Artifacts

## 5.DA.2 The student will create multiple data representations to make predictions and conclusions.

a. Formulate questions that require the collection or acquisition of data.
b. Collect data to use in creating charts, graphs, and models.
c. Analyze data as evidence to draw conclusions and make predictions.
d. Propose solutions to problems or questions based on data analysis.

## Understanding the Standard

Creating and evaluating multiple data representations to make predictions and conclusions helps to develop data analysis skills. The data cycle is the process of formulating questions to be explored with data, collecting or acquiring data, organizing and representing data, and analyzing and communicating results. It is used in Computer Science, Mathematics, and Science, and it closely aligns with the Scientific and Engineering Practices in Science. Data analysis engages students in applying the full data cycle, beginning with the formulation of questions that require data collection. Students may use digital tools and computing devices to gather, organize, and visualize data generated through experimentation. At this stage, they are expected to implement each phase of the data cycle: posing questions, collecting data, analyzing results, and reflecting on findings to develop a deeper understanding of data-driven inquiry. (Mathematics 5.PS.1).

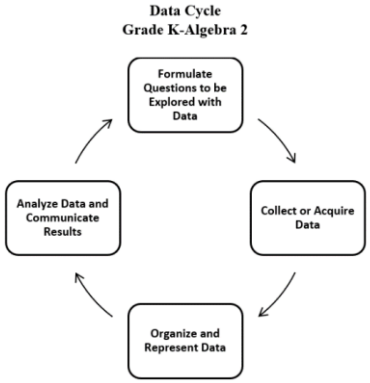



Image from: Virginia Department of Education Mathematics Standards of Learning Instructional Guide © 2024: Grade 5 [August 2024]

**[5.DA.2a]** The data cycle begins with teaching students how to identify what information is needed so that they can formulate real world questions that become manageable data collection tasks. Statistical investigations should be active, with students formulating questions about something in their environment and determining ways to answer the questions (Mathematics 5.PS.1ab).

The use of open-ended questions allows for exploration and requires data collection to answer them. Questions students can ask to collect data include, but are not limited to, the following:

- How do different search algorithms perform on the same dataset?
- How does the frequency of user input affect the responsiveness of a program?
- How can data visualization techniques help us better understand trends in a dataset?
- How does the number of iterations in an algorithm affect its overall execution time?

**[5.DA.2b]** The next step of the data cycle involves collecting or gathering the appropriate type of data needed and organizing data. The teacher can provide data sets to students in addition to students engaging in their own data collection or acquisition (e.g., polls, observations, measurements, or experiments). Investigations that support collecting data can be brief class surveys or more extended projects occurring over multiple days. Data may be acquired from resources that are already created. Consider the following examples:

- Amount of screen time spent each evening over the course of a month
- Most popular hours for screen time usage in the evenings
- Weather data over several days or months (e.g., temperature, precipitation amounts)
- Population data over a period of time, etc. (Mathematics 5.PS.1b)

Students implement the next step of the data cycle by determining how they want to organize their data and prepare it for visual representation. The way data is displayed is often dependent upon what question is being investigated and what someone is trying to communicate (Mathematics 5.PS.1).

- Different situations call for different types of graphs (e.g., visual representations). At this point, students benefit from exploring previously taught graphical representations when justifying how to best represent data clearly and accurately. Technology tools (e.g., graphing software, spreadsheets) can be used to collect, organize, and visualize data. These tools support progression to analysis of data in a more efficient manner.
- Students should have experience displaying data in a variety of graphical representations, and determining which representation is most appropriate (e.g., a representation that is more helpful in analyzing and interpreting the data to answer questions and make predictions). Interpretations of the data that include comparisons, inferences, and recognizing trends that may exist are made by examining characteristics of a data set displayed in a variety of graphical representations (Mathematics 5.PS.1).
- Comparing different types of representations (e.g., charts, graphs, line plots, stem-and-leaf plots) provides students with opportunities to learn how different graphs can show different aspects of the same data. Following the construction of representations, discussions around what information each representation provides or does not provide should occur (Mathematics 5.PS.1).
- Graphing is crucial in Computer Science, Mathematics, and Science as it helps students visualize patterns and relationships and supports students as they work to understand data. It also helps with the identification of trends so that predictions can be made. Students use data to model real-world scenarios and are fundamental to the development and understanding of algorithms and data structures. Graphing enhances the comprehension of data and promotes critical thinking and problem solving across the disciplines (Mathematics 5.PS.1cde).

Students continue to construct charts and graphs to represent data, but their learning progresses as they begin using data to develop models.

Sample Chart or Table

| Email Content | Label |
|---|---|
| “Congratulations! You won a free vacation!” | Spam |
| “Your teacher sent the class notes for today.” | Not Spam |
| “Hurry! Last chance to buy a new laptop online!” | Spam |
| “Science test is rescheduled for next week” | Not Spam |

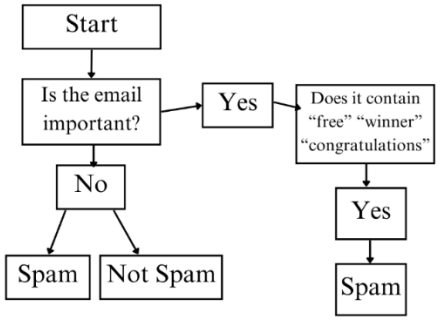
Prior to Grade 5, students engaged with object graphs, picture graphs, pictographs, tables, bar graphs, and line graphs. In Grade 5, students focus on data represented in line plots (also referred to as dot plots) and stem-and-leaf plots (Mathematics 5.PS.1c).

Sample Dot Plot


Source: https://www.mathsisfun.com/data/dot-plots.html

Computer scientists, scientists, mathematicians, and programmers construct and use models to better conceptualize and understand phenomena under investigation or to develop a possible solution to a proposed problem. Models include diagrams, physical replicas, mathematical representations, analogies, and computer simulations. Models are used to represent a system (or parts of a system) under study, to aid in the development of questions and explanations, to generate data that can be used to make predictions, and to communicate ideas to others.

Sample Flow Chart Model




REDO FLOWCHART

Image created in Canva

**[5.DA.2c]** Data analysis includes opportunities to describe the data, recognize patterns or trends, make predictions, and draw conclusions. Once students create visual representations of data, the emphasis is on the data analysis and communication of the analysis. Students are interested in and notice individual data points and are able to describe parts of the data such as where their own data falls on the graph, which value occurs most frequently, and which values are the largest and smallest. It is important to build student understanding as they start to view and interpret the data as a complete set.(Mathematics 5.PS.1e).

When comparing different data representations, it is important to ask questions such as: (Mathematics 5.PS.1e)

- What inferences can you make?
- What do you notice about the data?
- In which representations can you identify individual data points?
- In which representations can you quickly identify the mode or median? The range?
- Which parts of the data have special characteristics?
- Explain the meaning of the greatest, the least, or the same in the data.

**[5.DA.2d]** The last step in the data cycle is to communicate the results of the analysis. Students build upon their prior knowledge by proposing solutions to problems or questions based on data analysis. Statements representing an analysis and interpretation of the characteristics of the data in the visual representations should be included in students' proposals and questions (e.g., patterns or trends of increase and/or decrease, and least and greatest data value). Some sentence frames to guide students' solutions may include: (Mathematics 5.PS.1e)

- “To improve the current situation, I recommend ____ based on the trends observed in the data."
- "A potential solution could be to ____ because the data indicates that this action would result in ____."
- "Based on the patterns in the data, I suggest implementing ____ to address the issue of ____."
- "Given the data, it would be helpful to try ____ to improve ____."

## Concepts and Connections

### Concepts

Data-driven inquiry begins by formulating purposeful questions that guide the collection of relevant information. Visual tools such as charts, graphs, and models help organize and reveal patterns within the data. Analyzing these patterns supports evidence-based conclusions and informs practical solutions to problems.

### Connections

**Within this grade level:** At this grade level, students expand their data analysis skills as they analyze and use their data to create charts, graphs, and models so that they can make predictions, draw conclusions, and propose solutions to problems or questions based upon the data analysis (5.DA.2).
**Vertical progression:** In Grade 4, students formulated questions that require data collection, collected data, recognized and analyzed patterns, and analyzed visual representations so that they could make predictions and draw conclusions based on the data (4.DA.2). In Grade 6, students will begin using computational tools to visualize data, distinguish between which visual representations should be used, and synthesize and evaluate data from visual representations (6.DA.2).

### Across Content Areas

**English**

- **5.W.1B** Write expository texts to examine a topic and convey ideas that develops the focus with relevant facts, concrete details, or examples from multiple sources and are grouped logically.
- **5.W.1C** Write persuasive pieces on topics or texts, including media messages, supporting a clear perspective with adequate facts, reasons, and logically grouped information.
- **5.W.1D** Write in response to texts read (including summaries, reflections, and descriptions) in which students demonstrate their thinking with details, examples, and other evidence from the text that are logically grouped.
- **5.C.3A** Select, organize, and create engaging presentations that include multimedia components and visual displays.
- **5.C.3B** Strategically use two or more interdependent modes of communication to convey the intended message and enhance the development of main ideas or themes.

- **5.R.1A** Formulate questions that help narrow the topic and revise questions as needed based on research.
- **5.R.1E**. Organize and share information orally, in writing, or through visual display.

### Mathematics

- **5.PS.1** The student will apply the data cycle (formulate questions; collect or acquire data; organize and represent data; and analyze data and communicate results) with a focus on line plots (dot plots) and stem-and-leaf plots.

### Science

- **5.1** The student will demonstrate an understanding of scientific and engineering practices by c) interpreting, analyzing, and evaluating data; e) developing and using models; f) obtaining, evaluating, and communicating information; *5.1 standard is integrated within science content and not taught in isolation. Potential science concepts to apply 5.1 include: 5.2 (energy), 5.3 (force, mass, energy transfer),5.4 (energy), 5.5 (sound), 5.6 (light), 5.7 (matter).*

### DIGITAL LEARNING INTEGRATION

- **3-5.CT** Students develop and employ strategies for understanding and solving problems in ways that leverage the power of technological methods, including those that leverage assistive technologies, to develop and test solutions.
  B. Collect data or identify relevant data sets, use appropriate technologies to analyze them, and represent data in various ways to facilitate problem solving and decision-making.
- **3-5.KC** Students critically curate a variety of digital resources using appropriate technologies, including assistive technologies, to construct knowledge, produce creative digital works, and make meaningful learning experiences for themselves and others.
  C. Curate information from digital sources using a variety of tools and methods to create collections of resources that demonstrate meaningful connections or conclusions.
- **3-5.CC** Students communicate clearly and express themselves creatively for a variety of purposes using appropriate technologies (including assistive technologies), styles, formats, and digital media appropriate to their goals.
  C. Communicate complex ideas clearly and effectively by creating or using a variety of digital objects such as visualizations, models, or simulations.

## Opportunities for Computer Science Integration

Curriculum integration strengthens conceptual understanding and skill application. This can be done through multidisciplinary, interdisciplinary, and transdisciplinary approaches to integration. The examples below illustrate multiple ways to integrate computer science.

### History and Social Sciences

- Students explore how early cultures developed in North America by focusing on five Indigenous tribes. Using a Jigsaw strategy, they begin by formulating questions about each tribe's location, environment, and resource use. Students then collect and analyze data from texts, maps, and other sources to find evidence that addresses their questions. After sharing findings in expert groups, they collaborate to create a slideshow that highlights each tribe's unique characteristics and organizes the resources they used. This process supports critical thinking and helps students draw conclusions about how geography and environment shaped cultural development.

### Mathematics

- Students create a decision tree diagram, list, or chart to represent all possible ice cream combinations, defining the complete sample space. They use this representation to determine the probability of selecting a cone with chocolate ice cream and hot fudge. Through this process, students apply logical reasoning and organize data in a structured way to support their conclusions. The activity helps students develop a foundational understanding of probability and strengthens their ability to model real-world scenarios mathematically. By analyzing patterns and outcomes, students make sense of how different combinations impact the likelihood of a specific event. By way of example:

    A. The fifth grade has won an ice cream party. They have the option of selecting one item from each category. They may have their ice cream in a cup or a cone. They may choose a scoop of chocolate, strawberry or vanilla, and they may choose one topping of either sprinkles, whipped cream, or hot fudge.

### Science

- Students create an investigation to determine which types of surfaces create the most friction. They build and design ramps of the same height and place different surfaces at the bottom of the ramp (tile, carpet, and sandpaper). The students conduct three trials for each surface in which they have a matchbox car go down the ramp, measure the distance it travels, and record their data in a chart. After conducting their trials, they analyze their data to draw conclusions about which surfaces create the most friction.

- Students explore tectonic plates by collecting data on where various convergent, divergent, and transform boundaries exist on the earth's surface. They use an online simulator to create a model of the boundary types and their effects upon the earth. After viewing and analyzing the boundaries' effects, students make predictions about future landforms based on where tectonic plates meet and the types of boundaries that are located on those plates.

## Skills in Practice

Students should engage in the following practices to deepen their conceptual understanding and enhance the application of skills aligned with the Computer Science *Standards of Learning*. These practices are explained in more detail in Appendix A.

## B. Fostering Computational Thinking Practices:

1. Decompose Real-World Problems
2. Explore Common Features and Identify Patterns
3. Use Abstraction to Simplify, Represent, and Problem Solve
4. Apply Algorithmic Thinking to Problem Solve and Create
5. Apply Computational Thinking Practices to Select, Organize, and Interpret Data

## C. Fostering Iterative Design Practices:

1. Identify, Define, and Evaluate Real-world Problems
2. Plan and Design Artifacts
3. Create, Communicate and Document Solutions
4. Test and Optimize Artifacts

## 5.DA.3 The student will explain the significance of training data in machine learning.

a. Compare how training data is utilized in supervised, unsupervised and reinforcement learning.
b. Explain how training data is used to make classification predictions.
c. Discuss the need and significance of diverse, inclusive, and large datasets.

## Understanding the Standard

Machine learning is an approach that occurs when computers learn from examples instead of following exact instructions. Examples of machine learning include supervised learning, unsupervised learning, and reinforcement learning. In machine learning, training data is very important because it helps a computer program learn how to make decisions or predictions. The computer looks at data, figures out the pattern, and learns how to solve similar problems in the future. Training data is essential for enabling a computer to learn and make accurate predictions. High-quality training data is characterized by a large set of clear, relevant examples that enable effective learning by the program. In contrast, if the data is incomplete, inconsistent, or misleading, the program may produce inaccurate or unreliable results.

**[5.DA.3a]** Students begin comparing how data training is utilized in supervised, unsupervised, and reinforcement learning. Each type of learning uses training data in a different way to help a computer get smarter and make better decisions.

- Supervised learning is when a computer learns to make decisions or predictions by looking at examples that have the correct answers already provided. Consider the following example:
    - A computer is given lots of labeled examples (e.g., emails labeled as “spam” or “not spam”).
    - The computer processes the examples and identifies patterns (e.g., “This is spam.”).
    - After seeing enough examples, the computer makes its own guesses about new things it hasn't seen before.

- Unsupervised learning is when a computer tries to learn from data without being told what the answers are. It is like the computer is solving a puzzle all by itself, figuring out the patterns in the data without anyone giving it the correct labels or answers. Consider the following example:
    - A computer is fed various data on emails.
    - The computer analyzes emails and creates groups based on similar patterns that it identifies. It may see that a lot of emails have key words such as “winner,” “free,” “notes,” or “test.”
    - It determines that all the emails with “winner” and “free” belong together, while emails with keywords like “notes” or “test” belong together.

- Reinforcement learning is like teaching a computer through trial and error, where it learns by getting rewards or punishments for the actions it takes. Consider the following example:
    - A program has the goal of correctly labeling emails as “spam” or “not spam.”
    - It makes a guess based on clues from the sender, subject line, or key words. The computer learns that emails with certain words like "free" or "winner" are often spam, so it gets rewarded with a “correct” message for labeling these emails correctly.
    - If the computer labels a real email (one from a friend or work) as spam, it gets punished in the form of an “incorrect” message and learns not to make that mistake again.

**[5.DA.3b]** After a computer has learned from training data, it is provided with new data it has never seen before, and it tries to guess what it is. The computer uses what it learned from the examples to make its prediction by classifying the data into groups. Consider this example:

- A student provides training data to a computer with lots of labels for “spam” and “not spam.”
- The computer looks at the data and learns the pattern that emails with keywords like “free” and “winner” are spam, while emails with keywords such as “meeting,” “test,” or “notes” are “not spam.”
- The computer notices the patterns and classifies them by which words are most likely to show up in spam emails.
- When a new email is provided to the computer with a subject line such as, “Congratulations! You are the winner!”, the computer predicts that this email is spam.

**[5.DA.3c]** Having large and diverse datasets is important to ensure that a computer makes more fair and accurate predictions.

- Diverse means that a dataset has a variety of examples to show different kinds of things. Providing multiple examples of spam emails helps the computer to learn to recognize all kinds of spam, not just one type.
- Inclusive means that the dataset includes different groups of things that are important. When training a computer to distinguish between “spam” and “not spam”, the dataset needs to include a wide variety of emails from different sources, topics, and types of content, and represents different people, places, and cultures. The goal is to make sure that the dataset is not biased toward one group or type of email, so the computer can learn to classify all types of emails fairly and accurately.
- Large datasets means that the dataset has a lot of examples so that the computer can learn better. The more examples a computer can see, the better able it is to make good predictions and decisions. A large dataset has more examples to learn from, so the computer is less likely to make mistakes.

Another example is self-driving cars. A self-driving car uses data (like pictures from cameras, maps, and sensors) to learn how to drive. If the car is trained with a diverse and large dataset, it will learn to drive in different weather conditions (like rain, snow, or bright sun) and understand various road signs (like stop signs, speed limits, or construction zones). But if the dataset is too small or doesn't have examples of these situations, the car might get confused or make bad decisions.

## Concepts and Connections

### CONCEPTS

Training data is essential to how machine learning models learn patterns in supervised, unsupervised, and reinforcement learning. It enables algorithms to make accurate classification predictions based on learned relationships. The quality and representativeness of the data are critical for minimizing bias and ensuring reliable model performance.

### CONNECTIONS

**Within the grade level/course:** At this grade level, explore machine learning models to compare how training data is used, explain how it is used to make classification predictions, and discuss why large and diverse datasets are necessary in machine learning (5.DA.3).
**Vertical Progression:** In Grade 4, students expanded their data analysis skills by using computational models to represent attributes and behaviors associated with a concept. They explored various models of physical objects or processes, created their own models, and used the models to explain how a computer model illustrates a concept (4.DA.3). In Grade 6, students will investigate ways that humans provide training data, explore the role of human intervention in curating training data, and explain the accuracy of artificial intelligence systems based upon the training data used (6.DA.4).

### ACROSS CONTENT AREAS

**English**

- **5.RI.2C** The student will use textual evidence to demonstrate and build knowledge from a variety of grade level complex informational texts heard or read by determining the author's purpose(s) and describing how the author's perspective (i.e., beliefs, assumptions, biases) influences the meaning of the text. *(Note: only aligned if students are reading grade level informational text).*

**Mathematics**

- **5.PS.2** The student will solve contextual problems using measures of center and the range.
- **5.MG.3** The student will classify and measure angles and triangles, and solve problems, including those in context.

### DIGITAL LEARNING INTEGRATION

- **3-5.KC** Students critically curate a variety of digital resources using appropriate technologies, including assistive technologies, to construct knowledge, produce creative digital works, and make meaningful learning experiences for themselves and others.
  A. Plan and employ effective research strategies to locate information and other digital sources for their intellectual or creative pursuits.

## Opportunities for Computer Science Integration

Curriculum integration strengthens conceptual understanding and skill application. This can be done through multidisciplinary, interdisciplinary, and transdisciplinary approaches to integration. The examples below illustrate multiple ways to integrate computer science.

### Mathematics

- Students discover the importance of large and inclusive datasets as they manipulate data sets of whole numbers and decimals in lists, stem and leaf plots, or line plots (i.e., scores on tests, batting averages, temperatures, distances, etc.) to solve for the mean, median, mode, and range. They explore how the deletion or inclusion of additional data (i.e., receiving a grade of a “0”) impacts the measures of center in the data set.

- Students explore the concept of training data through a triangle sort activity. Students compare and contrast various triangles and classify them by angles (acute, right, obtuse, or straight) or by sides (equilateral, isosceles, or scalene). This process models a form of supervised learning, where labeled examples are used to "train" a system to recognize patterns and apply consistent classification rules. By identifying attributes and assigning categories, students build a foundational understanding of how machines use training data to learn and make decisions.

## Skills in Practice

Students should engage in the following practices to deepen their conceptual understanding and enhance the application of skills aligned with the Computer Science *Standards of Learning*. These practices are explained in more detail in Appendix A.

### B. Fostering Computational Thinking Practices:

1. Decompose Real-World Problems
2. Explore Common Features and Identify Patterns
3. Use Abstraction to Simplify, Represent, and Problem Solve
4. Apply Algorithmic Thinking to Problem Solve and Create
5. Apply Computational Thinking Practices to Select, Organize, and Interpret Data

## C. Fostering Iterative Design Practices:

1. Identify, Define, and Evaluate Real-world Problems
2. Plan and Design Artifacts
3. Create, Communicate and Document Solutions
4. Test and Optimize Artifacts

Back to Data and Analysis (DA)

## Impacts of Computing (IC)

### 5.IC.1 The student will analyze the impact of inappropriate use of computing technologies.

a. Predict consequences for inappropriate uses of computing technologies.
b. Describe how technology-related problems can be avoided or prevented.
c. Develop solutions for a scenario involving inappropriate use of computing technologies.

## Understanding the Standard

Responsible use of computing technologies begins with awareness of how digital actions affect oneself and others. Acceptable use policies emphasize the need for students to analyze these impacts, which can include actions such as cyberbullying, sharing harmful content, privacy violations, and inappropriate communication. Inappropriate use of technology can lead to significant emotional, social, legal, and academic repercussions.

**[5.IC.1a]** Teaching students to navigate the digital world responsible involves regular instruction in digital citizenship, including ethical technology use, online respect, and personal data protection. Understanding and predicting the consequences of inappropriate uses of computing technologies is vital for creating a proactive educational environment. For instance, unauthorized access to sensitive information can lead to severe privacy breaches and potential identity theft, impacting both students and their families. Sharing false information online may foster misinformation, leading to confusion and mistrust among peers. Predicting the consequences of digital actions promotes greater awareness of personal responsibility and the broader impact on others.

**[5.IC.1b]** Technology-related problems can often be avoided by following safe and responsible practices. For example, using strong passwords, updating software regularly, avoiding suspicious links, and backing up important files can help prevent issues like viruses, data loss, or unauthorized access. Understanding how to care for devices and use them properly also reduces the chance of hardware damage or technical errors. Teaching these habits helps students recognize potential risks and take steps to prevent problems before they happen.

## Concepts and Connections

### CONCEPTS

Inappropriate use of computing technologies can result in serious consequences, including disciplinary action, security risks, and harm to others. Establishing clear guidelines and promoting responsible digital habits help prevent these issues. When addressing misuse, students should apply critical thinking to evaluate situations and develop thoughtful, responsible solutions that support safe and ethical technology use.

### CONNECTIONS

**Within this grade level:** At this grade level, students begin to predict consequences associated with inappropriate use of computing technologies. They will also develop solutions to scenarios involving the inappropriate use of computing technologies (5.IC.1).
**Vertical progression:** In Grade 4,students focused on the economic impacts on computing industries in Virginia (4.IC.1). In Grade 6, students will examine the local impact of computing technologies. They research and analyze the implications of computing technologies (6.IC.1).

## Opportunities for Computer Science Integration

Curriculum integration strengthens conceptual understanding and skill application. This can be done through multidisciplinary, interdisciplinary, and transdisciplinary approaches to integration. The examples below illustrate multiple ways to integrate computer science.

### English

- Students participate in structured conversations surrounding the impacts of using computing devices. Students create posters and write samples to describe how technology-related problems can be prevented. They can also create scenarios involving inappropriate use of computing technologies.

### History and Social Science

- Students examine the societal and economic impacts of early technologies such as the cotton gin, the reaper, the steamboat, the steam locomotive, and the telegraph. They analyze how these innovations, while advancing productivity, may also led to unintended consequences. Students then draw connections to modern computing technologies, exploring how inappropriate or unethical use can produce similar outcomes.

## Skills in Practice

Students should engage in the following practices to deepen their conceptual understanding and enhance the application of skills aligned with the Computer Science *Standards of Learning*. These practices are explained in more detail in Appendix A.

### D. FOSTERING DIGITAL LITERACY PRACTICES:

1. Responsible Use Practices
2. Safeguard Well-Being of Self and Others
3. Evaluate Resources and Recognize Contributions

## 5.IC.2 The student will explain the potential impact of excessive screen time on academic performance.

a. Analyze data to determine the impact of screen time on academic performance.
b. Describe how academic behaviors that lead to academic success are impacted by daily screen time.
c. Differentiate usage of screen time that benefit and hinder academic performance.

## Understanding the Standard

**[5.IC.2a]** Research indicates that students who spend more time on screens than recommended often face several academic challenges. For instance, high screen time is linked to shorter attention spans, which can lead to difficulty concentrating during lessons and while doing homework. Additionally, students who spend excessive time on screens are often exposed to distractions, limiting their ability to focus on learning tasks. Students can collect data about their screen time use and set goals for balanced screen time that will have a positive impact on academic performance.

**[5.IC.2bc]**

| Screen Time Usage | Benefits to Academic Performance | Hindrances to Academic Performance |
|---|---|---|
| Educational Apps | Enhance learning through interactive content | Can lead to distraction if not properly monitored |
| Online Research | Provides access to a wealth of information | Over-reliance can lead to superficial understanding |
| Collaborative Tools | Encourages teamwork and communication among peers | Can become overwhelming with too many platforms and notifications |
| Gaming (Educational) | Application of problem-solving skills and critical thinking | Can consume excessive time leading to procrastination |
| Social Media | Can foster connections with educational communities | Potential for misinformation and distraction |
| Streaming Educational Videos | Visual aids enhance understanding and retention | Time management issues leading to reduced study time |
| Coding and Programming | Develops logical thinking and creativity | Frustration can lead to disengagement if too challenging |

## Concepts and Connections

### CONCEPTS

Evaluating screen time patterns supports the development of balanced technology habits that promote academic success. Differentiating between productive and non-productive screen use enables more intentional and goal-oriented technology use.

### CONNECTIONS

**Within this grade level:** At this grade level, students will examine the impact of screen time on academic performance. They will compare and contrast screen time that benefits and hinders academic performance (5.IC.2).
**Vertical progression:** In Grade 4, students explained the impact of excessive screen time on relationships and the impact at school (4.IC.2). In Grade 6, students will take an in-depth look at the impacts of screen time on their physical and mental health. Students will also examine social media use and cyberbullying. They will be able to define social media and cyberbullying along with their impact on their mental health (6.IC.2).

## Opportunities for Computer Science Integration

Curriculum integration strengthens conceptual understanding and skill application. This can be done through multidisciplinary, interdisciplinary, and transdisciplinary approaches to integration. The examples below illustrate multiple ways to integrate computer science.

### English

- Students write an opinion piece explaining how much screen time they believe is appropriate and how it affects academic performance, using reasons and evidence to support their viewpoint.

### Mathematics

- Students collect authentic data on their own screen time usage and other activities. Using the data cycle, they organize, represent, and analyze both individual and class data to identify patterns, make comparisons, and draw conclusions.

## Skills in Practice

Students should engage in the following practices to deepen their conceptual understanding and enhance the application of skills aligned with the Computer Science *Standards of Learning*. These practices are explained in more detail in Appendix A.

### D. FOSTERING DIGITAL LITERACY PRACTICES:

1. Responsible Use Practices
2. Safeguard Well-Being of Self and Others
3. Evaluate Resources and Recognize Contributions

## 5.IC.3 The student will identify the impact of computing technologies on the workforce, culture, and global society.

a. Research and analyze computing technology careers in global society.
b. Explore the impact of emerging technologies on workforce, culture, and global society.

## Understanding the Standard

Computing technologies have transformed nearly every aspect of modern life, reshaping how people work, communicate, and engage with the world. In the workforce automation and computing technologies have created new job opportunities while also changing the skills required for many careers. Culturally, people tend to adopt technologies that support the activities and customs important to their communities, such as communication, collaboration, or creative expression. These cultural needs often drive the development of new technologies designed to make everyday tasks more efficient or to support global connectivity. On a global scale, increased access to the Internet has accelerated innovation and economic development, while also influencing social norms and raising questions about equity, privacy, and responsible use.

**[5.IC.3a]** Computing technology careers play a vital role in today's global society by supporting innovation, communication, and problem-solving across industries. Careers such as software development, cybersecurity, data science, and artificial intelligence are in high demand worldwide, as businesses and governments rely on technology to operate efficiently and securely. These careers often require collaboration across countries and cultures, highlighting the importance of digital literacy and global awareness. As technology continues to evolve, computing careers will remain essential to addressing global challenges and advancing economic growth, education, and healthcare.

**[5.IC.3b]** Emerging technologies such as artificial intelligence, robotics, virtual reality, and advanced data analytics are rapidly changing the way people live and work. In the workforce, these technologies are creating new types of jobs while also automating tasks that once required human labor, leading to shifts in required skills and workplace structures. Culturally, emerging technologies influence how individuals interact, create, and share information, often reshaping traditions, communication styles, and social norms. On a global scale, these advancements are expanding access to information, education, and healthcare, while also raising critical questions about digital access, ethics, and the responsible use of technology across communities.

The use of technology, including computers, has allowed global communication and has revolutionized the everyday access of information, whether for business, scientific or personal use. Although there are many positive impacts in using technology, there are also times when computer use has impacted us in undesirable ways. As computer technology continues to advance and new generations of machines grow faster and have greater capabilities, the machines become more deeply fixed in daily life, magnifying both the benefits and the downside risks. Positive impacts include easy access to information, automated machinery, and fast and accurate data processing. Negative impacts include an increase in sedentary lifestyles, family and leisure interruption, and loss of privacy.

## Concepts and Connections

### CONCEPTS

Exploring computing technology careers reveals the expanding role of digital skills across diverse sectors in the global workforce. Analyzing emerging technologies highlights their transformative impact on economic structures, cultural norms, and societal interactions worldwide.

### CONNECTIONS

**Within the grade level/course:** At this grade level, students examine the global impact of technology related careers. They will also examine the diversity and inclusivity of computing careers (5.IC.3).

**Vertical Progression:** In Grade 4, students examined the various key computing technologies and their impact on the workforce (4.IC.3). In Grade 6, students will narrow their focus to exploring a computing technology career of their choice. They will also examine how computational thinking practices are used in the chosen career (6.IC.3).

### ACROSS CONTENT AREAS

**English**

- **5. DSR** The student will build knowledge and comprehension skills from reading a range of challenging, content-rich texts. This includes fluently reading and gathering evidence from grade-level complex texts, reading widely on topics to gain purposeful knowledge and vocabulary, and using reading strategies when comprehension breaks down.
- **5.RI** The student will use textual evidence to demonstrate and build knowledge from a variety of grade level complex informational texts heard or read. *(Note: only aligned if students read grade level informational texts)*
- **5.W** The student will compose various works for diverse audiences and purposes, linked to grade five content and texts.
- **5.C** The student will develop effective oral communication and collaboration skills to build a community of learners that process, understand, and interpret content together.

**History and Social Science**

- **USI.8** The student will apply history and social science skills to explain westward expansion and reform in America from 1801 to 1861 by a. describing how territorial expansion affected the political map of the United States including, but not limited to the Louisiana Purchase, the Lewis and Clark Expedition and the role of Sacagawea, the acquisitions of Florida, Texas, Oregon, and California, and the results of the Mexican-American War; b. describing the causes, course of events, and effects of the War of 1812, the role of Andrew Jackson, and the development of the Monroe Doctrine; c. identifying geographic, economic, and religious motivations that influenced the movement of settlers; d. analyzing the impact of westward expansion on Indigenous people including, but not limited to the Indian Removal Act (1830), the Trail of Tears, and the Seminole Wars; e. explaining technological advancements and innovations and their effects on life in America including, but not limited to the cotton gin, the reaper, the steam engine, and the steam locomotive; f.

describing major developments in the abolitionist and women's suffrage movements; and g. explaining how the expansion of U.S. territory led to increased momentum for the abolitionist and women's suffrage movements.

### Mathematics

- **5.PS.1** The student will apply the data cycle (formulate questions; collect or acquire data; organize and represent data; and analyze data and communicate results) with a focus online plot (dot plots) and stem-and-leaf plots.

## Opportunities for Integration

Curriculum integration strengthens conceptual understanding and skill application. This can be done through multidisciplinary, interdisciplinary, and transdisciplinary approaches to integration. The examples below illustrate multiple ways to integrate computer science.

### English

- Students research the impact of emerging technologies on the workforce, culture, and global society by analyzing a variety of informational texts, articles, and case studies. As they investigate, students cite textual evidence to support claims about both the benefits and challenges of technological advancements. They use this evidence to draw conclusions, make comparisons across sources, and communicate their findings through discussion or written responses.

### Mathematics

- Students research different computing-related careers such as game designer, robotics engineer, or cybersecurity analyst, and learn how these jobs help people and communities around the world. They read short articles or watch videos to find out what each job does, what tools are used, and how new technologies are changing the way people work. Using numerical data, students apply mathematical skills to create bar graphs, pictographs, or tables to compare and analyze the information. Through this process, they engage in data interpretation, identify meaningful patterns and trends, and develop evidence-based conclusions supported by quantitative reasoning.

## Skills in Practice

Students should engage in the following practices to deepen their conceptual understanding and enhance the application of skills aligned with the Computer Science *Standards of Learning*. These practices are explained in more detail in Appendix A.

### B. Fostering Digital Literacy Practices:

1. Responsible Use Practices
2. Safeguard Well-Being of Self and Others

3. Evaluate Resources and Recognize Contributions

## 5.IC.4 The student will observe and examine intellectual property rights when considering the use of open-source licenses and copyrights.

a. Distinguish between open-source licenses and copyrights.
b. Research risks associated with inappropriate use of various digital information sources.
c. Describe and use strategies to protect online digital content and resources.

## Understanding the Standard

**[5.IC.4]** Intellectual property rights (IPR) are legal protections granted to creators and inventors for their original works, designs, and inventions. Intellectual property rights are crucial for protecting the creations of individuals, whether they are literary works, musical compositions, software, or artistic pieces.

**[5.IC.4a]** Copyright is the legal protection that gives creators of original works (like literature, art, music, software, etc.) This protection helps ensure that creators can control how their works are used and benefit financially from them, generally lasting for the lifetime of the creator plus 70 years. On the other hand, open-source licenses allow people to use, change, and share a computer program (like an app or a game) for free, as long as they follow some simple rules. Open-source licenses enable a different approach as they allow software to be freely used, modified, and shared by anyone.

The focus of open-source licenses is to foster collaboration and innovation, with various terms and conditions specified to guide usage and modification. Understanding the distinction between these two forms of Intellectual Property (IP) protection is essential for educators, as it can influence how educational materials, software, and other resources are utilized and shared within a learning environment. While copyright offers strict control and protection, open-source licenses encourage a community-driven approach to development and dissemination.

**[5.IC.4b]** In today's digital age, the accessibility of vast amounts of information through various digital sources presents both opportunities and significant risks. Among the most critical risks are privacy breaches, misinformation, and exposure to inappropriate content. Using unverified or unauthorized digital information sources can lead to the dissemination of false information, which not only misleads students but can also damage the credibility of educational institutions.

Furthermore, there are significant privacy concerns; students and teachers must be cognizant of the sensitivity of personal data and the potential for data misuse. Cybersecurity threats, such as phishing scams, present additional dangers when interacting with untrusted sources.

**[5.IC.4c]** Protecting online digital content and resources is crucial for educators to ensure that educational materials remain secure and accessible only for the intended audiences. One of the primary strategies involves using strong, unique passwords and regularly updating them

to prevent unauthorized access. It's also important to implement two-factor authentication (2FA) wherever possible, which adds an extra layer of security.

Two-Factor Authentication (2FA) is a security mechanism that requires two types of credentials to verify authorization. It is like adding an extra lock to your online accounts. Think of it as needing two keys to unlock a door. The first key is something you know, like your password. The second key is something you have, like a code sent to your phone.

Two-Factor Authentication

1. Enter Your Password: First, you log in with your usual password.
2. Verify with a Second Factor: Next, you'll be asked for a second piece of information. This might be a code sent to your phone, a fingerprint scan, or a special app that generates a unique number.

The idea is that even if someone steals your password, they will still need the second "key" to access your account, making it much harder for unauthorized people to get in. This extra layer of security helps protect your online accounts from hackers.

## Concepts and Connections

### Concepts

Open-source licenses and copyrights define the permissions and restrictions associated with using, modifying, and distributing digital content. Misuse of digital resources can lead to legal violations and loss of credibility. Protective strategies such as proper attribution, selecting appropriate licenses, and managing access are essential for safeguarding digital content.

### Connections

**Within this grade level:** At this grade level, students learn to distinguish between open-source licenses and copyright, understanding how each governs the use and sharing of digital content. They also investigate the risks associated with the inappropriate use of digital resources, such as plagiarism, unauthorized sharing, and security concerns.
**Vertical progression:** In Grade 4, students described the importance of intellectual property rights. They defined intellectual property rights and understood that copyright gives legal protection. Students learned how to give proper attribution. They defined the various digital works that are protected against unauthorized use (4.IC.4). In Grade 6, students will explain why various licenses are used along with comparing and contrasting the positives and negatives of the various licenses (6.IC.4). Grade 6 students will focus on the identification level of the various licenses.

## Across Content Areas

### English

- **5. DSR** The student will build knowledge and comprehension skills from reading a range of challenging, content-rich texts. This includes fluently reading and gathering evidence from grade-level complex texts, reading widely on topics to gain purposeful knowledge and vocabulary, and using reading strategies when comprehension breaks down.
- **5.RI** The student will use textual evidence to demonstrate and build knowledge from a variety of grade level complex informational texts heard or read. *(Note: only aligned if students are reading grade level informational texts).*
- **5.W** The student will compose various works for diverse audiences and purposes, linked to grade five content and texts.
- **5.C** The student will develop effective oral communication and collaboration skills to build a community of learners that process, understand, and interpret content together.

### History and Social Science

- **USI.8** The student will apply history and social science skills to explain westward expansion and reform in America from 1801 to 1861 by a. describing how territorial expansion affected the political map of the United States including, but not limited to the Louisiana Purchase, the Lewis and Clark Expedition and the role of Sacagawea, the acquisitions of Florida, Texas, Oregon, and California, and the results of the Mexican-American War; b. describing the causes, course of events, and effects of the War of 1812, the role of Andrew Jackson, and the development of the Monroe Doctrine; c. identifying geographic, economic, and religious motivations that influenced the movement of settlers; d. analyzing the impact of westward expansion on Indigenous people including, but not limited to the Indian Removal Act (1830), the Trail of Tears, and the Seminole Wars; e. explaining technological advancements and innovations and their effects on life in America including, but not limited to the cotton gin, the reaper, the steam engine, and the steam locomotive; f. describing major developments in the abolitionist and women's suffrage movements; and
  explaining how the expansion of U.S. territory led to increased momentum for the abolitionist and women's suffrage movements.

### Mathematics

- **5.PS.1** The student will apply the data cycle (formulate questions; collect or acquire data; organize and represent data; and analyze data and communicate results) with a focus on line plots (dot plots) and stem-and-leaf plots.

## Digital Learning Integration

- **3-5.DC** Students recognize the rights, responsibilities and opportunities of living, learning and working in an interconnected digital world, and they act in ways that are safe, legal, and ethical.
  A. Cultivate and manage their digital identity and reputation and are aware of the permanence of their actions in the digital world.
  C. Demonstrate an understanding of and respect for the rights and obligations of using and sharing intellectual property.

- **3-5.KC** Students critically curate a variety of digital resources using appropriate technologies, including assistive technologies, to construct knowledge, produce creative digital works, and make meaningful learning experiences for themselves and others.
  B. Evaluate the accuracy, perspective, credibility, and relevance of information, media, data, and other digital sources.
- **3-5.IC** Students communicate clearly and express themselves creatively for a variety of purposes using appropriate technologies (including assistive technologies), styles, formats, and digital media appropriate to their goals.
  B. Create original works or responsibly repurpose or remix digital resources into new creations.

## Opportunities for Computer Science Integration

Curriculum integration strengthens conceptual understanding and skill application. This can be done through multidisciplinary, interdisciplinary, and transdisciplinary approaches to integration. The examples below illustrate multiple ways to integrate computer science.

### English

- Students investigate risks related to the inappropriate use of digital sources, including plagiarism and misuse of copyrighted material. They learn to apply strategies such as citing sources, using open-source content, and setting permissions to protect and use digital information responsibly.

### History and Social Science

- Students examine primary and secondary sources, use open-source materials to introduce concepts of licensing and copyright in a real-world context. This helps students not only analyze historical evidence but also understand how digital content can be legally accessed, used, and shared, reinforcing both historical inquiry skills and responsible digital citizenship.

## Skills in Practice

Students should engage in the following practices to deepen their conceptual understanding and enhance the application of skills aligned with the Computer Science *Standards of Learning*. These practices are explained in more detail in Appendix A.

### D. Fostering Digital Literacy Practices:

1. Responsible Use Practices
2. Safeguard Well-Being of Self and Others
3. Evaluate Resources and Recognize Contributions

## 5.IC.5 The student will examine the effects of social interactions due to computing technologies.

a. List and explain how advances in computing technologies impact communication and collaboration.
b. Describe how computing technologies can be designed to engage and interact with users including those with diverse needs.
c. Evaluate activities conducted in the physical and online environments.
d. Create an artifact that illustrates a solution to address the need or want of a user.

## Understanding the Standard

In a technology-driven world, students must understand the profound effects computing technologies have on social interactions. These technologies influence how people communicate, form relationships, and engage with communities both online and offline. Understanding these impacts helps students navigate digital spaces responsibly and recognize the role technology plays in shaping culture and society.

**[5.IC.5a]** Advances in computing technologies have revolutionized communication and collaboration. The Internet, social media platforms, and cloud-based tools allow people to connect instantly and work together in real time, regardless of location. These innovations have created new forms of social interaction, supported remote work environments, and enabled diverse teams to collaborate effectively across geographical boundaries. As a result, computing technologies continue to reshape how individuals build communities, share ideas, and solve problems together.

Understanding these impacts empowers students to navigate and leverage these technologies responsibly, maximizing their potential for positive social interactions and collaboration.

**[5.IC.5b]** Computing technologies play a crucial role in creating inclusive and engaging experiences for users with diverse needs. These technologies must be designed with accessibility in mind, ensuring that interfaces are user-friendly for individuals with varying abilities. This includes features like voice recognition for those with mobility impairments, screen readers for the visually impaired, and customizable interfaces for users with cognitive differences.

**[5.IC.5c]** Evaluating activities conducted in physical environments involves examining how this environment supports interaction, engagement, and hands-on learning. Face-to-face settings often allow for real-time feedback, nonverbal communication, and collaborative group work, which can enhance participation and deepen understanding. These environments are particularly effective for activities that benefit from physical materials, movement, or interpersonal connection.

In online environments, evaluation focuses on how digital tools support accessibility, flexibility, and communication. Online platforms enable learners to participate from various locations, often with asynchronous and synchronous options. Effective online activities are structured with clear instructions, interactive elements, and opportunities for collaboration through features like breakout rooms, shared documents, or discussion boards.

## Concepts and Connections

### Concepts

Computing technologies transform communication and collaboration by enabling seamless interaction, content sharing, and connectivity across contexts. Thoughtful design ensures these tools accommodate diverse user needs through accessible and engaging interfaces. Assessing user experiences in both digital and physical settings supports the creation of targeted solutions that address specific challenges or preferences.

### Connections

**Within this grade level:** At this grade level, students describe how computing technologies can impact those with diverse needs (5.IC.5).
**Vertical progression:** In Grade 4, students examined the impact of screen time for themselves (4.IC.2). In Grade 6,

### Across Content Areas

**English**

- **5.C** The student will develop effective oral communication and collaboration skills to build a community of learners that process, understand, and interpret content together.
- **5.C.1** Prepare for and participate in a range of sustained collaborative discussions with diverse partners on grade five topics and texts. This includes: Listening actively and speaking using agreed-upon discussion rules; Respectfully demonstrating agreement or disagreement with others' ideas; Asking and answering relevant questions to build on others' ideas, clarify ideas, and acquire or confirm information; Summarizing the main ideas being discussed, using evidence, examples, and details to support opinions and conclusions; Share responsibility for the learning based on assigned roles and/or task expectations.

### Digital Learning Integration

- **3-5.GC** Students use appropriate technologies, including assistive technologies, to broaden their perspectives and enrich their learning by collaborating with others and working effectively in teams locally and globally.
  B. Use collaborative technologies to work with others, including peers, experts, and community members to examine issues and problems from multiple viewpoints.
  D. Explore local and global issues and use collaborative technologies to work with others to investigate solutions.

## Opportunities for Computer Science Integration

Curriculum integration strengthens conceptual understanding and skill application. This can be done through multidisciplinary, interdisciplinary, and transdisciplinary approaches to integration. The examples below illustrate multiple ways to integrate computer science.

### History and Social Science

- Students create a digital artifact that illustrates how technological innovations up to 1865, such as the printing press, telegraph, or steam engine, changed the way Americans communicated, shared ideas, and interacted socially. Students then compare these historical technologies to modern computing technologies and their current impact on communication and society

### Mathematics

- Students can take a poll to about how students feel about engaging in activities in the physical environment and online environment. Then they can use the data cycle to then evaluate the data of activities in physical and online environment.

## Skills in Practice

Students should engage in the following practices to deepen their conceptual understanding and enhance the application of skills aligned with the Computer Science *Standards of Learning*. These practices are explained in more detail in Appendix A.

### A. Fostering Computing Practices:

1. Building Relationships and Norms
2. Include Multiple Perspectives
3. Create and Accept Feedback
4. Use Collaboration Tools

### D. Fostering Digital Literacy Practices:

1. Responsible Use Practices
2. Safeguard Well-Being of Self and Others
3. Evaluate Resources and Recognize Contributions

Back to Impacts of Computing (IC)

## Networks and the Internet (NI)

### 5.NI.1 The student will identify and describe cloud computing.

a. Define cloud computing.
b. List examples of cloud computing.
c. List the advantages and disadvantages of cloud computing.
d. Identify safe practices and potential security risks when using cloud computing.

## Understanding the Standard

**[5.NI.1a]** Cloud computing is a resource sharing model that allows computing devices the ability to use the Internet to store and access data software applications. When information is stored "in the cloud" it is stored on remote server instead of your personal device, providing access from any device connected to the Internet. Cloud computing overcomes the limitations presented when resources are only locally available.

**[5.NI.1b]** Examples of cloud computing include Google Drive, Dropbox, OneDrive, streaming services including Netflix or Disney+, or Gmail.

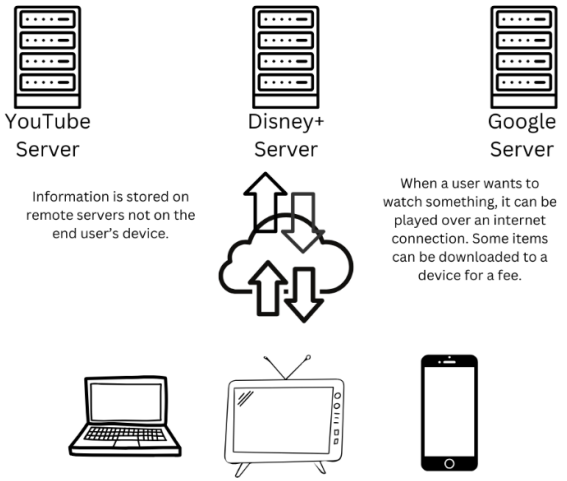



Image created using Canva

**[5.NI.1c]** Cloud computing has advantages and disadvantages. Advantages include the ability to access from anywhere, easier collaboration, automatic back-up, and saves space on local computing devices. Disadvantages include subscription fees, privacy and security concerns, and lack of access to information without Internet connection.

| Safe Practices of Cloud Computing | |
|---|---|
| Use Strong Passwords | Use unique and hard to guess passwords for your cloud accounts. |
| Don't share passwords | Use unique and hard-to-guess passwords for your cloud accounts. |
| Log Out After Use | Especially on shared devices like school computers, always log out when you're done. |
| Be Careful What You Share Online | Only share files with people you trust and be cautious with any personal information. |

**[5.NI.1d]** As with anything, there are risks involved with cloud computing. The user must determine if the benefits out way the risk and make it worth utilizing cloud computing. Additionally, it is important to know that sometimes cloud computing is the only available option.

| Security Risks of Cloud Computing | |
|---|---|
| Data Breaches | Hackers might try to break into cloud services to steal personal identifiable information. |
| Malware | Sometimes, files downloaded from the cloud can carry viruses. |
| Phishing Scams | Be cautious of fake emails or messages asking for your personal identifiable information or security measures such as a password. |

## Concepts and Connections

### CONCEPTS

Cloud computing provides access to data storage, processing power, and applications through internet-based services. It offers flexibility and scalability but also introduces security risks, making safe practices like encryption and secure authentication essential.

### CONNECTIONS

**Within the grade level/course:** At this grade level, students are beginning to explore the world of cloud computing (5.NI.1).
**Vertical Progression:** In Grade 4, students identified the interrelationship between computing devices and the computing network. They differentiated between tasks that require Internet access and those that don't (4.NI.1). In Grade 6, students will use computation thinking skills to take what they already know about cloud computing and make decisions using that information (6.NI.1).

### Across Content Areas

**English**

- **5.R** The student will conduct research and read a series of conceptually related texts on selected topics to build knowledge on grade-five content and texts, solve problems, and support cross-curricular learning.
- **5.R.1B** Identify search terms to locate information and gather relevant information from various print and digital sources to address the research. *(only if students are reading grade level informational texts)*

### Digital Learning Integration

- **3-5.GC** Students use appropriate technologies, including assistive technologies, to broaden their perspectives and enrich their learning by collaborating with others and working effectively in teams locally and globally.
  B. Use collaborative technologies to work with others, including peers, experts, and community members to examine issues and problems from multiple viewpoints.

## Opportunities for Computer Science Integration

Curriculum integration strengthens conceptual understanding and skill application. This can be done through multidisciplinary, interdisciplinary, and transdisciplinary approaches to integration. The examples below illustrate multiple ways to integrate computer science.

**English**

- Students can write letters to residents at local nursing homes explaining what cloud computing is. Students should provide examples of common uses of cloud computing currently being used in society. The letters should provide tips on how stay safe while using shared devices and utilizing cloud computing. Students should also ensure that they are only sharing information that is safe.

**Math**

- Students can research the cost of computer storage options ranging from USB drives, external hard drives, and available cloud drive options. After the data is collected, students can solve to determine the cost per gigabyte of storage for each option. Students can analyze the data to make recommendations on the best storage options compared to the cost. Students may also explore file sizes to help determine the amount of storage space that might be needed at a given time. Files sizes to explore would include an average movie, a musical album, and text documents. Students could consider the occupation of a fictional individual when making recommendations.

## Skills in Practice

Students should engage in the following practices to deepen their conceptual understanding and enhance the application of skills aligned with the Computer Science *Standards of Learning*. These practices are explained in more detail in Appendix A.

### B. Fostering Computational Thinking Practices:

1. Decompose Real-World Problems
2. Explore Common Features and Identify Patterns

### D. Fostering Digital Literacy Practices:

1. Responsible Use Practices
2. Safeguard Well-Being of Self and Others
3. Evaluate Resources and Recognize Contributions

Back to Networks and the Internet (NI)

# Appendix A

## K-5 Computer Science Skills and Practices Continuum

Students develop essential practices: collaboration, computational thinking, iterative design, and digital literacy. Students use these practices to engage with core computer science concepts, create artifacts, and problem-solve across disciplines. Artifacts can include but are not limited to prototypes, programs, planning documents, animations, or abstractions (e.g., visualizations, storyboards, flowcharts, decision trees, models, computer simulations).

| A. **Fostering Collaboration in Computing Practices** |
|---|
| 1. **Building Relationships and Norms:**<br>• **K-2**: Students work collaboratively with others. Students take turns in different roles on the project.<br>• **3-5**: Students work collaboratively with others. Students practice assigning roles within their teams and recognize group member strengths. |
| 2. **Include Multiple Perspectives:**<br>• **K-2:** Students differentiate their technology preferences from the technology preferences of others. Students will be presented with perspectives from people with different backgrounds, ability levels, and points of view.<br>• **3-5:** Students discuss design choices, compare preferences, ask questions, and seek input from group members with diverse abilities, experiences, and perspectives. |
| 3. **Create and Accept Feedback:**<br>• **K-2**: With teacher scaffolding, students seek help and share ideas to achieve a particular purpose. Students ask questions of others and listen to their opinions.<br>• **3-5:** Students provide and receive feedback related to computing in constructive ways. For example, pair programming is a collaborative process that promotes giving and receiving feedback. |
| 4. **Use Collaboration Tools:**<br>• **K-2:** Students collaboratively brainstorm by writing on a whiteboard or paper.<br>• **3-5:** Students use collaboration tools to manage teamwork and utilize online project spaces. They also begin to make decisions about which tools would be best to use and when to use them. |
| **Instructional Considerations for Collaboration Practices** |

Possible **instructional approaches** to foster collaboration practices:

1. Design instruction around authentic problems that require collaboration. Assign roles, provide clarifying and probing question stems, and model strategies students can use to identify and advocate for their needs.
2. Provide resources to support exploring different viewpoints and end users. Model curiosity, perspective-taking, and empathy.
3. Model sentence stems for constructive feedback, establish routines for self and group reflection, and practice incorporating diverse viewpoints. Implement pair programming with opportunities to practice giving and receiving feedback.
4. Model tool selection and project management structures. Provide opportunities to practice various methods and reflect.

**Instructional activities** may include but are not limited to:

- **Classroom Discussion:** Organize discussions that engage students in hearing differing perspectives.
- **Timeline Creation:** Have students create or evaluate and modify timelines that illustrate the steps needed to complete a task as a group.
- **Simulated Shark Tank Innovation Challenge:** Create an innovation design challenge where students collaboratively apply computer science content to solve a problem or launch a new idea.
- **Case Studies:** Provide case studies of design decisions that real computer scientists face and have students analyze and present their recommended choices based on computer science content knowledge.

## B. Fostering Computational Thinking Practices

1. **Decompose Real-World Problems:**
   - **K-2:** Students break problems, information, and processes into parts. Identify relationships and connections among parts. Reflect on how decomposition aids problem-solving across contexts.
   - **3-5:** Students further break problems into subproblems, apply systems thinking to explore interdisciplinary connections and integrate existing solutions or procedures (i.e. classroom processes, math procedures, school routines) Apply algorithms to break a problem into subtasks that can be solved and combined to solve the main problem.

2. **Explore Common Features and Identify Patterns:**
   - **K-2:** Students will be able to identify and describe repeated sequences in data or code through analogy to visual patterns or physical sequences of objects. Students will identify patterns, such as recognizing repeated patterns of code that could be more efficiently implemented as a loop.
   - **3-5:** Students analyze patterns to develop generalizations and models, test their limits, and validate inputs. Use patterns to analyze trends, justify design decisions, and create artifacts.

3. **Use Abstraction to Simplify, Represent, and Problem Solve:**
   - **K-2:** Students use and/or create abstractions (e.g., storyboards, flowcharts, decision trees, models) to simplify problems, represent information, organize thinking, communicate, and create artifacts. Artifacts can include but are not limited to prototypes, programs, planning documents, and animations.
   - **3-5:** Students use and/or create abstractions (e.g., visualizations and computer simulations) to simplify problems, represent information, organize thinking, communicate, and create artifacts. Students intentionally use abstractions to support the problem-solving process to aid in understanding, planning, and predictions.

4. **Apply Algorithmic Thinking to Problem Solve and Create:**
   - **K-2:** Students use algorithmic thinking to develop a sequence of steps to plan, create, test, and refine artifacts with and without technology.
   - **3-5:** Students use pseudocode and generalizations to organize, create and seek and incorporate feedback on more complex designs.

5. **Apply Computational Thinking Practices to Select, Organize, and Interpret Data:**
   - **K-2:** Students use computational thinking to organize data and make predictions. Explore parts and relationships within data sets.
   - **3-5:** Students visualize data. Use patterns and algorithmic thinking to organize data, identify trends, and make predictions. Use decomposition to explore parts and relationships within data sets. Ask questions about available data sources, and compare and analyze test results to inform decisions, plan, and refine designs.

**Instructional Considerations for Computational Thinking Practices**

Possible **instructional approaches** to foster computational thinking practices:

1. Model strategies for breaking complex information into smaller parts. Provide opportunities to analyze and discuss the relationship among parts.
2. Support students with recognizing patterns. Model how to analyze, interpret, and display patterns to make predictions and draw conclusions.
3. Model the use of abstraction (e.g., visualizations, storyboards, flowcharts, decision trees, models, computer simulations) to simplify problems; represent information (e.g., data, patterns, processes, phenomena, systems); organize thinking; and support sense-making. Support students with creating and evaluating abstractions and their limitations.
4. Plan opportunities for students to use sequencing in problem solving, incorporate user feedback, and check for bias, accessibility, and other design criteria. Model ways to systematically test, validate, evaluate, refine, and optimize algorithmic solutions. Provide opportunities to reflect on how algorithms are used in solutions.
5. Model abstraction, pattern analysis, and decomposition. Use models to develop and test predictions. Identify limitations and benefits of models.

**Instructional activities** may include but are not limited to:

- **Create an Artifact**: Students could create an app, program, animation, simulations, etc. to solve a community problem or creatively express an idea.
- **Identify Patterns to Make Predictions**: Students notice repetition in sequences of numbers or parts of a process to make predictions about future events or missing components.
- **Create Abstractions:** Students choose the best tool to use for problem solving using abstractions. Discuss which tools worked best for the team and the problem. Tools may include models, visualizations, storyboards, flowcharts, decision trees, generalizations, simulations.
- **Create Models:** Develop models to represent information such as patterns, relationships, inputs/outputs. Create models of systems (e.g., model networks, cybersecurity, emerging technologies) to understand how parts connect to perform a function. Students can create models to engage in systems thinking and modularization.
- **Evaluate existing models and programs:** Evaluate outputs for bias, accessibility, reliability or other established design criteria. Students can identify applicable parts or modules of existing programs and reuse to solve different problems.
- **Reflection and Transfer:** Have students reflect on how each computational practice facilitates problem-solving and identify opportunities to apply the practice to other situations. Support students identifying key points in feedback.

## C. Fostering Iterative Design Practices

1. **Identify, Define, and Evaluate Real-world Problems:**
   - **K-2:** With guidance from an educator, identify, define, and explore existing problems and potential solutions. Ask questions to view problems from different perspectives.
   - **3-5:** Students identify, define, and explore existing problems and potential solutions. Ask questions to understand problems from different perspectives. Clarify success criteria, identify constraints, and uncover missing information. Explore patterns and develop generalizations about the types of problems that benefit from computational solutions.

2. **Plan and Design Artifacts:**
   - **K-2:** With guidance from an educator, students will generate ideas for new solutions, incorporate peer feedback, and reflect on impact of diverse perspectives. Use tools like class or group discussions, outlines, flowcharts, and storyboards to plan prototypes.
   - **3-5:** Students will generate ideas for new solutions, incorporate peer feedback, and reflect on the impact of diverse perspectives. Use tools like outlines, flowcharts, and storyboards to plan prototypes. Predict the performance and impacts of prototypes, including potential errors, user needs, and accessibility.

3. **Create, Communicate and Document Solutions:**
   - **K-2:** Students create artifacts with or without technology, such as algorithms and programs using plans and outlines. Describe design choices and make connections to the design challenge, criteria, and constraints. Engage in giving and receiving feedback enhances communication skills.
   - **3-5:** Students create artifacts, such as algorithms and programs using plans and outlines. Describe design choices and make connections to the design challenge, criteria, and constraints. Engage in giving and receiving feedback to refine solutions and enhance communication skills.

4. **Test and Optimize Artifacts:**
   - **K-2:** Students test artifacts to ensure they meet criteria and constraints, comparing results to intended outcomes. Use computational thinking and other problem-solving strategies like trial and error to fix simple errors, debug, revise, and evaluate artifacts against design criteria.
   - **3-5:** Students test artifacts to ensure they meet criteria and constraints, comparing results to intended outcomes. Use computational thinking and other problem-solving strategies like trial and error to fix simple errors, debug, revise, and evaluate artifacts against design criteria. Reflect on how the iterative design and computational thinking practices facilitate program development.

**Instructional Considerations for Iterative Design Practices**

Possible **instructional approaches** to foster iterative design practices:

1. Design learning experiences where students identify real-world problems and evaluate the appropriateness of using computational tools to develop solutions.
2. Provide instructional time and model strategies to support students with using an iterative process to plan the development of an artifact while considering key features, time and resource constraints, and user expectations. Design instruction to provide students with multiple paths to solve problems.
3. Provide instructional time for students to prototype**,** justify, and document computational processes and solutions using iterative processes. Model how to listen to differing ideas and consider various approaches and solutions.
4. Provide instructional time and model strategies for evaluating artifacts using systematic testing and iterative refinement to enhance performance, reliability, usability, and accessibility as outlined in the design criteria.

**Instructional activities** may include but are not limited to:

- **Class Discussions:** Discuss pros and cons of using computing technologies to solve real-world problems. Consider examples like drones monitoring the environment; AI-generated art; or personalized learning applications. Progressive examples include, using machine learning in self-driving cars to interpret road conditions and make decisions, and robots assisting in surgeries for precision and reduced recovery times.
- **Prototype and Improve:** Create simple animated stories, solve pre-existing problems, and utilize coding platforms to simulate solutions. Incorporate available technology to develop physical models. Use peer feedback to refine designs and document changes while justifying improvements at each step.
- **Debug and Enhance:** Work with a pre-built program containing intentional errors and limited features to debug to optimize the program for performance and enhance it with new capabilities.
- **Accessibility Upgrade:** Emphasize empathy and inclusion in design by analyzing an existing program or interface (e.g., a basic website). Evaluate it for usability and accessibility. Propose iterative changes to improve the design, such as adding features like text-to-speech, adjustable font sizes, or simplified navigation and implement when available and appropriate.

## D. Fostering Digital Literacy Practices

1. **Responsible Use Practices:**
   - **K-2:** Students use technology in ways that are safe, legal, and ethical. Implement strategies to protect their digital identity, personal data, and the data of others.
   - **3-5:** Explore and ask questions about how computer science and emerging technologies work, and their benefits and risks. Students explore data privacy rights, data protections, terms of service and privacy policies. Weigh tradeoffs and risks with actions and decisions involving computer science.

2. **Safeguard Well-Being of Self and Others:**
   - **K-2:** Students reflect on their emotional response to the use of digital technology. Consider how the use of technology can impact others and make choices that benefit others and avoid harm. Identify the roles and responsibilities of humans in designing and using technologies. Practice empathy and engage in positive online practices as an upstander.
   - **3-5:** Students reflect on their emotional response to the use of digital technology and identify how to use technology in ways that support personal well-being. Consider how the use of technology can impact others and make choices that benefit others and avoid harm. Identify the roles and responsibilities of humans in designing and using technologies. Practice empathy and engage in positive online practices as an upstander.

3. **Evaluate Resources and Recognize Contributions:**
   - **K-2:** Students apply strategies for evaluating the accuracy, validity, accessibility, reliability, appropriateness, credibility, and relevance of digital sources.
   - **3-5:** Students apply strategies for evaluating the accuracy, validity, accessibility, reliability, appropriateness, credibility, and relevance of digital sources. Keep track of sources of information and give credit to the creators of information. Students evaluate the bias and relevance of sources. Identify false or misleading information.

### Instructional Considerations for Digital Literacy Practices

Possible **instructional approaches** to foster digital literacy practices:

1. Model how to use technology in ways that are safe, legal, and ethical. Model how to make decisions about data privacy and information sharing that protect individual and peer identify and digital footprint. Incorporate learning activities like discussions of digital dilemmas that help students explore different perspectives, benefits, risks, and tradeoffs.
2. Incorporate opportunities for students to reflect on possible positive and negative impacts of how they use computing technologies. Choose instructional technology that aligns with learning goals and use data on students learning to reflect on and assess the extent to which the technology is supporting learning outcomes. Provide opportunities to identify the role of humans in developing and using technology.

3. Model strategies for how to investigate the credibility of information sources and give appropriate attributions for content created by others.

**Instructional activities** may include but are not limited to:

- **Source Evaluation:** Assign students articles. Have students distinguish between fact and opinion within articles and evaluate the reliability of the sources.
- **Comparative Analyses:** Encourage students to explore ethical dilemmas, compare different approaches to data privacy and possible impacts across different time periods using evidence to support arguments.
- **Class Discussions:** Organize discussions where students take roles representing different perspectives and defend their positions.
- **Digital Dilemmas:** Discuss case studies of complex topics that do not have one right answer such as the CommonSense Education digital dilemmas.

# Appendix B

## Grade 5 Computer Science Vocabulary

| Vocabulary Word | Definition |
|---|---|
| Abstraction | A filtering process used to create a simplified representation of relevant data to identify essential details, excluding less important details. |
| Artificial Intelligence (AI) | A domain of computer science that focuses on the research and development of computers and systems that simulate or replicate tasks that require human intelligence. |
| Algorithm | Finite and specified set of step-by-step instructions designed to solve a problem or perform a task. |
| Algorithmic Thinking | Process of developing algorithms in a logical, systematic, and procedural way to solve problems or complete tasks. |
| American Standard Code for Information Interchange (ASCII) | A character encoding standard that represents text in numeric form, enabling multiple data representations in computing devices. |
| Acceptable Use Policy (AUP) | Rules and guidelines that define safe practices and responsible use of technology. |
| Assigning | Setting a value to a variable either manually or based on program logic. |
| Attribution | Giving credit for work created by someone else. |
| Authentication | A process used to verify a user's identity before accessing a network or computer system. |
| Author | The creator of a book, image, song, or object. |
| Automated Decision Making | The use of predefined algorithms or programs that enable a computing device the ability to make decisions independently based on the information it receives. |
| Binary | A number system that uses two digits, 0 and 1. |
| Bit | The smallest unit of data in computing. |
| Block-Based Programming | A visual drag and drop programming tool that users can use to create programs using command blocks. |
| Boolean | A data type that can only have two possible values: true or false; essentially representing a logical state where something is either true or not. |
| Byte | A unit of digital information that's made up of eight binary digits, or bits. |

| | |
|---|---|
| Character | A person or animal in a book, story, movie, or project. A letter, number, or symbol used in a password. |
| Cleaning | Making sure that the data is correct, consistent, and clear. |
| Client | A computer or software application that retrieves and uses information, resources, or services from another device over a network. |
| Cloud Computing | Storing and accessing information on the Internet. |
| Code | Any set of instructions expressed in a programming language. |
| Collecting | Gathering the appropriate type of data needed. |
| Computational Artifacts | Any creation made by a human using a computing device. It can include but are not limited to prototypes, programs, planning documents, animations, or abstractions (e.g. visualizations, storyboards, flowcharts, decision trees, models, computer simulations). |
| Computational Thinking | A logical and systematic problem-solving process that uses decomposition, pattern recognition, abstraction, and algorithm thinking to foster creativity and develop solutions. |
| Computer Science | The study of computers and algorithmic processes, including their principles, their hardware and software designs, their applications, and their impact on society. |
| Computer System | Integrated group of hardware and software that work together to store, process, and manage data. |
| Computing Device | An electronic device that can receive input, process data, store information, and produce output based on instructions (programs). |
| Conditional Control Structures | Conditional logic (e.g., if-else statements) to make decisions within a computer program. |
| Copyright | Legal protection that gives creators of original works (like literature, art, music, software, etc.) exclusive rights to use and distribute their creations. |
| Cybersecurity | Protection of data and information on networks and computing devices from unauthorized access, attacks, damage, or theft. |
| Data | Individual pieces of information about people, things, or events that can be processed, stored, and analyzed by computing devices. |
| Data Breach | Access or theft of private or sensitive information without authorization. |
| Data Cycle | Process of formulating questions to be explored with data, collecting or acquiring data, organizing and representing data, and analyzing and communicating results. |
| Data Representation | How data is visually represented, such as in graphs or charts. |

| | |
|---|---|
| Data Visualization | The representation of data through use of common graphics, such as charts, plots, infographics and even animations to make complex data more accessible and understandable. |
| Debug | Process of identifying, isolating, and fixing errors (often referred to as "bugs") in a set of instructions, code, or system. This can also include hardware and software. |
| Decomposition | Process of breaking down a problem, process, or task into smaller, more manageable components. |
| Design | Creation of a plan or prototype of a proposed solution. |
| Design Document | A detailed plan that outlines the structure, features, and implementation strategy of a project. It serves as a blueprint, providing clear specifications, goals, and guidelines for developers, designers, and stakeholders. Design documents often include diagrams, technical requirements, workflows, and rationale to ensure a shared understanding of the project's direction and execution. |
| Diagrams | Visual representation of data, information, or concepts. |
| Digital Divide | The difference in access to technology and the Internet between people who have access and those who do not. |
| Diverse | Variety of different elements, qualities, or characteristics. |
| Diverse Datasets | Datasets that include a variety of data types or data sources. |
| Encryption | The conversion of electronic data into another form, called ciphertext, which cannot be easily understood by anyone except authorized parties. |
| Evaluation | Assessment process that reviews test results and feedback to determine a design or product's effectiveness and identify necessary changes for improvement. |
| Event | An action or something that causes all of a program or only a certain portion of the program to run, e.g., a mouse click on the run block. |
| Expression | In a programming language, a combination of explicit values, constants, variables, operators, and functions interpreted according to the particular rules of precedence and of association which computes and then produces (returns, in a stateful environment) another value. |
| Extraction | Process of identifying, isolating, or deriving meaningful information. |
| Fair Use | Allows the limited use of copyrighted material without the copyright owner's permission. |

| | |
|---|---|
| Flowchart | A diagram that shows the steps in a process using shapes and arrows. It helps the user visualize how things happen in order. |
| Function | Like variables, except instead of storing data they store lines of code. Help to simplify the programming process and make code more readable. |
| Hardware | The physical components of a computing device that you can touch, such as the processor, memory, keyboard, and display. |
| Healthy Screen Habits | Practices that emphasize balanced use of digital devices to support physical, mental, and emotional well-being. |
| Implementation | The development or execution of a functional prototype, program, or product. |
| Inclusive | The dataset includes different groups of things that are important. |
| Information | Facts provided or learned about something or someone. |
| Input | Information or action you give to a computer or device to tell it what to do. |
| Input Devices | Hardware components that allow users to enter data into a computing device. |
| Initialize | Defining a variable and assign it an initial value. |
| Intellectual Property | A person's own creations of the mind, such as inventions, drawings, stories, and poems. |
| Intellectual Property Rights (IPR) | Legal protections granted to creators and inventors for their original works, designs, and inventions. |
| Internet | A global network of interconnected computing devices that allows devices to share information and resources. |
| Internet Protocol (IP) Address | A unique numeric value assigned to a computer or other device connected to the Internet so that it may be identified and located. |
| Iteration | Repeated actions. |
| Iterative Design Process | A systematic approach to creating and refining products, systems, or solutions through repeated cycles of design, evaluation, and improvement. |
| Key | A distinct identifier used to differentiate data elements within a set. |
| Large Datasets | Collection of large amounts of data that require specialized tools or methods to store, process, and analyze. |
| Library | A collection of books and periodicals. |

| | |
|---|---|
| List | A data structure that stores an ordered collection of elements, which can be of any type (numbers, strings, objects, etc.) |
| Logic Errors | An error that occurs when a program is executed and produces incorrect or unintended output without causing the program to crash or display an error message. |
| Loop | A set of instructions that are repeated until a specified condition is met, or a predetermined number of repetitions has occurred. |
| Machine Learning | A process that occurs when computers learn from examples instead of following exact instructions. Examples of machine learning include supervised learning, unsupervised learning, and reinforcement learning. |
| Malware | Malware is malicious software that can steal data, corrupt files, disrupt services, and/or damage networks and computing systems. |
| Memory | The physical storage in computing devices where data is processed and instructions for processing are stored. Memory types include RAM (Random Access Memory), ROM (Read-Only Memory), and secondary storage like hard drives, removable drives, and cloud storage. |
| Model | A simplified representation of an idea, object, system, or process that helps describe, test, or predict how something works often using diagrams, simulations, or code. |
| Naming Convention | Guidelines for the use of descriptive names for variables, functions, and classes. |
| Nested Conditional Control Structures | Conditional statements placed inside the body of another conditional statement. |
| Networking | A group of computing devices (personal computers, phones, servers, switches, routers, etc.) connected by cables or wireless media for the exchange of information and resources. |
| Numerical Data | Are values or observations that can be measured (e.g., quantitative). It may include heights, temperatures, scores or grades, or statistics. |
| Open-source License | People can use, change, and share a computer program (like an app or a game) for free, as long as they follow some simple rules |
| Operator | A symbol that establishes a relationship between two values. Common types include logical (AND, OR, NOT), relational (=, <, ≤, >, ≥), and arithmetic (+, -, ÷, ×) operators. |
| Output | Data or information produced by a computing device after processing input. |
| Output Devices | Hardware components that display processed data or information. |

| | |
|---|---|
| Packet | Packets are smaller pieces of data that can be sent across networks. |
| Password | A secret word or phrase used to protect devices and information from unauthorized access. |
| Pattern Analysis | Process of identifying commonalities, differences, and predictable relationships within data to understand, interpret, and make predictions. |
| Pattern Recognition | Ability to identify commonalities, similarities, or differences in recurring elements. |
| Personal Information | Data or information about a person that relates to their identify, characteristics, or activities. |
| Phishing | Deceptive online attack where scammers pretend to be a trusted source and trick individuals to share personal identifiable information. |
| Pixel | Picture element or tiny square of color that, when combined with other pixels, comprises a larger image. |
| Plain Language | A description of the steps and logic in simple terms that anyone can understand through the use of familiar analogies, real-life examples, and simple terms. |
| Private Information | Sensitive data or information that can identify a person or give others access to your personal life. |
| Problem Definition | Clearly identifying the problem or challenge that needs to be solved. |
| Procedure | Broadly used to refer to a process, which may include a method, function, subroutine, or module, depending on the programming language. |
| Processing Speed | How fast a computer can receive input, process the information, and complete tasks. |
| Program | The implementation of an algorithm (set of instructions) translated into a programming language that a computer can follow and execute to perform a specific task. |
| Programming Language | A structured system for writing instructions that a computer can understand and execute. It includes syntax, which defines the rules for how code is written, and semantics, which conveys the meaning of the instructions. Programming languages enable developers to build software, automate processes, and control computer hardware. |
| Prototype | A preliminary version of a final product or information system, typically created for demonstration purposes. |
| Pseudocode | An algorithm written in plain language instead of a programming language. |
| Public Information | Information that is okay to share with anyone and is typically available for everyone to see. |
| Purposeful Use of Computing Devices | Includes the understanding that technology has specific purposes, such as learning new things, solving problems, and communicating. |

| | |
|---|---|
| Ransomware | A type of malicious software (malware) that encrypts a victim's data or locks them out of their system, demanding payment (a ransom) to restore access. It is commonly spread through phishing emails, malicious attachments, or software vulnerabilities. |
| Reboot | To turn off the device and turn it back on. |
| Reinforcement Learning | A type of machine learning where a computer learns through trial and error. The computer receives feedback and adjusts its behavior to improve performance. |
| Router | A device that directs data from one network to another. |
| Screen Time | Time spent on a computing device. |
| Selection | Using conditions to manage the sequence of a program's execution. |
| Sensor | A computing component that detects, collects, or measure data that would otherwise be difficult to gather manually. |
| Sequence | The specific order in which instructions or steps are executed in an algorithm or program. |
| Server | A computer or software application that provides information, resources, or services to clients over a network. |
| Simulation | Replicating the behavior of a real-world process or system over a period of time. |
| Social Engineering | The process of bypassing computer security and gaining access to a device or system by tricking users into revealing passwords or other secure information. |
| Social Media | Applications that allow people to socialize, communicate, and share content with each other. |
| Software | A set of instructions that tells the computer how to act and respond but cannot be seen or touched. |
| Sort | To compare a set of objects in order to find similarities and differences, so that they may be arranged and organized. |
| Spyware | A type of malware designed to secretly monitor and collect information about a user’s activities without their knowledge or consent. |
| Storage | Location where data, programs, and files are kept permanently (until deleted). |
| Storage Capacity | The amount of space available to save data, which includes photos, sound, and files. |
| String | A string is a series of letters, numbers, or symbols. It can represent things like a name, address, or song title. Common operations with strings include finding their length, joining them together, and extracting parts of them. |

| | |
|---|---|
| Supervised Learning | A method of machine learning that involves a computer using large amounts of labeled datasets to recognize patterns, classify data, and make predictions. |
| Syntax | Rules or structure of a programming language. |
| Syntax Errors | An error caused by a mistake in the rules or structure of a programming language, such as missing parentheses, commas, or incorrect keywords. |
| Table | A structured format to organize and record information in rows and columns. |
| Testing | The process of evaluating a program or system to assess its results and outputs, ensuring accuracy, performance, and reliability, while identifying errors. |
| Training Data | A collection of labeled or unlabeled data used to teach a machine learning model how to recognize patterns, make predictions, or perform tasks. |
| Trends | Are long-term directions or movements in data or behavior that indicate a general tendency or shift over time. |
| Troubleshoot | Process used to diagnose why a system or process is not working as expected and systematically testing solutions to resolve the issue. |
| Two-Factor Authentication (2FA) | Security mechanism that requires two types of credentials to verify authorization. |
| Two-way Branching Conditional Control Structures | A programming concept that allows a program to make a decision based on two different paths. |
| Unauthorized Access | Information is accessed without the permission of the owner. |
| Unsupervised Learning | When a computer learns to find groups or patterns without anyone telling it what's right or wrong. |
| User Input | Data or information that a user provides to a computer program during its execution (2024). Data that is taken in by a computer for processing (2017). |
| Username | A unique name that people use to log into a device or online account. It is like a nickname that helps the computer recognize who is logging in. |
| Variables | A programming element that is a named storage location in memory that holds a value, which can be modified during the execution of a program. |
| Video Call | A live conversation where people can see and hear another person while talking through a connected device, like a tablet or computer. |

| | |
|---|---|
| Video Conference | A meeting where people in different locations use video and audio technology to communicate in real-time allowing participants to see and hear each other as if they were in the same room. |
| Virus | Malicious software (or malware) designed to spread from one computer to another, often without the user's knowledge. |
| Visualizations | Refer to graphical representations of data or information that help users understand patterns, trends, and relationships more effectively. Visualizations make complex data more accessible, interpretable, and actionable. |
| Websites | A collection of webpages on the Internet that people can visit using a web browser. |
| Wi-Fi | The device that allows computing devices to access the Internet without being connected to physical cables within a specific area using radio waves to send and receive data. |
| World Wide Web (WWW) | A system of interconnected web pages that users can access through the internet. |
| Worms | A type of malware designed to replicate itself and spread across computers or networks without needing to attach to a host program or file. |