K-6 CS Curriculum Framework

The K-6 Curriculum Framework is **a model for how CS concepts and standards may be addressed in a sequence of theme-based units.**

The Scope (what is taught) and Sequence (in what order is it taught) is available in **the Scope & Sequence Chart here.**

In the Framework, each grade level has two units.

Unit 1 explores the non-programming aspects of CS, addressing these core concepts:

1. ***Computing Systems***
2. ***Network & Internet***
3. ***Data & Analysis***
4. ***Impacts of Computing***

Most activities require no ("unplugged") or very little ("low tech") technology.

Unit 2 explores the programming aspects of CS, addressing this core concept:

Algorithms & Programming

Unit 2 is built upon device-based activities and may be taught with the use of either:

1. A single vendor package (curriculum with proprietary or recommended programming platform, supporting technology tools)
2. A combination of device-based and unplugged activities and technology tools

Units By Grade Band

K-2 "Computing and Life"

	Unit 1	Unit 2
Kindergarten	Computers and Life	Algorithms and Life
Grade 1	Networks and Life	Computers and Algorithms
Grade 2	Computer Systems, Data, and Life	Program Development

3-5 “Computing & Community”

	Unit 1	Unit 2
Grade 3	The Connected World	Introduction to Programming
Grade 4	Technology, Collaboration, and Cultural Diversity	Computational Participation
Grade 5	Computers and Information	Programming and Problem Solving

6-8 “Computing & People”

	Unit 1	Unit 2
Grade 6	The Internet and the Web	Website Development

K - 1: Computers and Life

Enduring Understanding

People use computing devices to perform a variety of tasks accurately and quickly. Computing devices interpret and follow the instructions they are given literally. Computing technology has positively and negatively changed the way people live and work. Computing devices can be used for entertainment and as productivity tools, and they can affect relationships and lifestyles.

Essential Questions

What is a computer?
What does a computer do?
Why do we need a computer?
How can computers affect lifestyles and relationships?

Core Concepts:

Computing Systems (Devices)
Impacts of Computing (Culture)

CSTA Standards

1A-CS-02 Use appropriate terminology in identifying and describing the function of common physical components of computing systems (hardware). (P7.2)
A computing system is composed of hardware and software. Hardware consists of physical components. Students should be able to identify and describe the function of external hardware, such as desktop computers, laptop computers, tablet devices, monitors, keyboards, mice, and printers.

1A-CS-01 Select and operate appropriate software to perform a variety of tasks, and recognize that users have different needs and preferences for the technology they use. (P1.1)
People use computing devices to perform a variety of tasks accurately and quickly. Students should be able to select the appropriate app/program to use for tasks they are required to complete. For example, if students are asked to draw a picture, they should be able to open and use a drawing app/program to complete this task, or if they are asked to create a presentation, they should be able to open and use presentation software. In addition, with teacher guidance, students should compare and discuss preferences for software with the same primary functionality. Students could compare different web browsers or word processing, presentation, or drawing programs.

1A-IC-16 Compare how people live and work before and after the implementation or adoption of new computing technology. (P7.0)
Computing technology has positively and negatively changed the way people live and work. In the past, if students wanted to read about a topic, they needed access to a library to find a book about it. Today, students can view and read information on the Internet about a topic or they can download e-books about it directly to a device. Such information may be available in more than one language and could be read to a student, allowing for great accessibility.

Related Resources and Toolkits
Sample Lessons View Samples
Lesson 1 and 2: What's a computer?
Lesson 3 and 4: That could be computer
Lesson 5 and 6: Computer all around us

- Find lesson plans and instructional materials to support Unit 1 in Lesson Resources.
- Browse books that introduce CS concepts in the Reading List.
- Explore more CS unplugged activities here.

K - 2 : Algorithms and Life

Enduring Understanding

People follow and create processes as part of daily life. Many of these processes can be expressed as algorithms that computers can follow. Computers follow precise sequences of instructions that automate tasks. Information in the real world can be represented in computer programs. Programs store and manipulate data, such as numbers, words, colors, and images.

Essential Questions

How do people use algorithms in daily life routines and in the real world?
How does a computer work?

Core Concepts:

Algorithms and Programming

CSTA Standards

1A-AP-08 - Model daily processes by creating and following algorithms (sets of step-by-step instructions) to complete tasks.

Composition is the combination of smaller tasks into more complex tasks. Students could create and follow algorithms for making simple foods, brushing their teeth, getting ready for school, participating in clean-up time.

1A-AP-09 - Model the way programs store and manipulate data by using numbers or other symbols to represent information.

Information in the real world can be represented in computer programs. Students could use thumbs up/down as representations of yes/no, use arrows when writing algorithms to represent direction, or encode and decode words using numbers, pictographs, or other symbols to represent letters or words.

1A-AP-10 - Develop programs with sequences and simple loops, to express ideas or address a problem.

Programming is used as a tool to create products that reflect a wide range of interests. Control structures specify the order in which instructions are executed within a program. Sequences are the order of instructions in a program. For example, if dialogue is not sequenced correctly when programming a simple animated story, the story will not make sense. If the commands to program a robot are not in the correct order, the robot will not complete the task desired. Loops allow for the repetition of a sequence of code multiple times. For example, in a program to show the life cycle of a butterfly, a

loop could be combined with move commands to allow continual but controlled movement of the character.

1A-AP-11 - Decompose (break down) the steps needed to solve a problem into a precise sequence of instructions.
Decomposition is the act of breaking down tasks into simpler tasks. Students could break down the steps needed to make a peanut butter and jelly sandwich, to brush their teeth, to draw a shape, to move a character across the screen, or to solve a level of a coding app.

1A-AP-14 - Debug (identify and fix) errors in an algorithm or program that includes sequences and simple loops.
Algorithms or programs may not always work correctly. Students should be able to use various strategies, such as changing the sequence of the steps, following the algorithm in a step-by-step manner, or trial and error to fix problems in algorithms and programs.

Equity and UDL

It's recommended that educators begin with Unplugged Activities to introduce CS concepts before moving onto Plugged (Device-based) Activities and Physical Computing (Robotics and Microcontrollers).

In the light of UDL, teachers are encouraged to offer flexibility in their classrooms. The goal of UDL is to use a variety of teaching methods to remove any barriers to learning. In the same way, we're here to provide you with many different resources and materials that you can choose to meet the needs of your students.

Related Resources and Toolkits

Curriculum

- Code.org: Pre-reader Express; CS Fundamentals Course A
- Apple: Everyone Can Code: Get Started with Code 1 (Tynker; CodeSpark); Everyone Can Code Puzzles.
- BootUp curriculum (K-2 with Scratch Jr.)

Programming Platforms

- Scratch Jr.
- CodeMonkey
- Kodable
- CodeSpark (The Foos)
- Tynker

Physical Computing Extensions

- BeeBot

- KIBO
- Puzzlets
- Dash and Dot
- Osmo
- Ozobot

Sample Lessons View Samples

- Code.org CS Fundamentals for Elementary Course A or pre-reader express
- Hello Ruby

- Find CS unplugged activities to introduce Algorithms & Programs here.
- Find lesson plans and instructional materials to support Unit 2 in Lesson Resources.
- Browse books that introduce CS concepts in the Reading List.

Sources & Attribution

Adapted primarily from Creative Commons licensed resources developed by

- Code.org
- Google CS First
- Common Sense Education
- Australia Digital Technologies Hub
- MIT Media Lab's Lifelong Kindergarten Group
- CSinSF.org
- NYC CS for All
- Hello Ruby
- CSTA CS K-12 Standards
- K-12 CS Framework
- Apple Everyone Can Code
- CS Unplugged

1 -1: Networks and Life

Enduring Understanding

Computer networks can be used to connect people to other people, places, information, and ideas. The Internet enables people to connect with others worldwide through many different points of connection. Connecting devices to a network or the Internet provides great benefit, care must be taken to use authentication measures, such as strong passwords, to protect devices and information from unauthorized access. People use computing technology in ways that can help or hurt themselves or others. Harmful behaviors, such as sharing private information and interacting with strangers, should be recognized and avoided.

Essential Questions

What is a network?
How do people connect and exchange info?
How do I stay safe online? Why do we need passwords?
How should I behave online? How do we protect and work respectfully?

Core Concepts:

Network and Internet (Cybersecurity)
Impacts of Computing (Social Interaction; Safety, Law and Ethics)

CSTA Standards

1A-NI-04 Explain what passwords are and why we use them, and use strong passwords to protect devices and information from unauthorized access. (P7.3)
Learning to protect one's device or information from unwanted use by others is an essential first step in learning about cybersecurity. Students are not required to use multiple strong passwords. They should appropriately use and protect the passwords they are required to use.

1A-IC-18 Keep login information private, and log off of devices appropriately. (P7.3)
People use computing technology in ways that can help or hurt themselves or others. Harmful behaviors, such as sharing private information and leaving public devices logged in should be recognized and avoided.

1A-IC-17 Work respectfully and responsibly with others online. (P2.1)
Online communication facilitates positive interactions, such as sharing ideas with many people, but the public and anonymous nature of online communication also allows intimidating and inappropriate behavior in the form of cyberbullying. Students could

share their work on blogs or in other collaborative spaces online, taking care to avoid sharing information that is inappropriate or that could personally identify them to others. Students could provide feedback to others on their work in a kind and respectful manner and could tell an adult if others are sharing things they

Related Resources and Toolkits

- DIgital Citizenship Curriculum from Common Sense Education
- Be Internet Awesome Curriculum from Google
- NetSmartz, online safety education program from NCMEC

Sample Lessons View Samples
Lesson 1-2: Going place safely
Lesson 3-4: Your Digital Footprint
Lesson 5-6: Password

- Find lesson plans and instructional materials to support Unit 1 in Lesson Resources.
- Browse books that introduce CS concepts in the Reading List.
- Explore more CS unplugged activities here.

1 -2 : Computers and Algorithms

Enduring Understanding

Algorithms and programming control all computing systems, empowering people to communicate with the world in new ways and solve compelling problems. Algorithms are translated into programs, or code, to provide instructions for computing devices. Algorithms are commonly implemented using a precise language that computers can interpret. Computers follow precise sequences of instructions that automate tasks literally. Program execution can also be non sequential by repeating patterns of instructions and using events to initiate instructions.

Essential Questions

What are loops and events in real world?
How does a computer trigger an action?
What are the strategies of fixing problems and debugging?

Core Concepts:

Algorithms and Programming

CSTA Standards

1A-AP-09 - Model the way programs store and manipulate data by using numbers or other symbols to represent information.

Information in the real world can be represented in computer programs. Students could use thumbs up/down as representations of yes/no, use arrows when writing algorithms to represent direction, or encode and decode words using numbers, pictographs, or other symbols to represent letters or words.

1A-AP-10 - Develop programs with sequences and simple loops, to express ideas or address a problem.

Programming is used as a tool to create products that reflect a wide range of interests. Control structures specify the order in which instructions are executed within a program. Sequences are the order of instructions in a program. For example, if dialogue is not sequenced correctly when programming a simple animated story, the story will not make sense. If the commands to program a robot are not in the correct order, the robot will not complete the task desired. Loops allow for the repetition of a sequence of code multiple times. For example, in a program to show the life cycle of a butterfly, a loop could be combined with move commands to allow continual but controlled movement of the character.

1A-AP-11 - Decompose (break down) the steps needed to solve a problem into a

precise sequence of instructions.
Decomposition is the act of breaking down tasks into simpler tasks. Students could break down the steps needed to make a peanut butter and jelly sandwich, to brush their teeth, to draw a shape, to move a character across the screen, or to solve a level of a coding app.

1A-AP-14 - Debug (identify and fix) errors in an algorithm or program that includes sequences and simple loops.
Algorithms or programs may not always work correctly. Students should be able to use various strategies, such as changing the sequence of the steps, following the algorithm in a step-by-step manner, or trial and error to fix problems in algorithms and programs.

Related Resources and Toolkits

Curriculum

- Code.org: Pre-reader Express; CS Fundamentals Course B
- Apple: Everyone Can Code: Get Started with Code 1 (Tynker; CodeSpark); Everyone Can Code Puzzles.
- BootUp curriculum (K-2 with Scratch Jr.)

Physical Computing Extensions

BeeBot
KIBO
Puzzlets
Dash and Dot
Osmo
Ozobot

Programming Platforms

Scratch Jr.
CodeMonkey
Kodable
CodeSpark (The Foos)
Tynker

Sample Lessons View Samples

Code.org CS Fundamentals Course B
Lesson 1: Steve and The big project
Lesson 2: My loopy Robotics Friends Jr.
Lesson 3: Move it, Move it
Lesson 4 Loops with Scrat
Lesson 5 Loops with Laurel
Lesson 6 Drawing Gardens with Loops

Lesson 7 The Big Event Jr. (Hello Ruby- Big Events)
Lesson 8 A Royal Battle with Events

- Find CS unplugged activities to introduce Algorithms & Programs here.
- Find lesson plans and instructional materials to support Unit 2 in Lesson Resources.
- Browse books that introduce CS concepts in the Reading List.

Sources & Attribution

Adapted primarily from Creative Commons licensed resources developed by

- Code.org
- Google CS First
- Common Sense Education
- Australia Digital Technologies Hub
- MIT Media Lab's Lifelong Kindergarten Group
- CSinSF.org
- NYC CS for All
- Hello Ruby
- CSTA CS K-12 Standards
- K-12 CS Framework
- Apple Everyone Can Code
- CS Unplugged

2 -1: Computer Systems, Data, and Life

Enduring Understanding

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

Essential Questions

What can computers help me to do?
What are software and hardware in a computing system?
What role does software play when using a computer?
What is data and how is the information stored?
What can we do when computers have problems?
How can data and information affect our life?
How can computers affect lifestyles and relationships?

Core Concepts:

Computing Systems (Hardware & Software; Troubleshooting)
Data and Analysis (Storage; Collection, Visualization & Transformation; Inference & Models)

CSTA Standards

1A-CS-01 Select and operate appropriate software to perform a variety of tasks, and recognize that users have different needs and preferences for the technology they use. (P1.1)

People use computing devices to perform a variety of tasks accurately and quickly. Students should be able to select the appropriate app/program to use for tasks they are required to complete. For example, if students are asked to draw a picture, they should be able to open and use a drawing app/program to complete this task, or if they are asked to create a presentation, they should be able to open and use presentation software. In addition, with teacher guidance, students should compare and discuss preferences for software with the same primary functionality. Students could compare different web browsers or word processing, presentation, or drawing programs.

1A-IC-16 Compare how people live and work before and after the implementation

or adoption of new computing technology. (P7.0)
Computing technology has positively and negatively changed the way people live and work. In the past, if students wanted to read about a topic, they needed access to a library to find a book about it. Today, students can view and read information on the Internet about a topic or they can download e-books about it directly to a device. Such information may be available in more than one language and could be read to a student, allowing for great accessibility.

1A-CS-03 Describe basic hardware and software problems using accurate terminology. (P6.2, P7.2)
Problems with computing systems have different causes. Students at this level do not need to understand those causes, but they should be able to communicate a problem with accurate terminology (e.g., when an app or program is not working as expected, a device will not turn on, the sound does not work, etc.). Ideally, students would be able to use simple troubleshooting strategies, including turning a device o and on to reboot it, closing and reopening an app, turning on speakers, or plugging in headphones. These are, however, not specified in the standard, because these problems may not occur.

1A-DA-05 Store, copy, search, retrieve, modify, and delete information using a computing device and define the information stored as data. (P4.2)
All information stored and processed by a computing device is referred to as data. Data can be images, text documents, audio files, software programs or apps, video files, etc. As students use software to complete tasks on a computing device, they will be manipulating data.

1A-DA-06 Collect and present the same data in various visual formats. (P7.1, P4.4)
The collection and use of data about the world around them is a routine part of life and influences how people live. Students could collect data on the weather, such as sunny days versus rainy days, the temperature at the beginning of the school day and end of the school day, or the inches of rain over the course of a storm. Students could count the number of pieces of each color of candy in a bag of candy, such as Skittles or M&Ms. Students could create surveys of things that interest them, such as favorite foods, pets, or TV shows, and collect answers to their surveys from their peers and others. The data collected could then be organized into two or more visualizations, such as a bar graph, pie chart, or pictograph.

1A-DA-07 Identify and describe patterns in data visualizations, such as charts or graphs, to make predictions. (P4.1)
Data can be used to make inferences or predictions about the world. Students could analyze a graph or pie chart of the colors in a bag of candy or the averages for colors

in multiple bags of candy, identify the patterns for which colors are most and least represented, and then make a prediction as to which colors will have most and least in a new bag of candy. Students could analyze graphs of temperatures taken at the beginning of the school day and end of the school day, identify the patterns of when temperatures rise and fall, and predict if they think the temperature will rise or fall at a particular time of the day, based on the pattern observed.

Related Resources and Toolkits

- How Computers Work, video playlist from Code.org

Sample Lessons View Samples

Lesson 1-2 : Computer Run Software
Lesson 3-4: Past, Presence and Future
Lesson 5-6: What is Data?

- Find lesson plans and instructional materials to support Unit 1 in Lesson Resources.
- Browse books that introduce CS concepts in the Reading List.
- Explore more CS unplugged activities here.

2 -2: Program Development

Enduring Understanding

People develop programs collaboratively for a purpose, such as expressing ideas or addressing problems. People work together to plan, create, and test programs within a context that is relevant to the programmer and users. Programming is used as a tool to create products that reflect a wide range of interests, such as video games, interactive art projects, and digital stories. Complex tasks can be broken down into simpler instructions, some of which can be broken down even further. Likewise, instructions can be combined to accomplish complex tasks.

Essential Questions

How and why do people develop a program?
What does decomposition look like in real world?
What are loops, events and conditionals in real world?

Core Concepts:

Algorithms and Programming

CSTA Standards

1A-AP-08 - Model daily processes by creating and following algorithms (sets of step-by-step instructions) to complete tasks.
Composition is the combination of smaller tasks into more complex tasks. Students could create and follow algorithms for making simple foods, brushing their teeth, getting ready for school, participating in clean-up time.

1A-AP-09 - Model the way programs store and manipulate data by using numbers or other symbols to represent information.
Information in the real world can be represented in computer programs. Students could use thumbs up/down as representations of yes/no, use arrows when writing algorithms to represent direction, or encode and decode words using numbers, pictographs, or other symbols to represent letters or words.

1A-AP-10 - Develop programs with sequences and simple loops, to express ideas or address a problem.
Programming is used as a tool to create products that reflect a wide range of interests. Control structures specify the order in which instructions are executed within a program. Sequences are the order of instructions in a program. For example, if dialogue

is not sequenced correctly when programming a simple animated story, the story will not make sense. If the commands to program a robot are not in the correct order, the robot will not complete the task desired. Loops allow for the repetition of a sequence of code multiple times. For example, in a program to show the life cycle of a butterfly, a loop could be combined with move commands to allow continual but controlled movement of the character.

1A-AP-11 - Decompose (break down) the steps needed to solve a problem into a precise sequence of instructions.
Decomposition is the act of breaking down tasks into simpler tasks. Students could break down the steps needed to make a peanut butter and jelly sandwich, to brush their teeth, to draw a shape, to move a character across the screen, or to solve a level of a coding app.

1A-AP-12 Develop plans that describe a program's sequence of events, goals, and expected outcomes. (P5.1, P7.2)
Creating a plan for what a program will do clarifies the steps that will be needed to create a program and can be used to check if a program is correct. Students could create a planning document, such as a story map, a storyboard, or a sequential graphic organizer, to illustrate what their program will do. Students at this stage may complete the planning process with help from their teachers.

1A-AP-13 Give attribution when using the ideas and creations of others while developing programs. (P7.3)
Using computers comes with a level of responsibility. Students should credit artifacts that were created by others, such as pictures, music, and code. Credit could be given orally, if presenting their work to the class, or in writing or orally, if sharing work on a class blog or website. Proper attribution at this stage does not require a formal citation, such as in a bibliography or works cited document.

1A-AP-14 - Debug (identify and fix) errors in an algorithm or program that includes sequences and simple loops.
Algorithms or programs may not always work correctly. Students should be able to use various strategies, such as changing the sequence of the steps, following the algorithm in a step-by-step manner, or trial and error to fix problems in algorithms and programs.

1A-AP-15 Using correct terminology, describe steps taken and choices made during the iterative process of program development. (P7.2)
At this stage, students should be able to talk or write about the goals and expected outcomes of the programs they create and the choices that they made when creating programs. This could be done using coding journals, discussions with a teacher, class

presentations, or blogs.

Related Resources and Toolkits

Curriculum

Code.org: Pre-reader Express; CS Fundamentals Course C
Apple: Everyone Can Code: Get Started with Code 2 (Tynker; CodeSpark); Everyone Can Code Puzzles.
BootUp curriculum (K-2 with Scratch Jr.)

Programming Platforms

Scratch Jr.
CodeMonkey
Kodable
CodeSpark (The Foos)
Tynker

Physical Computing Extensions

BeeBot
KIBO
Puzzlets
Dash and Dot
Osmo
Ozobot

Sample Lessons View Samples

Lesson 1 Getting Loopy
Lesson 2 Loops with Rey and BB-8
Lesson 3 Sticker Art with Loops
Lesson 4 Harvesting Crops with Loops
Lesson 5 The Big Event
Lesson 6 Build a Flappy Game
Lesson 7 Chase Game with Events

- Find CS unplugged activities to introduce Algorithms & Programs here.
- Find lesson plans and instructional materials to support Unit 2 in Lesson Resources.
- Browse books that introduce CS concepts in the Reading List.

Sources & Attribution

Adapted primarily from Creative Commons licensed resources developed by

Code.org
Google CS First
Common Sense Education
Australia Digital Technologies Hub
MIT Media Lab's Lifelong Kindergarten Group
CSinSF.org
NYC CS for All
Hello Ruby
CSTA CS K-12 Standards
K-12 CS Framework
Apple Everyone Can Code
CS Unplugged

3 -1: The Connected World

Enduring Understanding

Computing devices may be connected to other devices or components to extend their capabilities. Together, devices and components will form a system for a purpose. Information needs connections to be sent or received, which can take many forms, such as physical or wireless. Information transmitted over the network can be protected using various security measures.

Essential Questions

Why do computers need to connect to other devices?
What do peripheral devices do?
What are input and output devices?
How are computers connected (physical paths or wirelessly, Wifi, bluetooth)?
What can you do with the Internet?
How and why do computers communicate with each other?
What are some cybersecurity issues in the real world ?
What are some ways to protect information?
How do we keep personal information private?
What can you do about cyberbullying?

Core Concepts:

Computing Systems (Devices; Troubleshooting)
Network and Internet (Network Communication & Organization; Cybersecurity)
Impacts of Computing (Safety, Law and Ethics)

CSTA Standards

1B-CS-01 Describe how internal and external parts of computing devices function to form a system. (P7.2)
Computing devices often depend on other devices or components. For example, a robot depends on a physically attached light sensor to detect changes in brightness, whereas the light sensor depends on the robot for power. Keyboard input or a mouse click could cause an action to happen or information to be displayed on a screen; this could only happen because the computer has a processor to evaluate what is happening externally and produce corresponding responses. Students should describe how devices and components interact using correct terminology.

1B-CS-03 Determine potential solutions to solve simple hardware and software problems using common troubleshooting strategies. (P6.2)

Although computing systems may vary, common troubleshooting strategies can be used on all of them. Students should be able to identify solutions to problems such as the device not responding, no power, no network, app crashing, no sound, or password entry not working. Should errors occur at school, the goal would be that students would use various strategies, such as rebooting the device, checking for power, checking network availability, closing and reopening an app, making sure speakers are turned on or headphones are plugged in, and making sure that the caps lock key is not on, to solve these problems, when possible.

1B-NI-04 Model how information is broken down into smaller pieces, transmitted as packets through multiple devices over networks and the Internet, and reassembled at the destination. (P4.4)
Information is sent and received over physical or wireless paths. It is broken down into smaller pieces called packets, which are sent independently and reassembled at the destination. Students should demonstrate their understanding of this flow of information by, for instance, drawing a model of the way packets are transmitted, programming an animation to show how packets are transmitted, or demonstrating this through an unplugged activity which has them act it out in some way.

1B-NI-05 Discuss real-world cybersecurity problems and how personal information can be protected. (P3.1)
Just as we protect our personal property offline, we also need to protect our devices and the information stored on them. Information can be protected using various security measures. These measures can be physical and/or digital. Students could discuss or use a journaling or blogging activity to explain, orally or in writing, about topics that relate to personal cybersecurity issues. Discussion topics could be based on current events related to cybersecurity or topics that are applicable to students, such as the necessity of backing up data to guard against loss, how to create strong passwords and the importance of not sharing passwords, or why we should install and keep anti-virus software updated to protect data and systems.

Related Resources and Toolkits

- How Computers Work, video playlist from Code.org
- DIgital Citizenship Curriculum from Common Sense Education
- Be Internet Awesome Curriculum from Google
- NetSmartz, online safety education program from NCMEC

Sample Lessons

Lesson 1: Input Devices
Lesson 2: Output Devices

Lesson 3: Processing Inputs to Create Outputs
Lesson 4 Private and Personal Information
Lesson 5: Screen Out the Mean
Lesson 6: The power of words

- Find lesson plans and instructional materials to support Unit 1 in Lesson Resources.
- Browse books that introduce CS concepts in the Reading List.
- Explore more CS unplugged activities here.

3 -2: Introduction to Programming

Enduring Understanding

Program development is an iterative process, involving three stages: design, implementation and review. The planned design is expressed and implemented in a programming language. Decomposition facilitates people to think different aspects of program development.

Essential Questions

How do we break down problems into manageable parts?
What needs to be included when we create a project (such as, scenes, sprites to placed, background to be selected, programmed actions)?
What are some basic sequence and control structures (loops, event handler, conditionals)?

Core Concepts:

Algorithms and Programming

CSTA Standards

1B-AP-10 Create programs that include sequences, events, loops, and conditionals. (P5.2)
Control structures specify the order (sequence) in which instructions are executed within a program and can be combined to support the creation of more complex programs. Events allow portions of a program to run based on a specific action. For example, students could write a program to explain the water cycle and when a specific component is clicked (event), the program would show information about that part of the water cycle. Conditionals allow for the execution of a portion of code in a program when a certain condition is true. For example, students could write a math game that asks multiplication fact questions and then uses a conditional to check whether or not the answer that was entered is correct. Loops allow for the repetition of a sequence of code multiple times. For example, in a program that produces an animation about a famous historical character, students could use a loop to have the character walk across the screen as they introduce themselves.

1B-AP-11 Decompose (break down) problems into smaller, manageable subproblems to facilitate the program development process. (P3.2)
Decomposition is the act of breaking down tasks into simpler tasks. For example, students could create an animation by separating a story into different scenes. For each scene, they would select a background, place characters, and program actions.

1B-AP-15 Test and debug (identify and fix errors) a program or algorithm to ensure it runs as intended. (P6.1, P6.2)
As students develop programs they should continuously test those programs to see that they do what was expected and fix (debug), any errors. Students should also be able to successfully debug simple errors in programs created by others.

Related Resources and Toolkits

Curriculum

- Code.org: Express; CS Fundamentals Course D
- Apple: Everyone Can Code: Learn how to code (Swift Playgrounds);
- Google CS First: (Introductory Level: Storytelling, Music and Sound)
- Creative Computing Curriculum
- BootUP Curriculum (G3-6 with Scratch)

Programming Platforms

- Scratch
- MakeCode
- Snap!
- Blockly
- Swift Playgrounds (App)

Physical Computing Extensions

- KIBO
- Puzzlets
- Dash and Dot
- Ozobot

Sample Lessons

Google CS First Storytelling
Activity 1: Dialogue
Activity 2: Check It Out
Activity 3: Setting
Activity 4: Premise
Activity 5: Characterization
Activity 6: Interactive Storytelling
Activity 7: Personal Narrative
Activity 8: Your Innovation Story

- Find CS unplugged activities to introduce Algorithms & Programs here.
- Find lesson plans and instructional materials to support Unit 2 in Lesson Resources.
- Browse books that introduce CS concepts in the Reading List.

4 -1: Technology, Collaboration, and Cultural Diversity

Enduring Understanding
New technology is created and driven by people's needs and wants. Computing technology also allows collaboration locally and across the globe at different times. By communication and innovation, computing influences and are influenced by social interaction and cultural practices. However, the development of computing the Internet brings the ease of sending and receiving copies of digital media but also creates opportunities for unauthorized use and ethical complications.

Essential Questions
How is information transmitted and received over the Internet?
How do we collaborate online?
How computing technologies change the world and practices?
What are the ways to improve accessibility and usability of technology products?
What are some career that utilize computing and technology?
What' the effect of computing on society within economic, social, and cultural contexts?
What is copyright?
Why do we need copyright?
What is public domain and creative common?
How can we use materials found online legally?

Core Concepts:
Network and Internet (Network Communication & Organization)
Impacts of Computing (Culture; Social Interaction; Safety, Law and Ethics)

CSTA Standards
1B-NI-04 Model how information is broken down into smaller pieces, transmitted as packets through multiple devices over networks and the Internet, and reassembled at the destination. (P4.4)
Information is sent and received over physical or wireless paths. It is broken down into smaller pieces called packets, which are sent independently and reassembled at the destination. Students should demonstrate their understanding of this flow of information by, for instance, drawing a model of the way packets are transmitted, programming an animation to show how packets are transmitted, or demonstrating this through an unplugged activity which has them act it out in some way.
1B-IC-18 Discuss computing technologies that have changed the world, and express how those technologies influence, and are influenced by, cultural practices. (P7.1)

New computing technology is created and existing technologies are modified for many reasons, including to increase their benefits, decrease their risks, and meet societal needs. Students, with guidance from their teacher, should discuss topics that relate to the history of technology and the changes in the world due to technology. Topics could be based on current news content, such as robotics, wireless Internet, mobile computing devices, GPS systems, wearable computing, or how social media has influenced social and political changes.

1B-IC-19 Brainstorm ways to improve the accessibility and usability of technology products for the diverse needs and wants of users. (P1.2)

The development and modification of computing technology are driven by people's needs and wants and can affect groups differently. Anticipating the needs and wants of diverse end users requires students to purposefully consider potential perspectives of users with different backgrounds, ability levels, points of view, and disabilities. For example, students may consider using both speech and text when they wish to convey information in a game. They may also wish to vary the types of programs they create, knowing that not everyone shares their own tastes.

1B-IC-20 Seek diverse perspectives for the purpose of improving computational artifacts. (P1.1)

Computing provides the possibility for collaboration and sharing of ideas and allows the benefit of diverse perspectives. For example, students could seek feedback from other groups in their class or students at another grade level. Or, with guidance from their teacher, they could use video conferencing tools or other online collaborative spaces, such as blogs, wikis, forums, or website comments, to gather feedback from individuals and groups about programming projects.

1B-IC-21 Use public domain or creative commons media, and refrain from copying or using material created by others without permission. (P7.3)

Ethical complications arise from the opportunities provided by computing. The ease of sending and receiving copies of media on the Internet, such as video, photos, and music, creates the opportunity for unauthorized use, such as online piracy, and disregard of copyrights. Students should consider the licenses on computational artifacts that they wish to use. For example, the license on a downloaded image or audio file may have restrictions that prohibit modification, require attribution, or prohibit use entirely.

Related Resources and Toolkits

- DIgital Citizenship Curriculum from Common Sense Education
- Be Internet Awesome Curriculum from Google
- NetSmartz, online safety education program from NCMEC
- Resources: How The Internet Works

Sample Lessons

Lesson 1-2 The Internet
Lesson 3-4 Changes in Technology
Lesson 5-6 Accessibility and usability
Lesson 7-8 Digital Citizenship

- Find lesson plans and instructional materials to support Unit 1 in Lesson Resources.
- Browse books that introduce CS concepts in the Reading List.
- Explore more CS unplugged activities here.

4 -2: Computational Participation

Enduring Understanding

Program development is an iterative process. Repeating these steps enables people to improve programs. The design stages is a planning stage where programmers sketch out a solution before coding. People often work collaboratively and reuse existing code or remixing other projects within a community and give appropriate attributions. The community is created by people who share and provide feedback on one another's creations. People continuously review whether programs work as expected and they fix or debug.

Essential Questions

How and why do we remix projects online?
How and why do people work collaboratively on programming?
How do we debug an error if programs do not run as intended?

Core Concepts:

Algorithms and Programming

CSTA Standards

1B-AP-08 Compare and refine multiple algorithms for the same task and determine which is the most appropriate. (P6.3, P3.3)
Different algorithms can achieve the same result, though sometimes one algorithm might be most appropriate for a specific situation. Students should be able to look at different ways to solve the same task and decide which would be the best solution. For example, students could use a map and plan multiple algorithms to get from one point to another. They could look at routes suggested by mapping software and change the route to something that would be better, based on which route is shortest or fastest or would avoid a problem. Students might compare algorithms that describe how to get ready for school. Another example might be to write different algorithms to draw a regular polygon and determine which algorithm would be the easiest to modify or repurpose to draw a different polygon.
1B-AP-10 Create programs that include sequences, events, loops, and conditionals. (P5.2)
Control structures specify the order (sequence) in which instructions are executed within a program and can be combined to support the creation of more complex programs. Events allow portions of a program to run based on a specific action. For example, students could write a program to explain the water cycle and when a specific component is clicked (event), the program would show information about that part of the water cycle. Conditionals allow for the execution of a portion of code in a program

when a certain condition is true. For example, students could write a math game that asks multiplication fact questions and then uses a conditional to check whether or not the answer that was entered is correct. Loops allow for the repetition of a sequence of code multiple times. For example, in a program that produces an animation about a famous historical character, students could use a loop to have the character walk across the screen as they introduce themselves.

1B-AP-11 Decompose (break down) problems into smaller, manageable subproblems to facilitate the program development process. (P3.2)

Decomposition is the act of breaking down tasks into simpler tasks. For example, students could create an animation by separating a story into different scenes. For each scene, they would select a background, place characters, and program actions.

1B-AP-12 Modify, remix, or incorporate portions of an existing program into one's own work, to develop something new or add more advanced features. (P5.3)

Programs can be broken down into smaller parts, which can be incorporated into new or existing programs. For example, students could modify prewritten code from a single-player game to create a two-player game with slightly different rules, remix and add another scene to an animated story, use code to make a ball bounce from another program in a new basketball game, or modify an image created by another student.

1B-AP-13 Use an iterative process to plan the development of a program by including others' perspectives and considering user preferences. (P1.1, P5.1)

Planning is an important part of the iterative process of program development. Students outline key features, time and resource constraints, and user expectations. Students should document the plan as, for example, a storyboard, flowchart, pseudocode, or story map.

1B-AP-14 Observe intellectual property rights and give appropriate attribution when creating or remixing programs. (P7.3)

Intellectual property rights can vary by country but copyright laws give the creator of a work a set of rights that prevents others from copying the work and using it in ways that they may not like. Students should identify instances of remixing, when ideas are borrowed and iterated upon, and credit the original creator. Students should also consider common licenses that place limitations or restrictions on the use of computational artifacts, such as images and music downloaded from the Internet. At this stage, attribution should be written in the format required by the teacher and should always be included on any programs shared online.

1B-AP-15 Test and debug (identify and fix errors) a program or algorithm to ensure it runs as intended. (P6.1, P6.2)

As students develop programs they should continuously test those programs to see that they do what was expected and fix (debug), any errors. Students should also be able to successfully debug simple errors in programs created by others.

1B-AP-16 Take on varying roles, with teacher guidance, when collaborating with peers during the design, implementation, and review stages of program development. (P2.2)

Collaborative computing is the process of performing a computational task by working in pairs or on teams. Because it involves asking for the contributions and feedback of others, effective collaboration can lead to better outcomes than working independently. Students should take turns in different roles during program development, such as note taker, facilitator, program tester, or “driver” of the computer.
1B-AP-17 Describe choices made during program development using code comments, presentations, and demonstrations. (P7.2)
People communicate about their code to help others understand and use their programs. Another purpose of communicating one's design choices is to show an understanding of one's work. These explanations could manifest themselves as in-line code comments for collaborators and assessors, or as part of a summative presentation, such as a code walk-through or coding journal.

Related Resources and Toolkits

Curriculum

- Code.org: Express; CS Fundamentals Course E
- Apple: Everyone Can Code: Learn how to code 1 and 2(Swift Playgrounds);
- Google CS First: (Basic or Intermediate Level: Storytelling, Music, Art, Friends, Fashion and Design)
- Creative Computing Curriculum
- BootUP Curriculum (G3-6 with Scratch)

Programming Platforms

- Scratch
- MakeCode
- Snap!
- Blockly
- Swift Playgrounds (App)

Physical Computing Extensions

- KIBO
- Puzzlets
- Dash and Dot
- Ozobot
- Makey Makey
- Micro:bit

Sample Lessons

Google CS First: Multiple-Day Activity: Music
Creative Computing Curriculum

- Find CS unplugged activities to introduce Algorithms & Programs here.

- Find lesson plans and instructional materials to support Unit 2 in Lesson Resources.
- Browse books that introduce CS concepts in the Reading List.

5 -1: Computers and Information

Enduring Understanding

Hardware and software work together to accomplish tasks, such as sending, collecting, receiving, processing and storing units of information. In a computing system, information is stored and represented as bits, the basic unit of data. Data is organized, sorted, and transformed to provide clarity and to emphasize aspects. People use data to highlight relationships, make Inferences and predictions.

Essential Questions

How do hardware and software work together?
What's inside a computer?
How is data stored in computers?
How can we convert real-life things (ex, pictures) that we want to store in a computer?
What is bit and binary system?
What are some applications in real life ?
How do we use software and hardware to collect data?
How do we use data to solve a problem?
What are infographics, charts and graphs?

Core Concepts:

Computing Systems (Hardware and Software)
Data and Analysis (Collection, Visualization & Transformation; Inference & Models)
Impacts of Computing (Social Interaction)

CSTA Standards

1B-CS-02 Model how computer hardware and software work together as a system to accomplish tasks. (P4.4)
In order for a person to accomplish tasks with a computer, both hardware and software are needed. At this stage, a model should only include the basic elements of a computer system, such as input, output, processor, sensors, and storage. Students could draw a model on paper or in a drawing program, program an animation to demonstrate it, or demonstrate it by acting this out in some way.
1B-CS-01 Describe how internal and external parts of computing devices function to form a system. (P7.2)
Computing devices often depend on other devices or components. For example, a robot depends on a physically attached light sensor to detect changes in brightness, whereas the light sensor depends on the robot for power. Keyboard input or a mouse click could cause an action to happen or information to be displayed on a screen; this could only

happen because the computer has a processor to evaluate what is happening externally and produce corresponding responses. Students should describe how devices and components interact using correct terminology.
1B-DA-06 Organize and present collected data visually to highlight relationships and support a claim. (P7.1)
Raw data has little meaning on its own. Data is often sorted or grouped to provide additional clarity. Organizing data can make interpreting and communicating it to others easier. Data points can be clustered by a number of commonalities. The same data could be manipulated in different ways to emphasize particular aspects or parts of the data set. For example, a data set of sports teams could be sorted by wins, points scored, or points allowed, and a data set of weather information could be sorted by high temperatures, low temperatures, or precipitation.
1B-DA-07 Use data to highlight or propose cause and-effect relationships, predict outcomes, or communicate an idea. (P7.1)
The accuracy of data analysis is related to how realistically data is represented. Inferences or predictions based on data are less likely to be accurate if the data is not sufficient or if the data is incorrect in some way. Students should be able to refer to data when communicating an idea. For example, in order to explore the relationship between speed, time, and distance, students could operate a robot at uniform speed, and at increasing time intervals to predict how far the robot travels at that speed. In order to make an accurate prediction, one or two attempts of differing times would not be enough. The robot may also collect temperature data from a sensor, but that data would not be relevant for the task. Students must also make accurate measurements of the distance the robot travels in order to develop a valid prediction. Students could record the temperature at noon each day as a basis to show that temperatures are higher in certain months of the year. If temperatures are not recorded on non-school days or are recorded incorrectly or at different times of the day, the data would be incomplete and the ideas being communicated could be inaccurate. Students may also record the day of the week on which the data was collected, but this would have no relevance to whether temperatures are higher or lower. In order to have sufficient and accurate data on which to communicate the idea, students might want to use data provided by a governmental weather agency.

Related Resources and Toolkits

- How Computers Work, video playlist from Code.org
- Data Visualization- Infographic Tools: Venngage, Visme, Canva, Piktochart
- Data Tools: Google Sheets, Google Forms (Google's Applied Digital Skills Curriculum)

Sample Lessons

Lesson #1 What makes something a computer?
Lesson #2 Hardware and software
Lesson #3 Binary Bracelets
Lesson #4 Binary Images
Lesson #5 What is Data?
Lesson #6 Data and Processing
Lesson #7 Collecting Data
Lesson #8 Data to answer questions
Lesson #9 Data Visualization Project

- Find lesson plans and instructional materials to support Unit 1 in Lesson Resources.
- Browse books that introduce CS concepts in the Reading List.
- Explore more CS unplugged activities here.

5 -2: Programming and Problem Solving

Enduring Understanding

People write programs to solve problems. Variables are the vehicle through which computer programs store different types of data.

Essential Questions

What is variables in real world?
Why do we need "for loops", variables, and functions?
How do we include "for loops", variables, and functions in projects?

Core Concepts:

Algorithms and Programming

CSTA Standards

1B-AP-08 Compare and refine multiple algorithms for the same task and determine which is the most appropriate. (P6.3, P3.3)
Different algorithms can achieve the same result, though sometimes one algorithm might be most appropriate for a specific situation. Students should be able to look at different ways to solve the same task and decide which would be the best solution. For example, students could use a map and plan multiple algorithms to get from one point to another. They could look at routes suggested by mapping software and change the route to something that would be better, based on which route is shortest or fastest or would avoid a problem. Students might compare algorithms that describe how to get ready for school. Another example might be to write different algorithms to draw a regular polygon and determine which algorithm would be the easiest to modify or repurpose to draw a different polygon.
1B-AP-09 Create programs that use variables to store and modify data. (P5.2)
Variables are used to store and modify data. At this level, understanding how to use variables is sufficient. For example, students may use mathematical operations to add to the score of a game or subtract from the number of lives available in a game. The use of a variable as a countdown timer is another example.
1B-AP-10 Create programs that include sequences, events, loops, and conditionals. (P5.2)
Control structures specify the order (sequence) in which instructions are executed within a program and can be combined to support the creation of more complex programs. Events allow portions of a program to run based on a specific action. For example, students could write a program to explain the water cycle and when a specific component is clicked (event), the program would show information about that part of the water cycle. Conditionals allow for the execution of a portion of code in a program

when a certain condition is true. For example, students could write a math game that asks multiplication fact questions and then uses a conditional to check whether or not the answer that was entered is correct. Loops allow for the repetition of a sequence of code multiple times. For example, in a program that produces an animation about a famous historical character, students could use a loop to have the character walk across the screen as they introduce themselves.

1B-AP-11 Decompose (break down) problems into smaller, manageable subproblems to facilitate the program development process. (P3.2)

Decomposition is the act of breaking down tasks into simpler tasks. For example, students could create an animation by separating a story into different scenes. For each scene, they would select a background, place characters, and program actions.

1B-AP-12 Modify, remix, or incorporate portions of an existing program into one's own work, to develop something new or add more advanced features. (P5.3)

Programs can be broken down into smaller parts, which can be incorporated into new or existing programs. For example, students could modify prewritten code from a single-player game to create a two-player game with slightly different rules, remix and add another scene to an animated story, use code to make a ball bounce from another program in a new basketball game, or modify an image created by another student.

1B-AP-13 Use an iterative process to plan the development of a program by including others' perspectives and considering user preferences. (P1.1, P5.1)

Planning is an important part of the iterative process of program development. Students outline key features, time and resource constraints, and user expectations. Students should document the plan as, for example, a storyboard, flowchart, pseudocode, or story map.

1B-AP-14 Observe intellectual property rights and give appropriate attribution when creating or remixing programs. (P7.3)

Intellectual property rights can vary by country but copyright laws give the creator of a work a set of rights that prevents others from copying the work and using it in ways that they may not like. Students should identify instances of remixing, when ideas are borrowed and iterated upon, and credit the original creator. Students should also consider common licenses that place limitations or restrictions on the use of computational artifacts, such as images and music downloaded from the Internet. At this stage, attribution should be written in the format required by the teacher and should always be included on any programs shared online.

1B-AP-15 Test and debug (identify and fix errors) a program or algorithm to ensure it runs as intended. (P6.1, P6.2)

As students develop programs they should continuously test those programs to see that they do what was expected and fix (debug), any errors. Students should also be able to successfully debug simple errors in programs created by others.

1B-AP-16 Take on varying roles, with teacher guidance, when collaborating with peers during the design, implementation, and review stages of program development. (P2.2)

Collaborative computing is the process of performing a computational task by working in pairs or on teams. Because it involves asking for the contributions and feedback of others, effective collaboration can lead to better outcomes than working independently. Students should take turns in different roles during program development, such as note taker, facilitator, program tester, or “driver” of the computer.
1B-AP-17 Describe choices made during program development using code comments, presentations, and demonstrations. (P7.2)
People communicate about their code to help others understand and use their programs. Another purpose of communicating one's design choices is to show an understanding of one's work. These explanations could manifest themselves as in-line code comments for collaborators and assessors, or as part of a summative presentation, such as a code walk-through or coding journal.

Related Resources and Toolkits

Curriculum

- Code.org: Express; CS Fundamentals Course F
- Apple: Everyone Can Code: Learn how to code 3(Swift Playgrounds);
- Google CS First: (Intermediate or advanced Levels)
- Creative Computing Curriculum
- BootUP Curriculum (G3-6 with Scratch)

Programming Platforms

- Scratch
- MakeCode
- Snap!
- Blockly
- Swift Playgrounds (App)

Physical Computing Extensions

- Sphero
- Makey Makey
- Dash and Dot
- Micro:bit
- Finch

Sample Lessons

Google CS First : Art
Creative Computing Curriculum

- Find CS unplugged activities to introduce Algorithms & Programs here.
- Find lesson plans and instructional materials to support Unit 2 in Lesson Resources.

- Browse books that introduce CS concepts in the Reading List.

6 -1: The Internet and the Web

Enduring Understanding

Computers send and receive information based on a set of rules called protocols, which define how messages between computers are structured and sent. The information across networks can be protected from unauthorized access and modification in a variety of ways.

Essential Questions

Why do people create websites?
How can I incorporate content I find online into my own webpage?
How do I safely and appropriately make use of the content published on the Internet?

Core Concepts:

Network and Internet (Network Communication & Organization)
Impacts of Computing (Culture; Safety, Law and Ethics)

CSTA Standards

2-IC-20 - Compare tradeoffs associated with computing technologies that affect people's everyday activities and career options.
2-AP-13 - Decompose problems and subproblems into parts to facilitate the design, implementation, and review of programs.
2-IC-23 - Describe tradeoffs between allowing information to be public and keeping information private and secure.
2-AP-16 - Incorporate existing code, media, and libraries into original programs, and give attribution.
2-IC-21 - Discuss issues of bias and accessibility in the design of existing technologies.
2-NI-04 Model the role of protocols in transmitting data across networks and the Internet. (P4.4)

Related Resources and Toolkits

- Code.org CS Discoveries Unit 2
- Intro to HTML/CSS: Making webpages (Khan Academy)
- Exploring Computer Science (ECS) Unit 3- Web Design

Sample Lessons

Lesson 1: Exploring Websites
Lesson 2: Websites for Expression
Lesson 3: Internet
Lesson 4: Digital Footprint

Lesson 5: Intellectual Property and Images
Lesson 6: Source and Search Engines

- Find lesson plans and instructional materials to support Unit 1 in Lesson Resources.
- Browse books that introduce CS concepts in the Reading List.
- Explore more CS unplugged activities here.

6 -2: Website Development

Enduring Understanding

People design meaningful solutions for other by considering the diverse needs and wants of community. Web is used as a medium for creative expression and collecting and sharing information.

Essential Questions

How can text communicate content and structure on a web page?
What strategies can I use when coding to find and fix issues?
How do I modify the appearance and style of my web pages?

Core Concepts:

Algorithms and Programming

CSTA Standards

2-AP-19 - Document programs in order to make them easier to follow, test, and debug.
2-AP-15 - Seek and incorporate feedback from team members and users to refine a solution that meets user needs.
2-AP-16 - Incorporate existing code, media, and libraries into original programs, and give attribution.
2-AP-17 - Systematically test and refine programs using a range of test cases.
2-AP-18 - Distribute tasks and maintain a project timeline when collaboratively developing computational artifacts.

Related Resources and Toolkits

- Code.org CS Discoveries Unit 2
- Intro to HTML/CSS: Making webpages (Khan Academy)
- Exploring Computer Science (ECS) Unit 3- Web Design

Sample Lessons

Lesson 1: Intro to HTML
Lesson 2: Headings
Lesson 3: Lists
Lesson 4: Clean Code and Debugging
Lesson 5: Project - Multi-Page Websites
Lesson 6: Styling Text with CSS
Lesson 7: Styling Elements with CSS
Lesson 8: RGB Colors and Classes
Lesson 9: Project - Personal Portfolio Website

- Find CS unplugged activities to introduce Algorithms & Programs here.
- Find lesson plans and instructional materials to support Unit 2 in Lesson Resources.
- Browse books that introduce CS concepts in the Reading List.